TRIANGULAR μOS 1.42 SDK

FOR

COMMODORE 128

PERSONAL COMPUTER

PROGRAMMERS REFERENCE GUIDE

CONTENTS:

A. INTRODUCTION

Programmer's Reference Guide for TRIANGULAR µOS 1.42 SDK (Software Development Kit) explains technical aspects of TRIANGULAR µOS, a GUI (graphic user interface) operating system for 8-bit Commodore computers.

Goal of creating this system was to develop a GUI for 8-bit Commodore computers with the lowest amount of memory: that is Commodore PET with at least 4 KB of memory. Next it was expanded for Commodore VIC-20 with standard 5 KB of memory and later with more expansion RAM was required. And in subsequent versions µOS was adapted for Commodore 64, CBM-II and Plus/4. This iteration of TRIANGULAR µOS (version 1.42) is designed to run on Commodore 128.

This software was written in Commodore BASIC language (subset of Microsoft BASIC) using CBM prg Studio 4.2.0, and is designed to run on Commodore 128 in its standard C128 40-column mode. This version of TRIANGULAR µOS is designed to support BASIC 7.0 and works in color text mode. Commodore BASIC (a runtime interpreted language similar in basic concept to JAVA RTM or C# CLI) is default language used in 8-bit Commodore computers and also functions as their OS and user interface. In similar fashion to early Microsoft Windows (1.0 to 3.11), µOS sits atop of BASIC and KERNAL (Commodore's kernel) and Commodore DOS, which is implemented in every Commodore disk drives or 3rd party solutions in order to load µOS programs or modules, load/save settings and documents, perform operations on floppy disks and communicate with disk drive(s).

Package contains files:

- *TRIANGULAR uOS 1.42 for Commodore 128 Programmer's Reference Guide.pdf* – this document
- *Source Code* folder with 19 source code files
- *TRIANGULAR uOS 1.42.d81* – empty, preformatted System Disk
- *TRIANGULAR uOS 1.42 Documents.d81* – empty, preformatted Documents Disk
- *TRIANGULAR uOS 1.42 SDK TEMPLATE.d81* – disk with *SDK TEMPLATE* program helping start creating µOS programs

B. WHAT YOU NEED

In order to change and/or compile TRIANGULAR µOS 1.42 from source code, you need to do this using external program like CBM prg Studio 4.2.0 (which was used in development and for compilation of µOS 1.42). Using BASIC 7.0 code on real hardware or emulator is out of question, since source code uses extensive line concatenation (lines up to 255 bytes long). Standard BASIC won't present program lines properly (especially print statements) and its screen/program editor won't be able alter those lines properly either.

Download CBM prg Studio here:
www.ajordison.co.uk

For fast creation and modification of disk content DirMaster is recommended. µOS disks are formatted with custom PETSCII characters in Disk name and Disk ID using DirMaster.

Download DirMaster here:
style64.org/dirmaster

For testing and debugging use either real Commodore 128 or emulator (freeware VICE emulator was used for testing of µOS).

Download VICE emulator here:
vice-emu.sourceforge.io

Commodore 128 emulator VICE must be configured with enabled disk drive that can read 800KB 3.5¨ diskette (.d81 file): recommended CBM 1581*. Also, you should enable joystick. You can easily configure it as Numpad keys:

- Up (8), Down (2), Left (4), Right (6)
- You can move diagonally e.g., Up-Left (7)
- 0 or right Ctrl: Fire (click/select)

You can also enable the printer in the VICE emulator.
Do this in: Settings -> Peripheral devices -> Printers. You can choose printer as device #4 - #7, although #4 is standard and is recommended.

** Using 5.25" disk drives: 1571 (default), alternatively 1541 type drive (1541-II) is possible, but System Disk and Documents disk images first must be converted to .d71 or .d64 file in external program (e.g., DirMaster). Additionally using 1571 disk drive amounts to over twice disk drive speed reduction, while 1541 type drive brings speed to default Commodore 64 levels (~10 times slower than 1581) thus only 1581 type drive is officially supported.*

C. How to compile Triangular µOS 1.42

TRIANGULAR µOS source code is stored in the ***Source code*** folder in ****bas*** with ***.cfg.seq**, ***.dat.seq**, ***.sav.dat**, ***.scr.prg**, ***.scl.prg*** and ***.spr.prg*** files. Segments of programs are commented with simple descriptive caption-like comments:
!- characters at the beginning of the line are used to mark comments.

After compiling all ***.bas*** files in CBM prg Studio into *.prg files, add them to System Disk. Remember to put the ***UOS*** file first (to properly load the system with LOAD "*",8 or DLOAD"*" commands).

Next add ***uos.cfg.seq*** file (it should have SEQ property) and place it in the middle of UOS and GUI (that's µOS convention).

Next add all graphic assets resource ***.spr.prg**, ***.sav.dat**, ***.scr.prg*** and ***.scl.prg*** files.
Lastly, add ***clipboard.dat.seq*** file (this is file where system apps stories copied data).

