C64 Debugger

by SLAJEREK/SAMAR

This is Commodore 64 code and memory debugger that works in real time.
It is quick prototyping tool where you can play with Commodore 64 machine
and its internals.

C64 Debugger embeds VICE v3.1 C64 emulation engine created by The VICE Team.

See a promo video here: https://youtu.be/_s6s7qnXBx8

documentation layout by fieserWolF / Abyss-Connection

C64 Debugger v0.64.56 manual

Table of Contents

Index of Tables

.

1 Preface

1.1 Installation

On Windows you need to install Visual Studio C++ 2008 Redistributable package.
On Windows 10 it is advised to run executable in Windows 7 compatibility mode.
Windows binary is now signed. Thanks to Yugorin/Samar for certificate donation!

On Linux you need GTK3 libraries.

1.2 Beer Donation

If you like this tool and you feel that you would like to share with me some beers, then you can use this link: http://tinyurl.com/C64Debugger-PayPal

Or send me some Bitcoins using this address: 1G3ZRT7j27QycHnkoo176t9j5a2J49fsXc

Donations will help me in development, thanks!

1.3 Facebook page

Join C64 Debugger Facebook page here: http://tinyurl.com/C64Debugger-Faceboo

2 Global keyboard shortcuts

Alt+Enter	Toggle fullscreen (MS Windows only)
Ctrl+F1	Show only C64 screen
Ctrl+F2	Show C64 disassembler, memory map and data dump
Ctrl+F3	Show C64 disassembler with hex codes, memory map, data dump and VIC state
Ctrl+F4	Show C64 and 1541 disk disassembler and memory maps
Ctrl+F5	Show states of chips
Ctrl+F6	Show C64 disassembler and a big memory map
Ctrl+F7	Show C64 and 1541 disk disassembler
Ctrl+F8	Show Monitor console and debugging tools
Ctrl+Shift+F1	Show zoomed C64 screen.
Ctrl+Shift+F2	Show cycle-exact debugging tools with C64 screen zoom and code labels
Ctrl+Shift+F4	Show VIC Display "lite" screen
Ctrl+Shift+F5	Show VIC Display screen
Ctrl+Shift+F6	Show VIC Editor screen
TAB	Change focus to next view
Shift+TAB	Change focus to previous view
F9	Show Main menu screen
Ctrl+B	Show Breakpoints screen
Ctrl+Shift+S	Show Snapshots screen
Ctrl+T	Mute sound On/Off
Ctrl+W	Replace memory dump view with watches view
Ctrl+[	Set slower emulation speed
Ctrl+]	Set faster emulation speed
Ctrl+8	Insert D64 file
Ctrl+Shift+8	Detach D64 file
Ctrl+O	Load PRG file
Ctrl+L	Reload PRG & Start
Ctrl+0	Attach cartridge
Ctrl+Shift+0	Detach cartridge
Ctrl+Shift+A	Toggle auto-load first PRG from inserted disk
Ctrl+F	Cartridge freeze button
Ctrl+R	Soft reset C64
Ctrl+Shift+R	Hard reset C64

Ctrl+Alt+R	Reset 1541 Disk drive
Ctrl+Shift+D	Detach everything
Ctrl+P	Limit emulation speed On/Off (warp mode)
Ctrl+Y	Use keyboard arrows as joystick On/Off, Right Alt to fire
F10	Pause code or run to next instruction (step)
Ctrl+F10	Step to next line (step over JSR)
Shift+F10	Run one CPU cycle
F11	Run/continue emulation
Ctrl+M	Toggle data memory map/dump taken directly from RAM or as-is with I/O and ROMs selected by $0001
Ctrl+E	Toggle show current raster beam position
Ctrl+S	Store snapshot to a file
Ctrl+D	Restore snapshot from a file
Shift+Ctrl+1, 2, 3, ..., 6	Quick store snapshot to slot #1,#2,#3, ..., or #6
Ctrl+1, 2, 3, ..., 6	Quick restore snapshot from slot #1,#2,#3, ..., or #6
Ctrl+U	Dump C64's memory to file
Ctrl+Shift+U	Dump 1541 Drive's memory to file
Ctrl+Shift+E	Save current screen data to file
Ctrl+BACKSPACE	Clear memory markers
Ctrl+Shift+P	Save C64 screenshot and sprite bitmaps to PNG files
F7	Browse attached disk image
F3	Start first PRG from disk image
Ctrl+;	Select next code symbols segment
Ctrl+'	Select previous code symbols segment

Table 1: global keyboard shortcuts

3 Debugger elements

3.1 Disassembly view

Mouse Click on memory address	Add/remove breakpoint
` (~ tilde key)	Add/remove breakpoint
Arrow Up/Down	Scroll code one instruction up/down
Page Up/Page Down or Shift+Arrow Up/Shift+Arrow Down	Scroll code by $100 bytes up/down
Space	Toggle tracking of code display by current PC
Enter	Enter code editing mode (assemble)
[or]	Scroll code one byte up/down
Arrow Left/Right	If not editing code: follow code jumps and branches using Right-Arrow key, and move back with Left-Arrow key. When argument is a memory address then Memory Dump view will be scrolled to that address If editing code and hex values visible: change edited hex value
CTRL+G <addr>	Move cursor to specific address (f.e. CTRL+G EA31)
CTRL+J	JMP to current cursor's address (change CPU PC)
Mouse wheel	Scroll code (faster with Shift pressed)

Table 2: disassembly view

.

3.2 Data dump view

Mouse Click on hex value	Select hex value
Double Mouse Click on hex value	Scroll disassemble view to selected address
Arrow keys	Move editing cursor
Page Up/Page Down or Shift+Arrow Up/Shift+Arrow Down	Scroll code by $100 bytes up/down
Enter or 0-9 or A-F	Start editing value
Ctrl+Mouse Click	Scroll Disassembly to code address that stored that value
Alt+Shift	Change CBM charset
Ctrl+K	Change colour mode on/off for sprites/characters
Ctrl+G <addr>	Move cursor to specific address (f.e. CTRL+G 0400)
Ctrl+V	Paste hex codes from clipboard into memory. Simple separators are parsed, also the text can contain addresses as 4 hex digits

Table 3: data dump view

3.3 Memory map view:

Memory map shows current values of memory cells. Ctrl+M switches bank to RAM. Each memory cell value is mapped into RGB or Gray or None. In RGB mode red are values from 0 to 85, green are values from 85 to 170 and blue are values from 170 to 255. In Gray mode all values are mapped into grayscale colors.

