Commodore 64-specific information for cc65

Ullrich von Bassewitz
Greg King

An overview over the C64 runtime system as it is implemented for the cc65 C compiler.

---

1. Overview

2. Binary format

3. Memory layout

4. Linker configurations

5. Extras

6. Platform-specific header files

7. Loadable drivers

8. Limitations

9. Other hints

10. License

---

1. Overview

This file contains an overview of the C64 runtime system as it comes with the cc65 C compiler. It describes the memory layout, C64-specific header files, available drivers, and any pitfalls specific to that platform.

Please note that C64-specific functions are just mentioned here, they are described in detail in the separate function reference. Even functions marked as "platform dependent" may be available on more than one platform. Please see the function reference for more information.

2. Binary format

The standard binary output format generated by the linker for the C64 target is a machine language program with a one line BASIC stub, which calls the machine language part via SYS. This means that a program can be loaded as BASIC program and started with RUN. It is of course possible to change this behaviour by using a modified startup file and linker config.

3. Memory layout

cc65 generated programs with the default setup run with the I/O area and the kernal ROM enabled (memory under the kernal may be used for graphics or as extended memory - see the sections about graphics and extended memory drivers). The BASIC ROM is disabled, which gives a usable memory range of $0800 - $CFFF. This means that kernal entry points may be called directly, but using the BASIC ROM is not possible without additional code.

Special locations:

Text screen

The text screen is located at $400 (as in the standard setup).

Stack

The C runtime stack is located at $CFFF and growing downwards.

Heap

The C heap is located at the end of the program and grows towards the C runtime stack.

4. Linker configurations

The ld65 linker comes with a default config file for the Commodore 64, which is used via `-t c64`. The c64 package comes with additional secondary linker config files, which are used via `-t c64 -C <configfile>`.

4.1 default config file (`c64.cfg`)

The default configuration is tailored to C programs. It supplies the load address and a small BASIC stub that starts the compiled program using a SYS command.

4.2 `c64-asm.cfg`

This configuration is made for assembler programmers who don't need a special setup. The default start address is $801. It can be changed with the linker command line option `--start-addr`. All standard segments with the exception of `zeropage` are written to the output file and a two byte load address is prepended.

To use this config file, assemble with `-t c64` and link with `-C c64-asm.cfg`. The former will make sure that correct character translation is in effect, while the latter supplies the actual config. When using `cl65`, use both command line options.

Sample command line for `cl65`:

```
cl65 -o file.prg -t c64 -C c64-asm.cfg source.s
```

To generate code that loads to $C000:

```
cl65 -o file.prg --start-addr $C000 -t c64 -C c64-asm.cfg source.s
```

It is also possible to add a small BASIC header to the program, that uses SYS to jump to the program entry point (which is the start of the code segment). The advantage is that the program can be started using RUN.

To generate a program with a BASIC SYS header, use

```
cl65 -o file.prg -u __EXEHDR__ -t c64 -C c64-asm.cfg source.s
```

Please note that in this case a changed start address doesn't make sense, since the program must be loaded to the BASIC start address.

5. Extras

5.1 80 Columns conio driver

The C64 package comes with an alternative software driven 80 columns module `c64-soft80.o` which uses the memory under I/O between $D000 and $FF3F.

In memory constrained situations the memory from $400 to $7FF can be made available to a program by calling `_heapadd ((void *) 0x0400, 0x0400);` at the beginning of `main()`. Doing so is beneficial even if the program doesn't use the heap explicitly because loading a driver uses the heap implicitly.

Using `c64-soft80.o` is as simple as placing it on the linker command line like this:

```
cl65 -t c64 myprog.c c64-soft80.o
```

Note that the soft80 conio driver is incompatible with the `c64-ram.emd (c64_ram_emd)` extended memory driver and the `c64-hi.tgi (c64_hi_tgi)` graphics driver.

80 Columns conio driver (monochrome)

In an (even more) memory constrained situation, a size optimized version of the software driven 80 columns module may be used, which only supports one common text color for the whole screen.

```
cl65 -t c64 myprog.c c64-soft80mono.o
```

6. Platform-specific header files

Programs containing C64-specific code may use the `c64.h` or `cbm.h` header files. Using the later may be an option when writing code for more than one CBM platform, since it includes `c64.h` and declares several functions common to all CBM platforms.

6.1 C64-specific functions

The functions listed below are special for the C64. See the function reference for declaration and usage.