You can use empty, preformatted System Disk file to speed up this process: ***TRIANGULAR uOS 1.42.d81***

For more information about SYSTEM DISK check section G. System Disk Content.

To create new windowed programs for in TRIANGULAR µOS use ***SDK TEMPLATE*** program as a starting point. To learn more about it, check section E. SDK Template.

D. TROUBLESHOOTING

When error is encountered in GUI and SDK TEMPLATE programs, Purple Screen of Death will show up, detailing type of error and line number containing it. There are two actions possible: F1 will try run program again, while F8 will shut down TRIANGULAR µOS and quit to BASIC.

Loading of a module of TRIANGULAR µOS can "freeze" in the process of inter-loading the next µOS module or disk program (a very rare occurrence). This happens when the loading screen does not proceed to the next module for over 1 minute. When the loading screen is not responsive for a longer time, it can mean an error in inter-loading procedure, most probably the keyboard buffer was not filled with keys properly. To see what really happened change the color of the cursor to blue (press Control + 7) and enter command COLOR 0,2 and hit Return key. This should change the background color to white which will show the underlying black text of the loading sequence message. If the computer doesn't change the cursor or background color and try again. If still there is no effect it might be a real freeze. Then restart the computer and start the µOS again.

If the color change procedure succeeds, try using the RUN command to see if the program will start or go to the top of the screen (Home key) and press Return in order to reload the program. If it will load successfully enter the RUN command again. If that does not work check if the load command is correct. It should have format: LOAD "[filename]", [device # (1 or 8 - 11)] like in e.g.: LOAD "GUI", 8. If none of it works then start the system anew. To prevent this kind of freeze, try not to use the keyboard when an inter-loading procedure is performed (it can slip an improper key into the keyboard buffer, which most often leads to this error).

E. SDK TEMPLATE

SDK TEMPLATE is a starter program to create new windowed apps for TRIANGULAR µOS. SDK TEMPLATE is a simple windowed program with system core embedded that demonstrates basic principles of functioning of µOS and is a great starting point for the development of new µOS apps. It addresses system calls through BASICALLY API JUMP TABLE and is written in a manner that allows easy edit on a real machine or emulator as well as external IDE like CBM prg Studio. (Caution: modification of system core still requires CBM prg Studio).

You can start creating your program either with help of ***SDK TEMPLATE.bas*** files (in Source code folder) or simply use ***SDK TEMPLATE*** program from ***SDK TEMPLATE.d81*** disk.

To run your newly created program, add it to any disk and insert this disk into the computer or attach it to the emulator, start µOS and use DISK or CMD program to navigate to and start it.

Also feel free to modify and experiment with the TRIANGULAR µOS system core.

SDK TEMPLATE program consist of TRIANGULAR µOS system core, BASICALLY API with Jump Table and template programs: ***SDK*** folder and ***TEMPLATE*** program.

After system initiation in program lines 0 to 2, at the end of line 2, instruction GOTO1000 gives control to ***SDK*** folder program. This and ***TEMPLATE*** program demonstrate how to write program using solely BASICALLY API Jump Table calls.

In line 1000 ***SDK*** program starts by setting default h[orizontal] and v[ertical] position of folder window with variable H3 and V3. This portion of code is performed only once in program instance.

```
1000 h3=7:v3=3
```

Line 1010 retrieves system settings from System Registry [*60101*] and updates system colors [*6100*]. Next it draws background & task bar [*61150*] and places icons on desktop [*61151*]. ***SDK TEMPLATE*** program add ***SDK*** icon on desktop that opens ***SDK*** folder.
Before drawing ***SDK*** folder window [*61200*], H and V position variables are passed as W1 and W2 to mark upper left corner of ***SDK*** window. WH and WV define length and height of windows. WN$ sets windows name as SDK, while WS set as 0 disables change page element. Finally this line starts subprogram in line 1090 that draws icon TEMPLATE in SDK window.

```
1010r=0:gosub60101:gosub61000:gosub61150:gosub61151:w1=h3:w2=v3:wh=11:wv=12:wn$="sdk":ws=0:gosub612
00:gosub1090
```

Line 1020 sets text area to whole screen [*61101*], creates mouse pointer on screen [*61100*] and gives control to mouse pointer routine [60001]. After joystick fire / mouse click this routine gets back with mouse position stored in H and V variables. Since those are values in pixels routine in [*60000*] converts them to character positions H0 and V0.

```
1020 gosub61101:gosub61100:gosub60001:gosub60000
```

Lines 1030 to 1070 check mouse position (by checking H0 and V0) and trigger action if user clicked on given element

```
1030 ifv0=w2andh0=w1+wh-2then61152
1035 ifv0=24andh0=39then61152
1040 ifv0>w2+2andh0>w1+2andv0<w2+8andh0<w1+8then1100
1050 ifv0=24andh0>18andh0<21thengosub61154
1055 ifv0=w2andh0>w1-1andh0<w1+whthengosub61160:goto1010
1060 ifv0<w2orh0<w1orv0>w2+wv-1orh0>w1+wh-1thenr=5:goto61153
1065 ifr>2thenr=1:goto1030
1070 ifr=1thenr=.:goto1010
```