Memory access:

- white shows current PC
- blue marks read access
- red marks write access

You can change colours to ICU-standard (read marked by green) in Settings.

You can Mouse Click inside memory map to scroll data dump view to a clicked memory address. You can double Mouse Click to scroll disassemble view to a memory address under cursor. You can Ctrl+Mouse Click to scroll Disassembly to code address that stored value under cursor.

You can zoom-in using mouse wheel and move around by holding right mouse click (Windows, Linux, MacOS) or use mulitouch gestures such as pinch zoom and scroll using two fingers (MacOS only). You can select desired control behaviour in Settings.

3.4 Commodore 64 screen

All keys are mapped as original Commodore 64 keyboard. RUN+STOP is mapped to ESC key. Left Control key is not mapped and reserved for keyboard shortcuts.

Right Control is mapped into C64 Control key. RESTORE is not mapped, but you can chang this in Settings.

When joystick is turned on then you can control selected ports using arrow keys, and right-alt as fire.

3.5 SID state view

You can click waveforms to mute SID channels. Detected musical notes are displayed, these are based on standard 440Hz A4 notation.

3.6 VIC state view

This view shows state of VIC registers. You can lock colors using Mouse Left Click, or change them using Mouse Right Click, these will be reflected in previews like Memory Dump or VIC Display view.

3.7 VIC Display screen

The VIC Display screen is like an X-Ray for the VIC chip. Whole frame is recorded and you can access state of VIC and CPU for each cycle of the frame. It can be activated by Ctrl+Shift+F5. VIC Display renders exact state of VIC for selected cycle. As you know, a lot of effects are using tricks of the VIC chip, so it will not show the C64 screen in its entirety, as it is not meant to.
It will always show a screen how it would be rendered for selected cycle of the VIC chip.

You can just move mouse cursor over the VIC Display frame and see how VIC registers impact rendering of the C64 screen. Note that status of CPU registers and VIC state view is marked in light-red color background, this is to indicate that state is locked and shows selected raster cycle. The disassembly code is moved to the place where raster beam was executing code in the frame. Space bar changes disassembly code lock. Also, when you move the cursor over VIC Display, the memory dump view cursor points to address which is under mouse cursor.

When you click on the VIC Display you can lock cursor and then move it with keyboard arrow keys, holding Shift will increase the step. You can unlock the cursor by pressing Space Bar or by Mouse-Clicking on locked cursor.

There are buttons to control the VIC Display, you can see what are current values of VIC bank, screen, bitmap etc. and you can force and change them by clicking on values: green color means it is a current state for selected cycle, red is when you forced the selection. Do not forget, that if you select something making it red, then the VIC Display will show your selection, not what is currently going on on the screen.

button	function
Scroll	will switch if VIC scroll register should be applied to VIC Display position. When code is opening side borders then applying the scroll register may make the display jump a lot, so you can select if you need this behaviour.
Bad Line	shows a bad line condition when text is red, switching it on will display lines that are in bad line condition.
Border	changes if side border should be shown. It has three states: no border, viewable area with border, full frame scan.
Grid	changes if a grid should be displayed.
Sprites	changes if sprites graphics should be rendered in the VIC Display.
Frames	changes if frames around sprites should be visible.
Break	adds VIC raster breakpoint, the text is in red when a selected line has already the breakpoint set.
Show PC for	informs in which auto-scroll code disassembly mode we are, you can click on the mode and change it to other mode (Raster / Screen / Bitmap / Colour).
VIC Display	records state of VIC each cycle in the frame and with the mouse cursor you can see what is in the frame. The X key changes what we "look" at: where was the code in a given cycle (Raster mode) or where the code saved the pixel in memory (Raster / Screen / Bitmap / Colour). For Screen / Bitmap / Colour modes the memory view under C64 screen will be moved to address that holds the value at cursor. For charset mode the memory view cursor will point to a charset and definition of char under cursor.

Table 4: VIC display screen buttons

You can Right-Click on C64 Screen in right top to replace it to a zoomed raster view.

3.7.1 VIC Display screen keys

Arrow keys	Move locked cursor
Shift+Arrow keys	Move locked cursor in large steps
` (~ tilde key)	Toggle VIC raster breakpoint
L	Lock/Unlock mouse cursor
Space Bar	Lock/Unlock Disassemble auto-scroll code
X	Select next auto-scroll code mode
R	Select auto-scroll code to Raster
S	Select auto-scroll code to Screen (Text)
B	Select auto-scroll code to Bitmap
C	Select auto-scroll code to Colour

Table 5: VIC display screen keys

.

3.8 VIC Editor screen

The VIC Editor screen is for displaying and editing graphics in real time. All painting is done in a live C64 emulation and is immediately reflected in C64 RAM and VIC chip.

Layers window shows available layers, default layers are:

Unrestricted	you can paint on this layer in so-called hires unrestricted mode, thus using C64 colors without any limits
Sprites	this layer is used for painting on all visible sprites in this frame. These are virtual sprites, so in particular de-multiplexed sprites. Note, that displaying of virtual sprites from this layer is not implemented yet, thus the "V" button changes only if you can paint on this layer. However, you can see these sprites as they are rendered in the C64 Screen, so actually you can paint on them and see changes.
C64 Sprites	these are sprites that are in a raster line under cursor, just the same like in VIC Display view.
Display	this is the same VIC Display which works exactly the same way, thus exact state of VIC for selected cycle under cursor is rendered.
Reference	this works like Unrestricted but has full palette and images imported into that layer are displayed as-is. Painting on that layer is disabled by default, you can paint only when this layer is selected.
C64 Screen	this is the C64 Screen as it was rendered by VIC, note that if emulation is paused, then painting on this layer will not have immediate affect - the VIC must render the screen first to have changes visible.

Table 6: VIC editor screen layers

There are “V” buttons near layers names, these set visibility of the layer.

You can select the layer by clicking on it:

- When layer is selected, all painting is done on that selected layer, even if it is not visible.
- When no layer is selected, then painting is made from top-to-bottom, it is driven to a layer that has higher priority first, that is visible and has a pixel under selected x/y mouse position which is inside that layer (for example if there are no sprites under mouse cursor, then sprites layer will be skipped and painting will be done on C64 bitmap).

