Commodore Plus/4 specific information for cc65

Ullrich von Bassewitz

An overview over the Plus/4 runtime system as it is implemented for the cc65 C compiler.

1. Overview

This file contains an overview of the Plus/4 runtime system as it comes with the cc65 C compiler. It describes the memory layout, Plus/4 specific header files, available drivers, and any pitfalls specific to that platform.

Please note that Plus/4 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.

Since the Plus/4 and the Commodore 16/116 are almost identical (the latter are missing the 6551 ACIA and do only have 16KB of memory), the C16 documentation is also worth a look. The difference between both cc65 targets is that the Plus/4 runtime uses banking to support full 64K RAM, while the C16 does not use banking and supports up to 32K RAM. Because banking is not needed, most C16 programs will be somewhat smaller than the same program compiled for the Plus/4. However, programs compiled for the C16 will always run on the Plus/4, while the reverse is not necessarily true.

2. Binary format

The standard binary output format generated by the linker for the Plus/4 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 kernal and basic banked out. This gives a usable memory range of \$1000 - \$FD00. Having the kernal and basic ROMs banked out means, that no ROM entry points may be called directly from user code.

Special locations:

Text screen

The text screen is located at \$C00 (as in the standard setup).

Color RAM

The color RAM is located at \$800 (standard location).

Stack

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

Heap

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

4. Platform specific header files

Programs containing Plus/4 specific code may use the `plus4.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 `plus4.h` and declares several functions common to all CBM platforms.

Please note that most of the header file declarations from the `plus4.h` header file are shared between the C16 and Plus/4 configurations. For this reason, most of it is located in a common header file named `cbm264.h`.

4.1 Plus/4 specific functions

There are currently no special Plus/4 functions.

4.2 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

4.3 CBM specific CPU functions

Some CPU related functions are available for some of the Commodore machines. See the function reference for declaration and usage.

- fast
- slow
- isfast

4.4 Hardware access

The following pseudo variables declared in the `plus4.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.

`TED`

The `TED` structure allows access to the TED chip. See the `_ted.h` header file located in the include directory for the declaration of the structure.

`COLOR_RAM`

A character array that mirrors the color RAM of the Plus/4 at $0800.

5. Loadable drivers

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

5.1 Graphics drivers

No graphics drivers are currently available for the Plus/4.

5.2 Extended memory drivers

No extended memory drivers are currently available for the Plus/4.

5.3 Joystick drivers

`plus4-stdjoy.joy (plus4_stdjoy_joy)`

Supports up to two joysticks connected to the standard joysticks port of the Plus/4.

5.4 Mouse drivers

No mouse drivers are currently available for the Plus/4.

5.5 RS232 device drivers

`plus4-stdser.ser (plus4_stdser_ser)`

Driver for the 6551 ACIA chip built into the Plus/4. Supports up to 19200 baud, requires hardware flow control (RTS/CTS) and does interrupt driven receives. Note that because of the peculiarities of the 6551 chip transmits are not interrupt driven, and the transceiver blocks if the receiver asserts flow control because of a full buffer.

You need an adapter to use the builtin port, since the output levels available at the user port don't follow the RS232 standard.

6. Limitations

7. Other hints

7.1 Passing arguments to the program

Command line arguments can be passed to `main()`. Since this is not supported 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.

7.2 Program return code

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

7.3 Interrupts

The runtime for the Plus/4 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.

8. 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.