Lines 1080 goes back to line 1020, which gives control to mouse pointer routine if no action was triggered.

```
1080 goto1020
```

Lines 1090 draws icon by setting I1, I2 – upper-left starting point of icon, IC$, an PETSCII graphic string of 5x5 icon and IN$, an icon name.

```
1090 i1=w1+3:i2=w2+3:ic$="{reverse on}{light green}O{cm y*3}P{cm g}tem{cm m}{cm g}pla{cm m}{cm g}te
{cm m}L{cm p*3}{sh @}":in$="template":gosub61202:return
```

Lines 1100 to 1180 do similar actions like in ***SDK*** folder, but to ***TEMPLATE*** program.

```
1100 h3=5:v3=2
1110r=.:gosub60101:gosub61000:gosub61150:gosub61151:w1=h3:w2=v3:wh=20:wv=14:wn$="template":ws=.:gos
ub61200:gosub1190
1120 gosub61101:gosub61100:gosub60001:gosub60000
1130 ifv0=w2andh0=w1+wh-2then1000
1135 ifv0=24andh0=39then61152
1140 ifv0=w2+6andh0>w1+4andh0<w1+16thenti$="000000"
1142 ifv0>w2+8andh0>w1+5andv0<w2+12andh0<w1+12then1000
1150 ifv0=24andh0>18andh0<21thengosub61154
1155 ifv0=w2andh0>w1-1andh0<w1+whthengosub61160:goto1110
1160 ifv0<w2orh0<w1orv0>w2+wv-1orh0>w1+wh-1thenr=5:goto61153
1165 ifr>2thenr=1:goto1130
1170 ifr=1thenr=.:goto1110

1180 goto1120
```

Lines 1190 and 1191 create 2 labels.

```
1190 l1=w1+3:l2=w2+3:lr=1:lc$="{cyan}":lt$="{126}:":gosub61201
1191 l1=w1+8:l2=w2+3:lr=1:lc$=w$:fl=pi:gosub60103:lt$=sg$:gosub61201
```

Lines 1192 and 1194 create 2 button.

```
1192 b1=w1+4:b2=w2+5:bt=.:bt$="reset clock":gosub61203
1194 b1=w1+6:b2=w2+9:bt=1:bt$="exit":gosub61203:return
```

F. BASICALLY API

Below are listed functions of BASICALLY API of TRANGULAR µOS 1.42.

Basic functions of BASICALLY API for TRIANGULAR µOS 1.42

1. LEFT$(S$,*X*) – will display *X* number of spaces (max *X* = 40)
2. LEFT$(V$,*X*) – will move cursor down *X* number of times (max *X* = 24)
3. LEFT$(H$,*X*) – will move cursor right *X* number of times (max *X* = 40)

BASICALLY API JUMP TABLE for TRIANGULAR µOS 1.42

System calls of BASICALLY API JUMP TABLE are meant to be permanent and will work on later µOS versions.

Rudimentary functions

Convert mouse sprite position to window loop objects position:
GOSUB 60000

Mouse pointer steering:
GOSUB 60001

Clear SID registers:
GOSUB 60002

Beep sound:
GOSUB 60003

Mouse pointer steering SYSTEM tab:
GOSUB 60004

Mouse pointer steering MATH:
GOSUB 60005

Time TI$ to hh:mm:ss parts and seconds mark:
GOSUB 60006

Parts of hh:mm:ss to time TI$:
GOSUB 60007

Save settings to UOS.CFG file & store variables in system memory:
GOSUB 60100

Retrieve memory variables:
GOSUB 60101

Load program:
GOSUB 60102

Change float [fl] to neatly trimmed string [sg$]:
GOSUB 60103

GUI drawing functions

Update color variables:
GOSUB 61000

1st time VIC-II initialize & sprites off & sprites space clear & sprites shapes:
GOSUB 61001

VIC-II initialize & sprites off & sprites space clear & sprites shapes:
GOSUB 61002

Turn off, reset and move all sprites to bottom-right corner:
GOSUB 61003

Move all sprites to bottom-right corner:
GOSUB 61004

Create mouse pointer:
GOSUB 61100

Full size window area:
GOSUB 61101

Draw load:
GOSUB 61102

Load sequence:
GOSUB 61103

Restart system:
GOSUB 61104

Shut down:
GOSUB 61105

Go back to µOS:
GOSUB 61106

Draw Background & Task bar:
GOSUB 61150

Draw DESKTOP icons:
GOSUB 61151

Go to DESKTOP:
GOSUB 61152

Go to DESKTOP loop:
GOSUB 61153

Go to Start Menu:
GOSUB 61154

Move window:
GOSUB 61160

Window generator:
GOSUB 61200

> ***Window generator**: draws an empty window based on data in variable arguments. Before evoking this function assign desired values to those variables:*
> ***W1** - window top-left horizontal position*
> ***W2** - window top-left vertical position*
> ***W3** - length horizontal position*
> ***W4** - height bottom-right vertical position*
> ***WN$** - window name which will be displayed on title and task bar*
> ***WS** - window slider (0 – disable / 1 - enable)*
>
> *Next evoke this function with BASICALLY API system call **GOSUB 61200***
>
> *Caution: variable J is used in FOR=TO:NEXT loops.*