Painting depends on selected mode. In all modes you are free to paint, however if you exceed available number of colors the painting will be blocked. To un-block and force color replace you can hold Ctrl key (this can be configured in Settings).

The replacement color will be selected and it will be replaced:

- in Hires Bitmap this will be color under cursor in 8x8 char,
- in Multi-Color Bitmap this will be color that is less-used in 8x8 char (has least number of pixels),
- on Sprite this will always be individual sprite color ($D027+).

Painting with RMB On Sprite will always use background color ($D021).

You can paint in dither mode by holding Alt key: pixel colors are alternating between LMB and RMB. When you paint first pixel, a dithering grid will be created, and by holding Alt key this grid will be used for painting. The dithering grid will be reset when you release the Alt key.

Sprite window shows current selected sprite. You can lock selected sprite by clicking mouse on sprite with Ctrl+Shift. Then you can select color to use for paiting by clicking on the color in Sprite window or by pressing (0, Shift+1, Shift+2, Shift+3).
To change locked color just select a new color from palette, this will replace the color of locked Sprite. Note that $D021, $D025 and $D026 are shared with other sprites. If you not select color in Sprite window, then painting on Sprite will try to use selected color from the Palette, if color is not in available colors then the painting will be blocked and to un-block and force color replace use the Ctrl key.

You can change multicolor, horizontal or vertical stretch by clicking buttons.

You can change positions of Sprites the same way, just lock/select a Sprite by pressing Ctrl+Shift+Mouse Click on that sprite, and then use Arrow Keys to move that selected sprite (Arrow Key Left/Right/Up/Down).

Note, that changing colors, positions and settings of Sprites is quasi-intelligent:
in current frame's code that was run, places of LDA/LDX/LDY and STA/STX/STY pairs are found for sprite colors or positions and values of LDA's are replaced based on current raster position. Thus, you can write your own display and colors multiplexer code, run it and when you change colors of sprites the code in C64 RAM will be replaced accordingly, even if you use Sprite multiplexing.

Charset window shows current charset, you can select char and use it for painting in text modes.

It is possible to import PNG and convert it to current mode, hires/multicolor bitmap and hires text modes are supported. PNG must have resolution of 320x200 or 384x272.
Colors are matched to nearest C64 colors (nearest neighbour). For bitmap modes colors in 8x8 are set based on most used color values in 8x8, thus first a color that has largest number of occurrences in 8x8 char is found and it is replaced for converting in that 8x8 char, then in multicolor the another one, etc. If sprites are present in the screen, colors are matched to colors selected in the sprite and pixels are converted, note that no automatic color replacing is possible at this moment. The 384x272 resolution includes also borders, so if you have sprites in side border the pixels will be converted accordingly.

Note, that a workflow with sprites is that you should have some init PRG procedure that sets position of sprites. The converter is quasi-intelligent, thus is trying to find places of LDA/STA for colors in the current frame.

You can also import KLA (Bitmap Multi-Color), ART (Bitmap Hires), DD (Bitmap Hires) and export to KLA, ART or raw text depending on selected mode.

Note: when you export to raw text it contains these blocks:

$000-$3E7	Current character video memory (screen). For example values stored in $0400-$07E7.
$3E7-$7CE	Color memory (values stored in $D800-$DBE7)
$7CF	Background color (value stored in $D021)

Table 7: VIC editor screen: raw text export blocks

Zooming and panning of the canvas is performed using mouse, you can use Mouse Scroll for zooming and hold Space Bar for panning. Also, you can Mouse Right-Click on Preview Window to quickly move the painting area to selected position. When you zoom-in deeply then numbers such as pixel addresses and values will be also shown.

3.8.1 VIC editor shortcuts

(LMB=left mouse button, RMB=right mouse button)

Ctrl+N	Create new picture and setup C64 for painting
LMB, RMB	Paint using selected color
Alt+LMB, Alt+RMB	Paint dither
Ctrl+LMB or Ctrl+RMB	Force painting / replace color
Shift+LMB	Get color at cursor as LMB color
Shift+LMB	Get color at cursor as RMB color
X	Exchange LMB/RMB colors
0	Set LMB color from $D021 color
Shift+0	Get color at cursor as background ($D021) color
RMB on Preview Window	Move display
Space Bar (hold in main display)	Move display
Mouse Scroll	Zoom in/out the canvas
Shift+Mouse Scroll	Quickly zoom in/out the canvas
[or]	Select Circle Brush size
Ctrl+[or Ctrl+]	Select Rectangle Brush size
/	Change Preview Window scale
'	Show cursor pointer in Preview Window
` (tilde key)	Select next visible layer
12345678QWERTYUI	Select color
Shift+1, Shift+2, Shift+3	Select sprite painting color num ($D025, $D027+, $D026)
F	Show/hide all windows
D	Show/hide preview window
P	Show/hide colors palette

C	Show/hide character set window
S	Show/hide sprite window
L	Show/hide layers window
Ctrl+G	Show/hide sprite frames
Ctrl+Shift+Mouse Click	Lock/Select sprite
Arrow Left/Right/Up/Down	Move selected sprite
Ctrl+Backspace	Clear screen
Ctrl+Z	Undo
Ctrl+Shift+Z	Redo
Ctrl+S	Save image in VIC Editor (*.vce) format
Ctrl+O	Load/Import image (vce, png, kla, art, dd)
Ctrl+Shift+E	Export image to kla/art/raw text
Ctrl+B	Toggle top bar with icons
ESCAPE	Back to C64 Debugger

Table 8: VIC editor shortcuts

3.9 Monitor screen

You can use these instructions in code monitor:

HELP	shows help
DEVICE C / D / 8	set current device (C64/Disk/Disk)
F <from address> <to address> <value>	fill memory with value
C <from address> <to address> <destination address>	compare memory with memory
H <from address> <to address> <value> [<value> ...]	compare memory with values
T <from address> <to address> <destination address>	copy memory
L [PRG] [from address] [file name]	load memory (with option from PRG file)
S [PRG] <from address> <to address> [file name]	save memory (with option as PRG file)
D [NH] <from address> <to address> [file name]	disassemble memory (with option NH without hex codes)
G <address>	jmp to address

Table 9: monitor screen instructions

.

3.10 Breakpoints

Breakpoint stops the execution of code depending on some state and situation.

3.10.1 Breakpoints screen

In the Breakpoints screen (Ctrl + B) you can click using mouse, or Enter or Space key to enable or disable monitoring of selected type of the breakpoint.
New value can be added by selecting "...." either by moving the cursor with the arrow keys or clicking using mouse.