- get_ostype

6.2 C64-specific accelerator functions

The functions listed below are accelerator functions for the C64. See the function reference for declaration and usage.

- detect_c128
- detect_c64dtv
- detect_c65
- detect_chameleon
- detect_scpu
- detect_turbomaster
- get_c128_speed
- get_c64dtv_speed
- get_c65_speed
- get_chameleon_speed
- get_scpu_speed
- get_turbomaster_speed
- set_c128_speed
- set_c64dtv_speed
- set_c65_speed
- set_chameleon_speed
- set_scpu_speed
- set_turbomaster_speed

6.3 CBM-specific functions

Some functions are available for all (or at least most) of the Commodore machines. See the function reference for declaration and usage.

- cbm_close
- cbm_closedir
- cbm_k_setlfs
- cbm_k_setnam
- cbm_k_load
- cbm_k_save
- cbm_k_open

- cbm_k_close
- cbm_k_readst
- cbm_k_chkin
- cbm_k_ckout
- cbm_k_basin
- cbm_k_bsout
- cbm_k_clrch
- cbm_k_tksa
- cbm_k_second
- cbm_load
- cbm_open
- cbm_opendir
- cbm_read
- cbm_readdir
- cbm_save
- cbm_write
- get_tv
- waitvsync

6.4 Hardware access

The following pseudo variables declared in the `c64.h` header file do allow access to hardware located in the address space. Some variables are structures, accessing the struct fields will access the chip registers.

VIC

The `VIC` structure allows access to the VIC II (the graphics controller). See the `_vic2.h` header file located in the include directory for the declaration of the structure.

SID

The `SID` structure allows access to the SID (the sound interface device). See the `_sid.h` header file located in the include directory for the declaration of the structure.

CIA1, CIA2

Access to the two CIA (complex interface adapter) chips is available via the `CIA1` and `CIA2` variables. The structure behind these variables is explained in `_6526.h`.

COLOR_RAM

A character array that mirrors the color RAM of the C64 at $D800.

7. Loadable drivers

The names in the parentheses denote the symbols to be used for static linking of the drivers.

7.1 Graphics drivers

Note: All available graphics drivers for the TGI interface will use the space below the I/O area and Kernal ROM; so, you can have hires graphics in the standard setup without any memory loss or need for a changed configuration.

You can use a mouse driver at the same time that you use a TGI driver. But, if you want to see the default mouse pointer on the graphics screen, then you explicitly must link a special object file into your program. It will put the arrow into the "high RAM" area where the bitmaps are put. Its name is "`c64-tgimousedata.o`". Example:

```
cl65 -t c64 -o program-file main-code.c subroutines.s c64-tgimousedata.o
```

`c64-hi.tgi (c64_hi_tgi)`

This driver features a resolution of 320*200 with two colors and an adjustable palette (that means that the two colors can be chosen out of a palette of the 16 C64 colors).

Note that the graphics drivers are incompatible with the `c64-ram.emd (c64_ram_emd)` extended memory driver and the `c64-soft80.o` software 80-columns conio driver.

7.2 Extended memory drivers

`c64-65816.emd (c64_65816_emd)`

Extended memory driver for 65816 (eg SCPU) based extra RAM. Written and contributed by Marco van den Heuvel.

`c64-c256k.emd (c64_c256k_emd)`

A driver for the C64 256K memory expansion. This driver offers 768 pages of 256 bytes each. Written and contributed by Marco van den Heuvel.

`c64-dqbb.emd (c64_dqbb_emd)`

A driver for the Double Quick Brown Box cartridge. This driver offers 64 pages of 256 bytes each. Written and contributed by Marco van den Heuvel.

`c64-georam.emd (c64_georam_emd)`

A driver for the Berkeley Softworks GeoRam cartridge. The driver will determine the available RAM from the connected cartridge. It supports 64KB up to 2048KB of RAM.

`c64-isepic.emd (c64_isepic_emd)`

A driver for the ISEPIC cartridge. This driver offers just 8 pages of 256 bytes each. Written and contributed by Marco van den Heuvel.

`c64-kerberos.emd (c64_kerberos_emd)`

A driver for the Kerberos MIDI Cartridge. The cartridge has 512 pages of 256 bytes for a total of 128KB.

`c64-ram.emd (c64_ram_emd)`

A driver for the hidden RAM below the I/O area and kernal ROM. Supports 47 256 byte pages. Please note that this driver is incompatible with any of the graphics drivers, or the soft80 conio driver!