Text label generator:
GOSUB 61201

> ***Text label generator**: Places text label based on data in variable arguments. Before evoking this function assign desired values to those variables:*
> ***L1** - text label top-left horizontal position*
> ***L2** - text label top-left vertical position*
> ***LR** – text reversed [0 - no / 1 - reversed]*
> ***LC$** - text label color [string variable]*
> ***LT$** - label text*
>
> *Next evoke this function with BASICALLY API system call **GOSUB 61201***

Icon generator:
GOSUB 61202

> ***Icon generator****: Places 5x5 icon based on data in variable arguments. Before evoking this function assign desired values to those variables:*
> ***I1*** *- icon top-left horizontal position*
> ***I2*** *– icon top-left vertical position*
> ***IC$*** *- icon 5x5 graphic [PETSCII string variable]*
> ***IN$*** *- icon name text*
>
> *Next evoke this function with BASICALLY API system call* ***GOSUB 61202***

Button generator:
GOSUB 61203

> ***Button generator****: Places button based on data in variable arguments. Before evoking this function assign desired values to those variables:*
> ***B1*** *- button top-left horizontal position*
> ***B2*** *– button top-left vertical position*
> ***BT*** *– buttor type [0 - slim / 1 - bulky]*
> ***BT$*** *- button text*
>
> *Next evoke this function with BASICALLY API system call* ***GOSUB 61203***
>
> *Caution: variable J is used in FOR=TO:NEXT loops.*

G. System Disk Content

#	Name	Type	Size Bytes	Size KB	Disk Blocks	Size on Disk KB
1	UOS	PRG	6,188	6.04	25	6.25
2	UOS.CFG	SEQ	66	0.06	1	0.25
3	GUI	PRG	24,796	24.21	98	24.50
4	WORDS	PRG	3,450	3.37	14	3.50
5	SYNTH	PRG	2,737	2.67	11	2.75
6	SIMCITY	PRG	4,639	4.53	19	4.75
7	SIMCITY.SAV	PRG	54	0.05	1	0.25
8	SIMCITY.SCR	PRG	840	0.82	4	1.00
9	SIMCITY.SCL	PRG	840	0.82	4	1.00
10	STAR WARS	PRG	6,113	5.97	25	6.25
11	CRAB IN NEW YORK	PRG	3,365	3.29	14	3.50
12	GP BRAZIL	PRG	3,275	3.20	13	3.25
13	GUI.SPR	BIN	512	0.50	3	0.75
14	WORDS.SPR	BIN	128	0.13	1	0.25
15	STAR WARS.SPR	BIN	512	0.50	3	0.75
16	CRAB IN NYC.SPR	BIN	512	0.50	3	0.75
17	GP BRAZIL.SPR	BIN	64	0.06	1	0.25
18	CLIPBOARD.DAT	SEQ	2	0.00	1	0.25
		TOTAL:	**58,093**	**56.73**	**241**	**60.25**

H. SYSTEM REGISTRY

Adress [DEC/$]	System Registry position	Values [DEC]	Function
3071 / $0BFF	Commodore computer line	0	Unknown
		1	PET 1.0
		2	PET 2.0-4.1
		3	VIC-20
		4	C64
		5	C128: C64 Mode
		6	C128
		7	Plus/4
		8	CBM-II P/500
		9	CBM-II B/600/700
		10	C65
		11	MEGA65
3070 / $0BFE	Screen width [SW]	Default: 40	Screen width
3069 / $0BFD	Memory size [FM]	0-255	Size in KB
3068 / $0BFC	Desktop background pattern [BP]	<>0	0: Default [223]
3067 / $0BFB	Desktop background reverse [BR]	0	Not reversed
		1	Reversed
3066 / $0BFA	Title bar color [TC]	0-255	4-bit
3065 / $0BF9	Desktop pattern color [BC]	0-255	4-bit
3064 / $0BF8	Mouse pointer horizontal position [H0]	0-40	Default: [20]
3063 / $0BF7	Mouse pointer vertical position [V0]	0-24	Default: [10]
3062 / $0BF6	GUI Program mode	0	None
		1	DESKTOP
		2	THIS PC
		3	SETTINGS: SYSTEM
		4	APPS
		5	GAMES
		6	SETTINGS: WINDOW
		7	DISK
		8	MATH
		9	CMD
		10	MONITOR
		11	SETTINGS: DESKTOP
3061 / $0BF5	Printer device # [PP]	3-7	Default: 3 [None]
3060 / $0BF4	Datasette availability [DD]	0	No
		1	Yes - #1 [Default]