These are possibilities:

VIC / CIA / NMI	stops when selected interruption occurs
CPU PC	the code will stop as the processor will start to perform instruction from selected address
MEMORY	stops when there will be attempt to write to the memory of the set value, for example: 4FFF <= 3F will stop code when there will be attempt to write to the cells 4FFF value less or equal to 3F. To break at any write access you can use <= FF
RASTER	stops when raster reaches the set line value

Table 10: breakpoint types

Breakpoints CPU type PC can also be set in the disassembler view by clicking the mouse cursor on the address or by pressing the ` (tilde) key.

The same applies to 1541 Drive breakpoints on right side of the screen.

3.10.2 Breakpoints screen keys

Arrow keys	Move around
Enter or Spacebar	Toggle value or start editing breakpoint

Table 11: breakpoints screen keys

3.10.3 Breakpoints file

Breakpoints file stores information about breakpoints, addresses and values.
Possible entries are:

break xxxx - break when PC reaches address xxxx	Example: break 3FFF
breakraster xxx - break when raster reaches line number xxx	Example: breakraster 40
breakmem xxxx oo yy - break on memory write to address xxxx when	expression oo yy is true. Possible operators oo are: ==, !=, <, <=, >, >= Example: breakmem D018<=FF
breakvic	break on VIC interrupt
breakcia	break on CIA interrupt
breaknmi	break on NMI interrupt
setbkg xxxx yy	fake marker, when PC reaches address xxxx then background colour register ($D020/$D021) is set to value yy, you can mix this type of breakpoint with normal "break" to also stop code execution

Table 12: breakpoints file entries

All entries are not case sensitive. Please check KickAssembler documentation, section 9.5: Writing to User Defined Files.

4 Invoking the debugger

4.1 Command line options

-help	show help
-layout <id>	start with layout id <1-12>
-breakpoints <file>	load breakpoints from file
-symbols <file>	load symbols (code labels)
-watch <file>	load watches
-wait <ms>	wait before performing tasks
-prg <file>	load PRG file into memory
-d64 <file>	insert D64 disk
-tap <file>	attach TAP/T64
-crt <file>	attach cartridge
-jmp <addr>	jmp to address, for example jmp x1000, jmp $1000 or jmp 4096
-autojmp	automatically jmp to address if basic SYS is detected
-alwaysjmp	always jmp to load address of PRG
-autorundisk	automatically load first PRG from inserted disk
-unpause	force code running
-snapshot <file>	load snapshot from file
-soundout <"device name" | device number>	set sound out device by name or number
-playlist <file>	load and start jukebox playlist from json file
-clearsettings	clear all config settings
-pass	pass parameters to already running instance if instance is not running a new one will be spawned

Table 13: command line options

Other command line options are the same as selected emulation engine (thus see Vice documentation for additional command line options).

4.2 Code labels (symbols)

You can load a symbols file wit code lables via -symbols <file> command line option. Also, if near loaded PRG a file with "labels" file extension is found then it is loaded automatically. Two file formats are accepted, a standard Vice code labels format and 64Tass compatible file format.

Vice code labels file format example:
al C:d019 .vic2_int_reg

Note, that label name's leading dot is skipped.

64Tass labels file format example:
vic2_int_reg = $D019

4.3 Watches

Watches view is a simple way to display selected values in memory with a label.
You can replace the memory dump view by watches view with Ctrl+W key. The feature is simple display of hex value stored in associated memory address, but this will be expanded in future to allow also different data representations.

To add watches you can do that only via external file that you can load from a command line.

The format of file is simple, and there are two formats accepted.

Simple watches format example:
d019 vic2_int_reg

64Tass-labels compatible format example:
vic2_int_reg = $D019

For example watches file please refer to:
https://sourceforge.net/p/c64-debugger/code/ci/master/tree/Examples/example.watch

4.4 KickAss debug symbols

With Mads Nielsen (Slammer/Camelot) we created integration based on Stein Pedersen's idea.
This was written with great help of Mads Nielsen:

4.4.1 C64Debugger - KickAss format

Here is the basic format. To make it easier to read I have given a param named 'values' that explains the values of the comma separated lists.

```
<C64debugger version="1.0">
  <Sources values="INDEX,FILE">
    0,KickAss.jar:/include/autoinclude.asm
    1,/Users/Mads/Code/C64CodeRepos/C64Code/atari/lib/atarifile_4bank.h
  <Sources/>

  <Segment name="BANK1" values="START,END,FILE_IDX,LINE1,COL1,LINE2,COL2">
```

```
    <Block name="Program">
      $1000,$1002,2,16,9,16,11
    <Block/>
    <Block name="Vectors">
      $1ffa,$1ffb,2,11,3,11,7
      $1ffc,$1ffd,2,12,9,12,13
      $1ffe,$1fff,2,13,9,13,13
    <Block/>
  <Segment/>

  <Segment name="BANK2" values="START,END,FILE_IDX,LINE1,COL1,LINE2,COL2">
    <Block name="Program">
      $1000,$1000,2,28,3,28,5
      $1001,$1001,2,29,3,29,5
      $1002,$1002,2,30,3,30,5
      $1003,$1004,2,35,3,35,5
      $1005,$1006,2,36,3,36,5

    <Block/>
    <Block name="Vectors">
      $1ffa,$1ffb,2,23,3,23,7
      $1ffc,$1ffd,2,24,9,24,13
      $1ffe,$1fff,2,25,9,25,13
    <Block/>
  <Segment/>

  <Labels values="SEGMENT,ADDRESS,NAME">
    Default,$d011,vic2_screen_control_register1
  </Labels>

  <Watches values="SEGMENT,ADDRESS,ARGUMENT">
        Default,$3000
        Default,$2001,2,hex8
        BANK2,$3000,,text
  </Watches>

  <Breakpoints values="SEGMENT,ADDRESS,ARGUMENT">
    BANK1,$1000,nmi
    BANK2,$1003,
  <Breakpoints/>
<C64debugger/>
```

So everything is inside a <C64debugger> tag with a version number. Inside are different tags:

tag	description
<C64debugger>	<C64debugger> tag with a version number
<Sources>	There will always be one <Sources> tag with all the source files and their indices.
	There will be one or more tags - one for each segment. Segments contains zero or more <Block> tags and inside these are the usual debug data.

