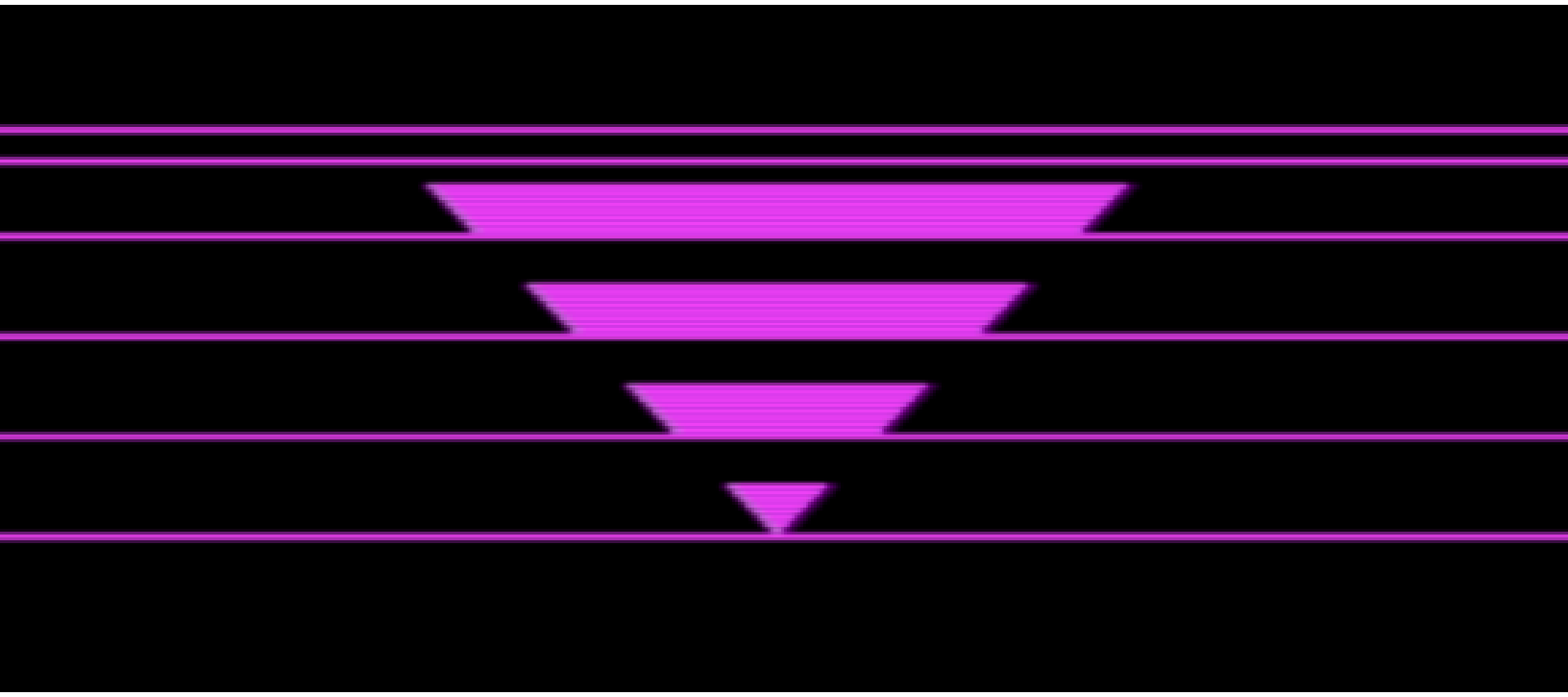

TRIANGULAR µOS 1.36 SDK

for

COMMODORE 128

PERSONAL COMPUTER

Programmers Reference Guide

Contents:

A. Introduction

Programmer's Reference Guide for TRIANGULAR µOS 1.36 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 1.36) 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 1.36 for Commodore 128 Programmer's Reference Guide.pdf* – this document
- *Source Code* folder with 4 files: *UOS.bas* (source code of UOS program), *GUI.bas* (source code of GUI program), *uos.cfg.seq* (default configuration file) and *TEMPLATE.bas* (source code of TEMPLATE program)
- *TRIANGULAR uOS 1.36.d81* – empty, preformatted System Disk
- *TRIANGULAR uOS 1.36 Documents.d81* – empty, preformatted Documents Disk
- *TRIANGULAR uOS 1.36 TEMPLATE.d81* – disk with TEMPLATE program helping in starting creating µOS programs