3051-3057 / $0BEB-$0BF0	Sequence of values in keyboard buffer for load module	-	-
3050 / $0BEA	TRIANGULAR µOS version	0-255	142 for version 1.42
3049 / $0BE9	Window color [W]	0-255	4-bit
3048 / $0BE8	Button color [B]	0-255	4-bit
3047 / $0BE7	Text color [G]	0-255	4-bit
3046 / $0BE6	Task bar color [A]	0-255	4-bit
3045 / $0BE5	Start Menu color [SM]	0-255	4-bit
3000 / $0BB8	Boot drive # [BD]	8-11	Device #
3001 / $0BB9	Work drive # [WD]	8-11	Device #
3011 / $0BC3	Device #8 detected	0 1	No Yes
3012 / $0BC4	Device #9 detected	0 1	No Yes
3013 / $0BC5	Device #10 detected	0 1	No Yes
3014 / $0BC6	Device #11 detected	0 1	No Yes
3015 / $0BC7	Device #8 code	0-255	Ref: Drive codes
3016 / $0BC8	Device #9 code	0-255	Ref: Drive codes
3017 / $0BC9	Device #10 code	0-255	Ref: Drive codes
3018 / $0BCA	Device #11 code	0-255	Ref: Drive codes

Keyboard buffer table

Keyboard	Buffer	Buffer size
Commodore 128	842-851	208 /$00D0

Tape buffer:

Commodore 128:	2816-3071 / $0B00-$0BFF

Drives codes

#	Drive name	Also used by
0	Unknown	
7	2031	
16	2040	
32	3040	
169	4040	
170	1541	1540
76	1541-II	8050, 8250, SFD-1001 & D9060/D9090
255	1551	
173	1571	1570
108	1581	
48	SD2IEC	* Experimental

I. BASIC VARIABLES

#	Variable	Type	Description	Value	Memory cell	Notes
1	**A**	Float	Task bar color ASCII code	0-255	3046 / $0BE6	
2	**A$**	String	Task bar color character	Any	-	
3	**A1**	Float	1st memory address	Any	-	Used only in MONITOR
4	**A2**	Float	2nd memory address	Any	-	Used only in MONITOR
5	**AC**	Float	GUI Program mode	0-8	3062 / $0BF6	
6	**AD**	Float	2-byte address variable	0-65535	-	Used only in MONITOR
7	**B**	Float	Button color ASCII code	0-255	3048 / $0BE8	
8	**B$**	String	Button color character	Any	-	
9	**B1**	Float	Button generator top-left [H0] position	0-255	-	
10	**B2**	Float	Button generator top-left [V0] position	0-255	-	
11	**BC**	Float	Desktop pattern color ASCII code	0-255	3065 / $0BF9	
12	**BC$**	String	Desktop pattern color character	Any	-	
13	**BD**	Float	Boot drive #	8-11	3000 / $0BB8	
14	**BP**	Float	Desktop background pattern ASCII code	<>0 (0 = Default [209])	3068 / $0BFC	
15	**BP$**	String	Desktop background pattern character	Any	-	
16	**BQ$**	String	2 char format disk ID	Any	-	Used only in CMD
17	**BR**	Float	Desktop background reverse OFF / ON	0 / 1	3067 / $0BFB	
18	**BR$**	String	Desktop background reverse character	{REVERSE OFF / ON}	-	
19	**BS**	Float	Temporary boot drive #	8-11	*3000 / $0BB8*	Used only in UOS
20	**BT**	Float	Button generator type	0 / 1	-	[0] Slim / [1] Bulky
21	**BT$**	String	Button generator text	Any	-	
22	**BW$**	String	Generated Task Bar string	Any	-	
23	**C$**	String	Command string	Any	-	Used only in CMD & MONITOR
24	**C1$**	String	First temporary string variable	Any	-	
25	**C2$**	String	Second temporary string variable	Any	-	
26	**CE**	Float	Error flag variable	Any	-	Used only in CMD
27	**CL**	Float	Command length	Any	-	Used only in CMD & MONITOR
28	**CS$**	String	Datasette availability display string	Any	-	Used only in CMD
29	**D1**	Float	Disk drive # change variable	Any	-	Used only in CMD
30	**D9**	Float	Database disk drive code	Any	-	Used only in CMD
31	**D9$**	String	Database disk drive string	Any	-	Used only in CMD
32	**DC**	Float	1 byte variable	0-255	-	Used only in MONITOR
33	**DD**	Float	Datasette availability	0-1	3060 / $0BF4	
34	**DI**	Float	Disk drive availability	Any	-	Used only in CMD
35	**DN**	Float	Disk drive code	Any	-	Used only in CMD
36	**DR**	Float	Active drive #	1 / 8-11	-	
37	**DR$**	String	Active drive # [DR] string	"1" / "8" - "11"	-	