<Breakpoints>	There will always be one <Breakpoints> tag. It contains one line for each breakpoint. First arg is the segment it is defined in (so if you turn on and off segments you can switch breakpoints on an off too). Second argument is the address it is defined at (You will not need it in eg. .break "nmi", but it is always there). Third argument is whatever the user writes in the .break argument and might be empty. So .break "nmi" and .break "cia" will give nmi and cia.
<Labels>	<Labels> tag adds a label at address. First argument is the segment name, second argument is the address and last argument is label text.
<Watches>	<Watches> is similar to labels but it will appear in watches view. First argument is the segment name, second argument is the address, then third argument is number of values to display, and fourth argument declares a representation which can be: hex8, hex16, hex32, or simply h, h8, h16, h32 is hex representation of value interpreted as 8, 16 or 32 bits. signed8, signed16, signed32, or simply s8, s16, s32 is a signed decimal representation of value interpreted as 8, 16 or 32 bits. unsigned8, unsigned16, unsigned32 or simply u8, u16, u32 is an unsigned decimal representation of value interpreted as 8, 16 or 32 bits. text signifies text representation.

Table 14: KickAss debug symbols tags

Please note that representation and number of values are not yet displayed in Watches view, this will be updated in upcoming version. Now, the Watches view displays only one hex 8-bit value.

4.5 JukeBox playlist and automated tests

JukeBox playlist is a way to automate things in the C64 Debugger. The idea is that you can write a JSON file in which actions and settings for the jukebox are set.
Examples of usage include:

- simply playing demos from a playlist, with fade out/fade in transitions, good for the big screen!
- set warp speed on, load demo, set warp off, automatically press space on notes, then dump memory in selected frames.
- load game, automatically move joystick, etc.

Note, that all timings selected are in seconds or milliseconds, but are re-calculated to VIC synchronization frames, so the timings will be always exact and synced to VIC refresh. The frame number depends on selected system (PAL, NTSC). Calls to move on with transitions, dumping memory and other actions are always synchronized to VIC and are performed at end of each VIC frame.

JSON format is as follows:

1. All "global" settings, such as if fast boot kernal patch should be included.
2. Entries for each file load.
3. Each entry has its own settings (such as file path) and actions (such as key strokes, joystick movements, memory dumps, etc).

For example file please refer to:
https://sourceforge.net/p/c64-debugger/code/ci/master/tree/Examples/jukebox-win32.json

4.5.1 Global settings variables

FastBootPatch=true/false	should kernal be patched with fast boot patch
DelayAfterReset=real number	pauses all actions after machine reset for selected number of milliseconds
ShowLoadAddress=true/false	shall the load address be displayed on screen?
FadeAudioVolume=true/false	should the audio be faded out/in on transitions?
SetLayout=integer number	set layout number on start
ShowPopup=true/false	should popup with demo details be displayed on transition?
PopupFadeTime=real number	duration time of fade out/in popup
PopupVisibleTime	duration time of popup visibility
Playlist=[]	array of playlist entries

Table 15: JukeBox playlist global settings variables

In Playlist array there are entries of files that will be loaded, in order. Each entry has its own seetings and actions.

Playlist Entry variables:

Name=string	name of demo/program to be displayed in popup
FilePath=string	path to a file to be loaded (can be d64, prg, crt or snap)
ResetMode=hard/soft	which reset mode should be used before loading this file
AutoRun=true/false	should file be auto run (auto run means: perform reset, load file and run)
RunFile=integer number	which entry from D64 directory should be loaded
WaitTime=real number	for how long this entry should be played, wait time before transition to next entry
DelayAfterReset=real number	pauses all actions after machine reset for selected number of milliseconds
FadeInTime=real number	time of fade in transition at start of this entry
FadeOutTime=real number	time of fade out transition at end of this entry
Actions=[]	array of actions to be performed during this entry

Table 16: JukeBox playlist entry variables

In Actions array there are actions that will be performed during playing of this entry.

4.5.2 Action object variables

DoAfterDelay=real number	Wait selected number of seconds and perform action.
KeyDown=string (one ASCII character)	Push and hold key on C64 keyboard. The key is ASCII character.
KeyUp=string (one ASCII character)	Key up and do not hold anymore a key on C64 keyboard. The key is ASCII character.
KeyDownCode=integer number	Push and hold key on C64 keyboard. The key is selected by its ASCII code. For list of special scan codes refer to: https://sourceforge.net/p/c64-debugger/code/ci/master/tree/MTEngine/Engine/Core/SYS_KeyCodes.h
KeyUpCode=integer number	Key up and do not hold anymore a key on C64 keyboard. The key is selected by its ASCII code.
Joystick1Down=string or Joystick1Up=string or Joystick2Down=string or Joystick2Up=string	Push selected joystick axis (Down) or release joystick axis (Up). The axis name is a string of these values: fire, up, down, left, right, sw, nw, se, sw
WarpMode=true/false	Set warp mode On/Off
DumpC64Memory=string	Dump C64 Memory to a file specified by path
DumpDiskMemory=string	Dump Disk drive Memory to a file specified by path
DetachCartridge=true/false	Detach (remove) cartridge from slot
SaveScreenshot=string	Save Screenshot as PNG to file specified by path

ExportScreen=string	Export Screen as kla/art/raw text to file specified by path. The file extension will be added automatically based on current C64 display mode
Shutdown	Shutdown the C64 Debugger (Quit program)

Table 17: JukeBox playlist action object variables

5 Appendix

Step over JSR works in a way that a temporary PC breakpoint is created in next line. Code will be stopped when PC hits that breakpoint address, in most situations just after returing from JSR. Note that if code never returns from JSR this breakpoint will still be "valid".

You can also drag & drop file into C64 Debugger window on MacOS & Windows. Depending on selected option in Settings the file can be auto-started, also from disk image file.

You can browse the contents of attached disk image by pressing F7 key, and run the first PRG by F3 key. Note that if C64 Screen is selected then these keys are normally sent to the C64. Thus to let these key shortcuts work you need to first un-select the C64 Screen.

5.1 Known bugs

When snapshot is loaded then selected settings are not updated in the Settings menu (such as SID type, C64 machine model, attached disks, etc).

Loading NTSC snapshot into PAL machine or vice-versa is not supported and will hard reset the C64.

It is not possible to zoom Drive 1541 memory map.

Clicking Drive 1541 memory map does not properly set selected value in memory dump view.