B. What you need

In order to change and/or compile TRIANGULAR µOS 1.36 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 1.36). 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 1.36

Source code of UOS and GUI programs is stored in the Source code folder in ***UOS.bas*** and ***GUI.bas*** 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 those files in CBM prg Studio, add compiled programs ***UOS.prg*** and ***GUI.prg*** to System Disk. File names on disk should always be UOS and GUI (in upper case/graphic mode) or uos and gui (in lower case mode). 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).

You can use empty, preformatted System Disk file to speed up this process: ***TRIANGULAR uOS 1.36.d81***

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

To create new windowed programs to use in TRIANGULAR µOS use ***TEMPLATE.bas*** (in Source code folder) as a starting point and compile it in CBM prg Studio or simply use TEMPLATE program from ***TEMPLATE.d81*** disk.

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 (where modification of system core still requires CBM prg Studio).

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

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

D. Troubleshooting

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. BASICALLY

Below are listed functions of BASICALLY API of TRANGULAR µOS 1.36.

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

Basic functions of BASICALLY API for TRIANGULAR µOS 1.36

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 1.36

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

Save settings to uos.cfg & store variables in memory system:
GOSUB 60100

Retrieve memory variables:
GOSUB 60101

Load program:
GOSUB 60102

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 uOS:
GOSUB 61106

Draw Background & task panel:
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.*

F. System Disk Content

#	Name	Type	Size B	Size KB	Disk Blocks	Size on Disk KB
1	**UOS**	PRG	4,657	4.55	19	4.75
2	**UOS.CFG**	SEQ	39	0.04	1	0.25
3	**GUI**	PRG	41,590	40.62	164	41.00
		TOTAL:	**46,286**	**45.20**	**184**	**46.00**

G. 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
		4	APPS
		5	GAMES
		6	COLORS
		7	DISK
		8	MATH
		9	CMD
		10	External program
3061 / $0BF5	Printer device # [PP]	0 or 4-7	Default: 0 [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	136 for version 1.36
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

H. BASIC variables

#	Variable	Type	Description	Value	Memory cell	Notes
1	**A1**	Float	1st memory address	Any	-	Used only in MONITOR
2	**A2**	Float	2nd memory address	Any	-	Used only in MONITOR
3	**AC**	Float	GUI Program mode	0-8	3062 / $0BF6	
4	**AD**	Float	2-byte address variable	0-65535	-	Used only in MONITOR
5	**BC**	Float	Desktop pattern color ASCII code	0-255	3065 / $0BF9	
6	**BC$**	String	Desktop pattern color character	Any	-	
7	**BD**	Float	Boot drive #	8-11	3000 / $0BB8	
8	**BP**	Float	Desktop background pattern ASCII code	<>0 (0 = Default [223])	3068 / $0BFC	
9	**BP$**	String	Desktop background pattern character	Any	-	
10	**BQ$**	String	2 char format disk ID	Any	-	Used only in CMD
11	**BR**	Float	Desktop background reverse OFF / ON	0 / 1	3067 / $0BFB	
12	**BR$**	String	Desktop background reverse character	{REVERSE OFF / ON}	-	
13	**BS**	Float	Temporary boot drive #	8-11	*3000 / $0BB8*	Used only in UOS
14	**BW$**	String	Generated Task Bar string	Any	-	
15	**C$**	String	Command string	Any	-	Used only in CMD
16	**C1$**	String	First temporary string variable	Any	-	
17	**C2$**	String	Second temporary string variable	Any	-	
18	**C7**	Float	Length of [IN$] string	Any	-	Used only in MONITOR
19	**CE**	Float	Error flag variable	Any	-	Used only in CMD
20	**CL**	Float	Command length	Any	-	Used only in CMD & MONITOR
21	**CS$**	String	Datasette availability display string	Any	-	Used only in CMD
22	**D1**	Float	Disk drive # change variable	Any	-	Used only in CMD
23	**D9**	Float	Database disk drive code	Any	-	Used only in CMD
24	**D9$**	String	Database disk drive string	Any	-	Used only in CMD
25	**DC**	Float	1 byte variable	0-255	-	Used only in MONITOR
26	**DD**	Float	Datasette availability	0-1	3060 / $0BF4	
27	**DI**	Float	Disk drive availability	Any	-	Used only in CMD
28	**DN**	Float	Disk drive code	Any	-	Used only in CMD
29	**DR**	Float	Active drive #	1 / 8-11	-	
30	**DR$**	String	Active drive # [DR] string	"1" / "8" - "11"	-	
31	**DT$**	String	Disk drive name	Any	-	Used only in CMD
32	**EN$**	String	Disk drive channel error name	Any	-	
33	**ER$**	String	Disk drive channel error code #	Any	-	
34	**ES$**	String	Disk drive channel error sector	Any	-	
35	**ET$**	String	Disk drive channel error track	Any	-	
36	**F$**	String	Char fetched from disk name	Any	-	
37	**FH$**	String	Disk header name	Any	-	
38	**FI**	Float	Number of disk entries	Any	-	