38	**DT$**	String	Disk drive name	Any	-	Used only in CMD
39	**EN$**	String	Disk drive channel error name	Any	-	
40	**ER$**	String	Disk drive channel error code #	Any	-	
41	**ES$**	String	Disk drive channel error sector	Any	-	
42	**ET$**	String	Disk drive channel error track	Any	-	
43	**F$**	String	Char fetched from disk name	Any	-	
44	**FH$**	String	Disk header name	Any	-	
45	**FI**	Float	Number of disk entries	Any	-	
46	**FI$(X)**	String array	String array of disk content or text lines	Any	-	
47	**FL**	Float	Float variable to be trimmed to [SG$]	Any	-	Used only in DISK
48	**FM**	Float	BASIC System memory in KB	0-255	3069 / $0BFD	
49	**FY$**	String	Disk header ID	Any	-	Used only in DISK
50	**G**	Float	Text label color ASCII code	0-255	3047 / $0BE7	
51	**G$**	String	Text label color character	Any	-	
52	**G0**	Float	Start of VIC-II registers	Constant	53248 / $D000	
53	**G1**	Float	Start of screen memory	Any	-	PEEK(2616)*256
54	**GC**	Float	Start of color memory	Any	-	((PEEK(648)AND3)+148)*256
55	**H**	Float	Mouse pointer sprite horizontal position	Any	3064 / $0BF8	
56	**H$**	String	String cursor right	{right*39} - Constant	-	
57	**H0**	Float	Mouse sprite loop position horizontal	0-40	-	
58	**H3**	Float	Window default [H0] position	0-255	-	
59	**H4**	Float	Backup H0 before moving window	0-255	-	
60	**H7**	Float	Mouse movement [H] max. limit	Any	-	[Default: 338]
61	**H8**	Float	Mouse movement [H] min. limit	Any	-	[Default: 26]
62	**HL**	Float	Desired HEX number length	Any	-	Used only in MONITOR
63	**HW$**	String	HEX number string	Any	-	Used only in MONITOR
64	**I**	Float	Temporary variable (used e.g., in FOR)	Any	-	
65	**I1**	Float	Icon generator top-left [H0] position	0-255	-	
66	**I2**	Float	Icon generator top-left [V0] position	0-255	-	
67	**IC$**	String	Icon generator icon PETSCII graphic	Any	-	5 x 5 PETSCII chars + special codes
68	**IN$**	String	Icon generator name text	Any	-	
69	**IP$**	String	Command prompt	Any	-	Used only in MONITOR
70	**J**	Float	Temporary variable (used e.g., in FOR)	Any	-	* or read Joystick with JOY command
71	**K**	Float	Temporary variable (used e.g., in FOR)	Any	-	
72	**K$**	String	Default key char variable	Any	-	
73	**L**	Float	Length of string	Any	-	
74	**L1**	Float	Label generator top-left [H0] position	0-255	-	
75	**L2**	Float	Label generator top-left [V0] position	0-255	-	
76	**L$(X)**	String array	Loading text and CBM DOS commands	Any	-	Used with subscript 1 or 2 only
77	**LC$**	String array	Label generator text color	Any	-	PETSCII color char
78	**LR**	Float	Label generator reversed mode	0-255	-	[0] - Reverse Off / [1] - Reverse On
79	**LN$**	String	Register number display padding	Any	-	Used only in MATH

80	**LT$**	String array	Label generator text	Any	-	
81	**MO**	Float	Mouse pointer movement increment	Any	-	[Default: 8]
82	**MV**	Float	Temporary variable for memory cell	Any	-	Used only in MONITOR
83	**N**	Float	Temporary variable with disk code	Any	-	
84	**N$**	String	String variable for file name or temporary	Any	-	
85	**NA**	Float	A register	Any	-	Used only in MATH
86	**NA$**	String	A register string	Any	-	Used only in MATH
87	**NB**	Float	B register	Any	-	Used only in MATH
88	**NB$**	String	B register string	Any	-	Used only in MATH
89	**NC**	Float	Temporary variable	Any	-	
90	**ND**	Float	Dot flag	Any	-	Used only in MATH
91	**NL**	Float	Length of trimmed [C$] string	Any	-	Used only in MONITOR
92	**NM**	Float	Variable of MATH clipboard	Any	-	
93	**NT$**	String	MATH register temporary string	Any	-	Used only in MATH
94	**PD**	Float	Disk listing page	Any	-	Used only in DISK
95	**PD$**	String	String of max. disk listing page [PD]	Any	-	Used only in DISK
96	**PM**	Float	Max. disk listing pages #	Any	-	Used only in DISK
97	**PM$**	String	String of max. disk listing page [PM]	Any	-	Used only in DISK
98	**PP**	Float	Printer device #	3-7	3061 / $0BF5	Default: 3 [None
99	**PS$**	String	String clearing disk header	Constant	-	Used only in DISK
100	**Q$**	String	Quotation mark	CHR$(34) ["]	-	
101	**R**	Float	Return from Start Menu	0-2	-	
102	**S**	Float	Start of SID sound memory	Constant	54272 / $D400	
103	**S$**	String	String space	{space*39} - Constant	-	
104	**SG$**	String	Trimmed string from [FL]	Any	-	Used only in DISK
105	**SM**	Float	Start Menu color ASCII code	0-255	3045 / $0BE5	
106	**SM$**	String	Start Menu color character	Any	-	
107	**SL**	Float	Length of [SG$] string	Any	-	Used only in DISK
108	**T**	Float	Temporary variable	Any	-	
109	**T$**	String	Time in TI$ form	"000000" - "235959"	-	
110	**T0$**	String	Temporary string for time changing	Any	-	
111	**T1$**	String	Hours	"0" - "23"	-	
112	**T2$**	String	Minutes	"0" - "59"	-	
113	**T3$**	String	Seconds	"0" - "59"	-	
114	**TC**	Float	Title bar color ASCII code	0-255	3066 / $0BFA	
115	**TC$**	String	Title bar color character	Any	-	
116	**TD**	Float	Processed [TN]	Any	-	Used only in MONITOR
117	**TD$**	String	1 HEX digit trimmed from string [IN$]	Any	-	Used only in MONITOR
118	**TN**	Float	Variable of [TD$]	Any	-	Used only in MONITOR
119	**TS$**	String	Seconds blinking	" " / ":"	-	
120	**V**	Float	Mouse pointer sprite vertical position	Any	3063 / $0BF7	
121	**V$**	String	String cursor down	{down*23} - Constant	-	