Command line arguments are passed to VICE. VICE complains that arguments that have been parsed by C64 Debugger are not OK.

On some window managers flavours in Linux system open/save file dialogs are behaving incorrectly.

When you move a Sprite in VIC Editor and Sprite is on top of other Sprite they will 'pile up', also there are no means to select Sprite below a Sprite... this is not ready yet and is planned for next release.

5.2 To do

- Add memory map zooming for Drive 1541.
- Add working on files directly instead of C64 memory (file adapter is ready), to view/edit files directly.
- Add custom layouts with layout editor.
- Add PAL CRT emulation.
- Add Save Screenshot keyboard shortcut.

5.3 Thanks for testing

- Mr Wegi/Elysium - valuable suggestions and cartridge knowledge
- ElfKaa/Avatar

- Don Kichote/Samar
- Isildur/Samar
- Yugorin/Samar
- Scan/House
- Dr.J/Delysid
- Brush/Elysium
- Ruben Aparicio
- 64 bites
- Stein Pedersen
- Mads Nielsen

5.4 Beer Donation

If you like this tool and you feel that you would like to share with me
some beers, then you can use this link: http://tinyurl.com/C64Debugger-PayPal

5.5 Contact

If you have ideas or found a bug feel free to contact me at slajerek@gmail.com

5.6 License

5.7 Acknowledgements

Portions of this Software may utilize the following copyrighted material,
the use of which is hereby acknowledged:

5.7.1 VICE License

5.7.2 Commodore ROMs

5.7.3 Libraries

libjpeg	is a free software library written for JPEG image compression.
libjson	Copyright 2010 Jonathan Wallace. All rights reserved.

libpng version 1.5.2	March 31, 2011 Copyright (c) 1998-2011 Glenn Randers-Pehrson (Version 0.96 Copyright (c) 1996, 1997 Andreas Dilger) (Version 0.88 Copyright (c) 1995, 1996 Guy Eric Schalnat, Group 42, Inc.)
LodePNG	version 20140801 Copyright (c) 2005-2014 Lode Vandevenne
minizip	Version 1.01e, February 12th, 2005 Copyright (C) 1998-2005 Gilles Vollant
mtrand	Coded by Takuji Nishimura and Makoto Matsumoto. Ported to C++ by Jasper Bedaux 2003/1/1 (see http://www.bedaux.net/mtrand/). The generators returning floating point numbers are based on a version by Isaku Wada, 2002/01/09 Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura, All rights reserved.
utf8	Copyright 2006 Nemanja Trifunovic
zlib	Copyright (C) 1995-2006, 2010 Mark Adler
portaudio	Copyright (c) 1999-2002 Ross Bencina and Phil Burk
pthread-win32	https://www.sourceware.org/pthreads-win32/
mman-32	https://github.com/witwall/mman-win32
libclipboard	Copyright (c) 2016 Jeremy Tan https://github.com/jtanx/libclipboard

Table 18: libraries

5.8 Change log

v0.64.56

- Added: PALette palette.
- Added: CHARSET mode in VIC Display
- Added: Joystick keys can be defined as regular keyboard shortcut keys
- Added: New actions to JukeBox scripting: save screenshot, export screen to kla/art/raw, shutdown the C64 Debugger
- Added: Edit values of registers in VIC/SID/CIA/VIA state views
- Added: Quick disassemble mouse scroll with Shift pressed
- Added: Copy (Ctrl+C) and Paste (Ctrl+V) assembly line as text in Disassembly view
- Added: Copy (Ctrl+C) and Paste (Ctrl+V) value (or address with Shift pressed) in Memory Dump view
- Added: Create new Picture in VIC Editor using Ctrl+N
- Added: Export and import Charset in VIC Editor
- Added: Export and import Sprite in VIC Editor
- Added: New unrestricted image Reference layer in VIC Editor
- Added: TAP/T64 load and Tape menu in Settings (thanks to Pontus Berg and Josefin Svensson for reminding me this!)
- Added: Loading sources and debug info from new *.dbg file format. You can view diassembled code together with original source code in new Ctrl+Shift+F3 view. Thanks to Mads Nielsen for his valuable suggestions on the integration of the debug info file format in his KickAssembler. Idea by Stein Pedersen.
- Added: Atari 800/65XE emulator aka 65XE Debugger (early draft)
- Added Linux: New compile Makefile by Kuba Skrzypnik, Eclipse is no more needed to compile sources!
- Changed: Cycler layout (Ctrl+Shift+F2) now contains also memory dump
- Changed: Ctrl+O starts a generic file dialog to open any supported file
- Bug Fixed: Drive CPU status was not displayed in Monitor view when Drive device was selected (thanks to Javier Martin for bug report)
- Bug Fixed: Markers are cleared automatically after PRG load (thanks to Alex Goldablat for bug report)
- Bug Fixed: Zooming in memory map was sometimes blocked
- Bug Fixed: Command line -help option now properly displays message box on Windows (thanks to Timsa Uotila for bug report)
- Bug fixed: Painting on vertically-stretched multicolor sprite caused crash (thanks to Isildur/Samar for bug report)

v0.64.2

- Bug fixed: VIC Sequencer state was displayed reversed (thanks to Mattias Weidhagen for bug reporting)
- Bug fixed: Muting a channel in Stereo/Triple SID state view did not work correctly
- Bug fixed: Automatic loading of *.watch file sometimes caused lock of the debugger (thanks to Yugorin/Samar for bug reporting)
- Bug fixed: Automatic focus for C64 screen was not triggered (thanks to Isildur/Samar for bug reporting)
- Bug fixed: Crash when ReSID emulation was selected and Run SID emulation option was set to No (thanks to Isildur/Samar for bug reporting)

- Bug fixed: When PRG was selected from command line and disk was attached with autorun set then the file entry from D64 was started instead of PRG (thanks to Isildur/Samar for bug reporting)
- Added: Saving VIC Display state with VCE file (thanks to Isildur/Samar for suggestion)

v0.64 (2017/12/24), X-Mas release!