39	FI$(X)	String array	String array of disk content or text lines	Any	-	
40	FL	Float	Float variable to be trimmed to [SG$]	Any	-	Used only in DISK
41	FM	Float	BASIC System memory in KB	0-255	3069 / $0BFD	
42	FY$	String	Disk header ID	Any	-	Used only in DISK
43	G0	Float	Start of VIC-II registers	Constant	53248 / $D000	
44	G1	Float	Start of screen memory	Any	-	PEEK(2616)*256
45	GC	Float	Start of color memory	Any	-	((PEEK(648)AND3)+148)*256
46	H	Float	Mouse pointer sprite horizontal position	Any	3064 / $0BF8	
47	H$	String	String cursor right	{right*39} - Constant	-	
48	H0	Float	Mouse sprite loop position horizontal	0-40	-	
49	H3	Float	Window default [H0] position	0-255	-	
50	H4	Float	Backup H0 before moving window	0-255	-	
51	H7	Float	Mouse movement [H] max. limit	Any	-	[Default: 338]
52	H8	Float	Mouse movement [H] min. limit	Any	-	[Default: 26]
53	HL	Float	Desired HEX number length	Any	-	Used only in MONITOR
54	HW$	String	HEX number string	Any	-	Used only in MONITOR
55	I	Float	Temporary variable (used e.g., in FOR)	Any	-	
56	IN$	String	Command string	Any	-	Used only in MONITOR
57	IP$	String	Command prompt	Any	-	Used only in MONITOR
58	J	Float	Temporary variable (used e.g., in FOR)	Any	-	
59	J0	Float	Read Joystick register with JOY command	0-255	-	
60	K	Float	Temporary variable (used e.g., in FOR)	Any	-	
61	K$	String	Default key char variable	Any	-	
62	L	Float	Cut string by L value (RIGHT$ or LET$)	Any	-	Used only in CMD & MONITOR
63	L$(X)	String array	Loading text and CBM DOS commands	Any	-	Used with subscript 1 or 2 only
64	LN	Float	Length of string	Any	-	Used only in MATH
65	LN$	String	Register number display padding	Any	-	Used only in MATH
66	MO	Float	Mouse pointer movement increment	Any	-	[Default: 8]
67	MV	Float	Temporary variable for memory cell	Any	-	Used only in MONITOR
68	N	Float	Temporary variable with disk code	Any	-	
69	N$	String	String variable for file name or temporary	Any	-	
70	NA	Float	A register	Any	-	Used only in MATH
71	NA$	String	A register string	Any	-	Used only in MATH
72	NB	Float	B register	Any	-	Used only in MATH
73	NB$	String	B register string	Any	-	Used only in MATH
74	NC	Float	Temporary variable for registers	Any	-	Used only in MATH
75	ND	Float	Dot flag	Any	-	Used only in MATH
76	NL	Float	Length of trimmed [IN$] string	Any	-	Used only in MONITOR
77	NM	Float	Variable of MATH clipboard	Any	-	
78	NT$	String	MATH register temporary string	Any	-	Used only in MATH
79	NZ	Float	Temporary variable for registers	Any	-	Used only in MATH
80	OO$	String	Color of SETTING label	Any	-	