122	**V0**	Float	Mouse sprite loop position vertical	0-24	-	
123	**V3**	Float	Window default [V0] position	0-255	-	
124	**V4**	Float	Backup [V0] before moving window	0-255	-	
125	**V7**	Float	Mouse movement [V] max. limit	Any	-	[Default: 244]
126	**V8**	Float	Mouse movement [V] min. limit	Any	-	[Default: 52]
127	**VR**	Float	Version number in System Registry	Any	3050 / $0BEA	142 / 100 for version 1.42
128	**VT**	Float	Code version number	Any	-	1.42 for version 1.42
129	**W**	Float	Window color ASCII code	0-255	3049 / $0BE9	
130	**W$**	String	Window color character	Any	-	
131	**W1**	Float	Window generator: top-left [H0] position	Any	-	
132	**W2**	Float	Window generator: top-left [V0] position	Any	-	
133	**WD**	Float	Work drive #	8-11	3001 / $0BB9	
134	**WH**	Float	Window generator: window width	Any	-	
135	**WN$**	String	Window generator window name	Any	-	
136	**WS**	Float	Window generator: window slider	0 / 1	-	[0] Disable / [1] Enable
137	**WV**	Float	Window generator: window height	Any	-	

J. CONFIG FILE [UOS.CFG]

#	Position name	Values [DEC]	Default value [DEC]	Variable	Adress [DEC/$]
1	TRIANGULAR µOS version	142	142	VR	3050 / $0BEA
2	Work drive #	8 - 11	8	WD	3001 / $0BB9
3	Datasette availability	0 / 1	1	DD	3060 / $0BF4
4	Desktop background reverse	0 / 1	0	BR	3067 / $0BFB
5	Desktop background pattern	Any	209	BP	3068 / $0BFC
6	Desktop pattern color	Any	30	BC	3065 / $0BF9
7	Title bar color	Any	129	TC	3066 / $0BFA
8	Printer device #	3-7	3	PP	3061 / $0BF5
9	Window color	Any	5	W	3049 / $0BE9
10	Button color	Any	152	B	3048 / $0BE8
11	Text color	Any	129	G	3047 / $0BE7
12	Task bar color	Any	129	A	3046 / $0BE6
13	Start Menu color	Any	5	SM	3045 / $0BE5

K. SUPPORT & LEGAL NOTE

More information about TRIANGULAR µOS for Commodore 128 and other computer systems is available on TRIANGULAR µOS website, where you can download SDK, report bug or get help: triangular-uos.blogspot.com

LEGAL NOTE:

L. CHANGELOG

TRIANGULAR µOS 1.42 for Commodore 128 [12-01-2024]:

- GUI: Purple Screen of Death is introduced
- SETTINGS is now divided into 3 tabs: SYSTEM, WINDOW and DESKTOP
- SETTINGS WINDOW tab can customize colors of GUI elements
- SETTINGS DESKTOP tab can customize desktop wallpaper
- MATH: Overflow error now just restarts MATH app
- Resource files with sprite data changed extension from .RES to .SPR
- WORDS: blinking sprite cursor & other improvements
- SIMCITY: new look, HUD and title screen. Zones now have 2 tiers. Zone plot can be cleared. Game can be saved and loaded
- GP BRAZIL improved and race track is based on Interlagos Circuit
- Improvements and bugfixes

TRIANGULAR µOS 1.41 for Commodore 128 [31-12-2023]:

- Wrongly assigned function keys are fixed
- System clipboard is stored on disk in CLIPBOARD.DAT file
- BIOS: Expanded error messages system
- GUI: New expanded START MENU
- GUI: Task bar changes color with Title bar
- GUI: Some icons updated
- GUI: Move windows procedure improved
- DISK: Misplaced labels when moving fixed
- BASICALLY API expanded with icons, buttons and text labels elements
- WORDS is 3x times faster type-in mechanism and other improvements
- GP BRAZIL: More realistic opponent cars placement and more random race track
- Variables simplified and reorganized
- Code restructured
- BASICALLY Jump Table expanded
- SDK TEMPLATE program is expanded
- Improvements & bugfixes
- Documentation updated and expanded