- Added: JukeBox playlist feature! Allows to play your favourite demos from playlist with transitions, run automated tests of your games and programs with keystrokes and joystick movements, run your productions in Warp mode and then do a memory dump after selected time... and more!
- Added: BASIC-compatible auto run
- Added: Setting CPU registers value in registers view
- Added: Setting for Stereo and Triple SID, showing registers of additional SIDs in SID state view
- Added: Switch off SID emulation in Settings
- Added: Mute audio using Ctrl+T shortcut, also select switch mute mode between just muting the volume, or switching SID emulation off when muted, selectable in Settings (thanks to Mojzesh/Arise and Wegi/Elysium for the help and idea)
- Added: Support of 64tass code labels
- Added: Automatically load Vice code labels if file with *.labels extension is found near loaded PRG
- Added: Watch selected memory locations (Ctrl+W), automatically load *.watch file with PRG. Simple for now, update soon!
- Added: Change menus colour theme and disassembly colour theme, new menus colour themes by Mojzesh/Arise and Isildur/Samar
- Added: Export sprite raw data with screen save
- Added: Show multi-colour charset in Vic Editor
- Added: Setting to adjust focus border width
- Change: You can now save current screen using Ctrl+Shift+E keyboard shortcut in any view, not only Vic Editor
- Change: Saving current screen to file also exports sprites data and charset data
- Change: Shift+0 in Vic Editor sets both $D020 and $D021 colors
- Bug fixed: On Windows it was not possible to enter opcodes in the disassembly pane due to keycodes mismatch (thanks to Scan/House for bug report)
- Bug fixed: On MacOS accent keys that needed double keystroke on ISO keyboards were not recognised (thanks to Ruben Aparicio for bug report and great help with fixing)
- Bug fixed: Importing key map from file caused corruption in key map editor (thanks to Ruben Aparicio for bug report)

v0.62 (2017/08/02), released at Riverwash demo party

- Added: MIDI support, the usual -midi* command line flags work as they normally do in VICE itself. Thanks to David Hogans for help
- Added: Select audio out device via command line (-soundout <"device name" | device number>)
- Added: Quick workaround for Linux open/save file dialogs problems on broken GTK, you can select custom open/save file dialogs in Settings/UI (no UTF support yet, sorry!)
- Bug fixed: Loading PRG while waiting after automatic Reset for previous PRG load caused Fatal Error
- Bug fixed: Painting on vertically-stretched sprite caused crash
- And other overall tweaks here and there.

v0.60 (2017/06/23), released at Silesia 8 demo party.
See a promo video here: https://youtu.be/_s6s7qnXBx8

- Added: Integrated Vice 3.1 emulation engine
- Added: new VIC Display screen (Ctrl+Shift+F5) and VIC Display lite (Ctrl+Shift+F4)
- Added: new VIC Editor screen (Ctrl+Shift+F6). Simple for now, more features on the way!
- Added: show music notes in SID State view
- Added: you can Ctrl+Click on Memory Dump or Memory Map view to scroll Disassembly to code address that stored that value
- Added: you can follow code jumps and branches in Dissasembly view using Right-Arrow key, and move back with Left-Arrow key, when argument is a memory address then Data Dump view will be scrolled to that address
- Added: colors are shown in VIC State, also you can lock (Left Click) or force (Right Click) these colors in previews
- Added: show code cycles in some Disassembly views
- Added: setting to completely stop SID emulation when in warp mode
- Added: setting to select nearest or billinear interpolation mode for rendering of the C64 Screen in Settings
- Added: setting to select VIC colors palette in Settings
- Added: reset only disk drive by Ctrl+Alt+R
- Added: zoomed full screen in Ctrl+Shift+F1
- Added: save C64 screenshot and sprite bitmaps to PNG files by Ctrl+Shift+P
- Added: key shortcut to browse and run PRG files from attached disk image (F7)
- Added: key shortcut to auto run first PRG file from the attached disk image (F3)
- Added: setting and command line option to auto load and run first PRG from inserted disk
- Added: key shortcut to switch auto run from disk (Ctrl+Shift+A)
- Added: setting and command line option to always jmp to loaded PRG address even if no Basic SYS is detected
- Added: setting and command line option to un-pause debugging code when PRG is loaded
- Added: setting to reset or hard reset C64 before starting PRG
- Added: key shortcut to detach disk image (Ctrl+Shift+8), cartridge (Ctrl+Shift+0) and everyting (Ctrl+Shift+D)
- Added: you can drag & drop file into C64 Debugger window on MacOS & Windows
- Added: mouse cursor is hidden when window is full screen, and only C64 Screen is shown (in Ctrl+F1 view)
- Change: default key mapping of OS '\' key changed to C64 key '='
- Change: default VIC colors palette changed to colodore
- Change: default SID model changed to 8580 FastSID
- Change: in Disassembly view you can move cursor to current address -1 by [key, and to address +1 by] key
- Change: Settings menu is now split into sub-menus
- Bug fixed: processor status flags were not correctly updated for N and Z flags (thanks to Flavioweb/Aura^Hokuto Force for reporting)
- Bug fixed: when loading PRG additional space in Basic SYS was not properly parsed giving wrong start address (thanks Yugorin/Samar for reporting)
- Bug fixed: when SYS is hidden by $00 trick the address was not properly parsed (thanks Yugorin/Samar for reporting)
- Bug fixed: stored folders paths for D64/PRG/CRT were not properly set in macOS Sierra open/save dialogs

- Bug fixed: the PC breakpoint did not stop code execution when it was placed on first instruction after manual jump or IRQ, now it's properly trapped (thanks to 64bites for reporting)
- Bug fixed: idle CIA timers were not properly updated when emulation was paused or in single stepping mode (thanks Scan/House Designs for reporting)
- Bug fixed: code labels are properly placed in disassemble view after PRG file load
- Bug fixed: drive memory breakpoints were not correctly set
- Bug fixed: menu items for resetting the C64 were not properly handled
- Bug fixed: some another not done key mappings on Windows reported by Isildur/Samar ;#)
- Windows binary is now signed. Thanks to Yugorin/Samar for donation!!