81	**OT**	Float	Temporary variable	Any	-	Used only in MONITOR
82	**PD**	Float	Disk listing page	Any	-	Used only in DISK
83	**PD$**	String	String of max. disk listing page [PD]	Any	-	Used only in DISK
84	**PM**	Float	Max. disk listing pages #	Any	-	Used only in DISK
85	**PM$**	String	String of max. disk listing page [PM]	Any	-	Used only in DISK
86	**PP**	Float	Printer device #	0 or 4-7	3061 / $0BF5	
87	**PS$**	String	String clearing disk header	Constant	-	Used only in DISK
88	**Q$**	String	Quotation mark	CHR$(34) ["]	-	
89	**R**	Float	Return from Start Menu	0-2	-	
90	**S**	Float	Start of SID sound memory	Constant	54272 / $D400	
91	**S$**	String	String space	{space*39} - Constant	-	
92	**SG$**	String	Trimmed string from [FL]	Any	-	Used only in DISK
93	**SL**	Float	Length of [SG$] string	Any	-	Used only in DISK
94	**T**	Float	Temporary variable	Any	-	
95	**T$**	String	Time in TI$ form	"000000" - "235959"	-	
96	**T0$**	String	Temporary string for time changing	Any	-	
97	**T1$**	String	Hours	"0" - "23"	-	
98	**T2$**	String	Minutes	"0" - "59"	-	
99	**T3$**	String	Seconds	"0" - "59"	-	
100	**TC**	Float	Title bar color ASCII code	0-255	3066 / $0BFA	
101	**TC$**	String	Title bar color character	Any	-	
102	**TD**	Float	Processed [TN]	Any	-	Used only in MONITOR
103	**TD$**	String	1 HEX digit trimmed from string [IN$]	Any	-	Used only in MONITOR
104	**TN**	Float	Variable of [TD$]	Any	-	Used only in MONITOR
105	**TS$**	String	Seconds blinking	" " / ":"	-	
106	**V**	Float	Mouse pointer sprite vertical position	Any	3063 / $0BF7	
107	**V$**	String	String cursor down	{down*23} - Constant	-	
108	**V0**	Float	Mouse sprite loop position vertical	0-24	-	
109	**V3**	Float	Window default [V0] position	0-255	-	
110	**V4**	Float	Backup [V0] before moving window	0-255	-	
111	**V7**	Float	Mouse movement [V] max. limit	Any	-	[Default: 244]
112	**V8**	Float	Mouse movement [V] min. limit	Any	-	[Default: 52]
113	**VR**	Float	Version number in System Registry	1.36	3050 / $0BEA	136 / 100 for version 1.36
114	**VT**	Float	Code version number	1.36	-	1.36 for version 1.36
115	**W1**	Float	Window creator: top-left [H0] position	Any	-	
116	**W2**	Float	Window creator: top-left [V0] position	Any	-	
117	**WD**	Float	Work drive #	8-11	3001 / $0BB9	
118	**WH**	Float	Window creator: window width	Any	-	
119	**WN$**	String	Window creator window name	Any	-	
120	**WS**	Float	Window creator: window slider	0 / 1	-	[0] Disable / [1] Enable
121	**WV**	Float	Window creator: window height	Any	-	

I. Config file [uos.cfg]

#	Position name	Values [DEC]	Default value [DEC]	Variable	Adress [DEC/$]
1	TRIANGULAR µOS version	136	136	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	223	BP	3068 / $0BFC
6	Desktop pattern color	Any	129	BC	3065 / $0BF9
7	Title bar color	Any	153	TC	3066 / $0BFA
8	Printer device #	0 or 4-7	0	PP	3061 / $0BF5

J. 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:

K. Changelog

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 bug 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 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
- Changelog is revised and integrated back into User’s Manual