TRIANGULAR µOS 1.40 for Commodore 128 [17-12-2023]:

- New theme called "Brazil"
- BIOS: Initializing procedure reworked
- GUI: Streamlined, code cleaned and improved
- MONITOR: Memory display mechanism is reworked and 7 times faster
- WORDS: branch out to separate file & improvements
- SYNTH: branch out to separate file & improvements
- STAR WARS: branch out to separate file & improvements
- CRAB IN NEW YORK: branch out to separate file & improvements
- SIMCITY: branch out to separate file & improvements
- GP BRAZIL added as separate game in place of TRASURE CHAMBER
- Sprites data moved into separate asset files
- BASICALLY API Jump Table removed from GUI
- Clipboard moved to System Registry
- Improvements & bugfixes
- Updated documentation

TRIANGULAR µOS 1.36 for Commodore 128 [04-12-2023]:

- SYSTEM DISK folder added for use with SD2IEC
- GUI: Fixed bug preventing moving windows up and left
- SETTINGS: Fixed labels dislocation when window is moved
- When windows are moved time is now updating
- MATH: Fixed bug in memory operations
- MATH: Fixed bug in reading clipboard values mechanism
- WORDS saves its text files in .txt format instead of .doc
- GAMES folder rearranged: STAR WARS beside SIMCITY, swapped positions with TREASURE
- STAR WARS: Sound effects added to its game engine
- Corrections of minor visual discrepancies
- Improvements and bugfixes
- Updated documentation

TRIANGULAR µOS 1.35 for Commodore 128 [23-06-2023]:

- BIOS improved and bug fixed
- Movable windows by clicking on title bar
- Mouse pointer change when in moving window mode
- Mouse pointer change to hourglass when waiting
- Desktop icons layout rearrangement
- Start menu orb from sprite
- MATH bug fixed
- GUI cleaned up and improved
- Sprites without flickering
- Updated loader graphic using sprite stripes
- Border COLOR reinforced
- Code refactored
- CMD and MONITOR text area handled by WINDOW command
- CMD info properly display amount of free memory from both memory banks
- MONITOR function calling machine program is much simpler
- Drive detection database is held in DATA statements
- CRAB IN NEW YORK error in which 1 collision with cars or trains takes 2 lives is fixed plus minor improvements
- STAR WARS: X-Wing vs TIE Fighter game added
- DO...WHILE...LOOP...UNTIT...EXIT and BEIN...BEND commands implemented
- FAST and SLOW utilized to speed up drawing PETSCII elements of inter-loading, also in BIOS and GAMES
- BASICALLY API Window creation function expanded
- BASICALLY API Jump Table added
- Improvements and bugfixes
- SDK adds TEMPLATE windowed program with its source files
- Updated and augmented documentation and SDK documentation now in single PDF file

TRIANGULAR µOS 1.34 for Commodore 128 [28-05-2023]:

- Windows loops changes to relative windows position
- Mouse routine improved and mouse position changed
- More BASIC 7.0 commands added
- SIMCITY bug fixed
- CRAB IN NEW YORK, a 3rd game added
- Bugfixes

TRIANGULAR µOS 1.33 for Commodore 128 [17-05-2023]:

- Mouse routine reworked and improved
- BASICALLY API Window creator reworked with window displaying mechanism
- TREASURE CHAMBER, game by Fabrizo Caruso added
- More BASIC 7.0 commands added (IF…THEN…ELSE, SLEEP) and RESTORE command expanded
- Bugfixes

TRIANGULAR µOS 1.32 for Commodore 128 [11-02-2023]:

- BASIC 7.0 WINDOW command utilized
- Enlarged DESKTOP area
- Change versioning scheme and version held as numeric value in memory
- Change file system from filename>ext (-ension) to filename.ext (-ension) e.g.: uos>cfg to uos.cfg
- BASIC 7.0 sound commands implemented
- Minor improvements and bugfixes

TRIANGULAR µOS 1.31/C128 for Commodore 128 [15-01-2023]:

- Mouse pointer routines redesigned which resulted in twice faster movements
- Some additional graphic operations converted to BASIC 7.0 syntax
- Minor improvements and bugfixes

TRIANGULAR µOS 1.30/C128 for Commodore 128 [12-01-2023]:

- Commodore 128 in its standard 40 column VIC-II C128 mode is supported
- Only 3.5” 1581 type disk drive is supported
- Loading and saving is 10x faster due to faster C128 1581 disk drive handling
- Color theme changed from C64 blue mesh to C128 pyramids
- BASIC 7.0 handles sprites and few needed instructions added
- GAMES folder contains only SIMCITY, other 3 games removed
- Minor improvements and bugfixes