v0.56

- Bug fixed: Loading of PRG is now always to RAM (skipping I/O), not based on value of $01 as previously (thanks DKT/Samar for spotting this)
- Bug fixed: Displaying Sprite bit states in compact VIC (Ctrl+F3) was showing repeated states 1-4 for 5-8 and Sprite Exp states were displayed only for Sprite #1 (thanks Scan/House for a bug report)
- Bug fixed: When no output audio device was found the debugger was closed silently on startup throwing error only to system console. Now additional error message box is displayed that audio device is missing (thanks Isildur/Samar for a bug report)
- Added: "pass" command line option to pass parameters to already running instance
- Added: S PRG function in monitor console to save memory dump as a PRG file
- Added: L PRG function in monitor console to load memory from a PRG file
- Added: D function in monitor to disassemble code, also to text file
- Added: Setting to adjust fade out speed of memory markers
- Added: Setting to customise grid lines and raster cross colors
- Added: Setting to show debugger window always on top
- Added: Paste (Ctrl+V) hex data from system clipboard into RAM in memory dump view

v0.54 (2016/09/03), released at Riverwash Demoparty 2016

- Bug fixed: S command in monitor saves last address byte inclusive as in VICE monitor
- Bug fixed: Memory map was showing wrong values in $0000 and $0001
- Bug fixed: Audio output is reactivated when emulation speed is higher than 10% (thanks Scan/House for bug report)
- Bug fixed: Cycle-by-cycle screen refreshing tweaks. 8 additional pixels for each VIC cycle were painted and sometimes one not needed additional line in last VIC cycle was copied, that caused over-painting of whole spurious background line to a current raster line
- Bug fixed: Breakpoints loaded from command line were not displayed in disassembly view
- Bug fixed: Memory breakpoints less & greater were checked inversely
- Bug fixed: Windows: shifted keys are again working (damn Windows WinAPI hell!). For C= + Shift press first Shift and then Left ALT
- Changed: When hex codes are not visible in disassembly view then all ???s are displayed as hex codes
- Added: Keyboard shortcuts to control emulation speed (CTRL+[and CTRL+])
- Added: Option in settings to switch on/off the PC-execute-aware disassemble (switch to use straight disassemble as in any monitor instead of PC-execute-aware)
- Added: New cycler-view (Ctrl+Shift+F2) for cycle-exact code debugging, with VIC states, code labels and zoomed C64 screen (view suggested by Brush/Elysium)
- Added: Loading and viewing Vice labels by new command line option: -vicesymbols <file-name>, visible in cycler-view.

- Added: autojmp command line option
- Added: New 0-cycle background value action for breakpoint. In breakpoints file you can set a background for PC address with "setbkg <addr> <value>"

v0.52 (2016/06/25)
- Bug fixed: Key "7" was not mapped to C64 (thanks Wolfram Heyer for spotting this)
- Bug fixed: Basic pointers $2D-$32, $AE/$AF were initialised when PRG is loaded and basic SYS is detected. That caused some decrunchers to not work properly when PRG was started automatically (thanks Michael Tackett for reporting and iAN CooG for help)
- C64 keyboard mapping screen in Settings
- Mapping keyboard shortcuts screen in Settings
- Mapping of C64 memory to a file (read/write via mmap on MacOS/Linux, read-only on Windows)
- Select Audio Output device in Settings
- Apply fast boot kernal patch in Settings
- When CPU is in jam state then CTRL+R will start running emulation automatically (thanks Marc Schoe Nefeld for suggestion)
- Hi-res sprites in VIC state are rendered with their colours if colour data rendering is selected (change with CTRL+K)
- Emulation Speed parameter in Settings
- Shortcut to Clear memory markers
- Save memory state & access markers to a CSV file (suggested by Wackee)

v0.5 (2016/06/04)
- First public release at "Stary Piernik 11", Torun 2016

v0.41
- PAL/NTSC machine model select
- Fixed fullscreen problem on Windows

v0.4
- Memory map zoom and better marking of code-execute.
- Bug fixes.

v0.32
- Bug fixes.

v0.31
- Step over JSR (Ctrl+F10), thanks Mr Wegi/Elysium for suggestion.
- Execute-aware code disassemble.
- Quick store & restore snapshots ([Shift+] Ctrl+1,2,...)
- UI tweaks suggested by Isildur.

v0.3
- Mark code execution (thanks Mr Wegi/Elysium for suggestion)
- Code monitor with basic commands DEVICE, F, C, H, T, L, S, G. (thanks DKT for suggestion)

v0.22

- Additional Settings: choose SID model and SID engine, ICU colours scheme, mute SID on pause, select joystick port, detach everything
- Store: default folders per file type, last screen layout
- Settings are stored and restored on startup
- Linux tweaks
- Bug fixes

v0.21 (2016/04/30)
- Cartridge bank peek bug fixed, found by Mr Wegi/Elysium (thanks!)
- SID state bug fixed and waveform views added
- All data is now embedded into executable
- Code optimizations

v0.2 (2016/04/23)
- Cartridge support and memory peek
- 1541 drive breakpoints and debugging
- Added to command line: 'wait', 'layout', 'cartridge'
- Bug fixes (thanks Isildur)
- Overall UI tweaks

v0.11 (2016/04/17)
- VICE chips state is displayed (including sprites)
- Some UX changes suggested by eLK/Avatar (thanks!)

v0.10 (2016/04/10)
- First Vice integration preview

v0.03 (2016/03/30) aka "Samar meeting version"
- Bug fixes.

v0.025 (2016/03/26)
- Tweaked disassemble code functionality
- Added: VIC/SID/CIA/Disk state screen
- Added: assemble mnemonics is possible in code view by pressing ENTER key
- Added: Linux GTK3 open/save dialogs

v0.024 (2016/03/19)
- Cleaned engine. Code refactoring
- Mangled keyboard shortcuts a bit
- Added: mouse wheel scroll now works
- Added: 1541 disk breakpoints

v0.023 (2016/03/12)
- You can click inside memory map to scroll data dump view.
- Added: reading breakpoints file
- Added: command line options
- Added: Ctrl+G for goto address in memory dump & disassemble
- Added: in disassemble view: Ctrl+J make JMP to address shown by cursor

v0.022 (2016/03/05)

- Traversing views in a main screen using TAB or Shift+TAB keys.
- Added: Show current raster beam position (Ctrl+E)
- Added: Snapshots menu. Store and restore full snapshots at any emulation cycle

v0.021 (2016/02/28)
- Fixed mapping of some keys on Windows (thx DKT & Isildur)
- ESC key returns, Alt+F4 closes app
- Added: Data dump view shows characters and sprites (Ctrl+K for colour mode)

v0.02 (2016/02/27)
- Loading PRG automatically starts if SYS basic command is detected
- Disassembled code can be scrolled using keyboard
- Added: Data dump screen with hex editing
- Added: Memory breakpoints

v0.01-test2 (2016/02/20)
- Added: Breakpoints screen
- Added: Settings screen

v0.01-test1 (2016/02/15)
- First internal release