`c64-ramcart.emd (c64_ramcart_emd)`

A driver for the RamCart 64/128 written and contributed by Maciej Witkowiak. Will test the hardware for the available RAM.

`c64-reu.emd (c64_reu_emd)`

A driver for the CBM REUs. The driver will test the connected REU to find out how much RAM is present.

`c64-vdc.emd (c64_vdc_emd)`

A driver for the VDC memory of the C128. Written and contributed by Maciej Witkowiak. Can be used if the program is running in C64 mode of the C128. Autodetects the amount of memory available (16 or 64K) and offers 64 or 256 pages of 256 bytes each.

`dtv-himem.emd (dtv_himem_emd)`

A driver for the C64 D2TV (the second or PAL version). This driver offers indeed 7680 pages of 256 bytes each.

7.3 Joystick drivers

The default drivers, `joy_stddrv (joy_static_stddrv)`, point to `c64-stdjoy.joy (c64_stdjoy_joy)`.

`c64-hitjoy.joy (c64_hitjoy_joy)`

Driver for the Digital Excess & Hitmen adapter contributed by Groepaz. See http://www.digitalexcess.de/downloads/productions.php on instructions how to build one. Up to four joysticks are supported.

`c64-ptvjoy.joy (c64_ptvjoy_joy)`

Driver for the Protovision 4-player adapter contributed by Groepaz. See Protovision shop for prices and building instructions. Up to four joysticks are supported.

c64-stdjoy.joy (c64_stdjoy_joy)

Supports up to two standard joysticks connected to the joysticks port of the C64.

c64-numpad.joy (c64_numpad_joy)

Supports one joystick emulated by the numberpad of the C128 in C64 mode, the firebutton is labeled "5" and ENTER.

7.4 Mouse drivers

You can use these drivers in text-mode or graphics-mode (TGI) programs. See the description of the graphics drivers.

The default drivers, `mouse_stddrv (mouse_static_stddrv)`, point to `c64-1351.mou (c64_1351_mou)`.

c64-1351.mou (c64_1351_mou)

Supports a standard mouse connected to port #0 of the C64.

c64-inkwell.mou (c64_inkwell_mou)

Supports the Inkwell Systems lightpens, connected to port #0 of the C64. It can read both the one-button 170-C and the two-button 184-C pens. (It can read other lightpens and light-guns that send their button signal to the joystick left-button pin or the paddle Y [up/down] pin.)

c64-joy.mou (c64_joy_mou)

Supports a mouse emulated by a standard joystick, e.g. 1350 mouse, in port #1 of the C64.

c64-pot.mou (c64_pot_mou)

Supports a potentiometer device, e.g. Koala Pad, connected to port #1 of the C64.

7.5 RS232 device drivers

c64-swlink.ser (c64_swlink_ser)

Driver for the SwiftLink cartridge. Supports up to 38400 baud, requires hardware flow control (RTS/CTS) and does interrupt driven receives. Note that, because of the peculiarities of the 6551 chip, together with the use of the NMI, transmits are not interrupt driven; and, the transceiver blocks if the receiver asserts flow control because of a full buffer.

8. Limitations

8.1 Realtime clock

The realtime clock functions use the CIA1 TOD clock. As that clock only stores the time but not the date, the date set by `clock_settime()` is simply stored inside the C library for retrieval in the same program via `clock_gettime()`.

9. Other hints

9.1 Escape code

For an Esc, press CTRL and the `[` key.

9.2 Passing arguments to the program

Command-line arguments can be passed to `main()`. Since this is not supported directly by BASIC, the following syntax was chosen:

```
RUN:REM ARG1 " ARG2 IS QUOTED" ARG3 "" ARG5
```

1. Arguments are separated by spaces.
2. Arguments may be quoted.
3. Leading and trailing spaces around an argument are ignored. Spaces within a quoted argument are allowed.
4. The first argument passed to `main()` is the program name.
5. A maximum number of 10 arguments (including the program name) are supported.

9.3 Program return code

The program return code (low byte) is passed back to BASIC by use of the `ST` variable.

9.4 Interrupts

The runtime for the C64 uses routines marked as `.INTERRUPTOR` for interrupt handlers. Such routines must be written as simple machine language subroutines

and will be called automatically by the interrupt handler code when they are linked into a program. See the discussion of the `.CONDES` feature in the assembler manual.

10. License

This software is provided 'as-is', without any expressed or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.