# MEGA 65

## DEVELOPER'S GUIDE

MEGA

MUSEUM OF ELECTRONIC GAMES & ART

# MEGA65 TEAM

**Dr. Paul Gardner-Stephen**
*(highlander)*
Founder
Software and virtual hardware architect
Spokesman and lead scientist

**Martin Streit**
*(seriously)*
Video and photo production
Tax and organization
Social media

**Dan Sanderson**
*(dddaaannn)*
Media and documentation
MEGA65.ROM

**Dr. Edilbert Kirk**
*(Bit Shifter)*
MEGA65.ROM
Manual and tools

**Gábor Lénárt**
*(LGB)*
Emulator

**Farai Aschwanden**
*(Tayger)*
Filehost and tools
Financial advisory

**Falk Rehwagen**
*(bluewaysw)*
GEOS

**Robert Steffens**
*(kibo)*
Network technology
Core bug hunting

**Detlef Hastik**
*(deft)*
Co-founder
General manager
Marketing and sales

**Oliver Graf**
*(lydon)*
Release management
VHDL and platform enhancements

**Antti Lukats**
*(antti-brain)*
Host hardware design and production

**Dieter Penner**
*(doubleflash)*
Host hardware support

**Mirko H.**
*(sy2002)*
Additional platforms and consulting

**Gürçe Işıkyıldız**
*(gurce)*
Tools and enhancements

**Daniel England**
*(Mew Pokémon)*
Additional code and tools

**Hernán Di Pietro**
*(indiocolifa)*
Additional emulation and tools

**Roman Standzikowski**
*(FeralChild)*
Open ROMs

**Anton Schneider-Michallek**
*(adtbm)*
Presentation and support

# Reporting Errors and Omissions

This book is being continuously refined and improved upon by the MEGA65 community. The version of this edition is:

```
commit 4e8e6c54bbda5a8817d0cf77d781ce67c3cf367c
date: Tue Jan 16 11:06:08 2024 -0800
```

We want this book to be the best that it possibly can. So if you see any errors, find anything that is missing, or would like more information, please report them using the MEGA65 User's Guide issue tracker:

https://github.com/mega65/mega65-user-guide/issues

You can also check there to see if anyone else has reported a similar problem, while you wait for this book to be updated.

Finally, you can always download the latest versions of our suite of books from these locations:

- https://mega65.org/mega65-book
- https://mega65.org/user-guide
- https://mega65.org/developer-guide
- https://mega65.org/basic65-ref
- https://mega65.org/chipset-ref
- https://mega65.org/docs

# Developing for the MEGA65

Published by
the MEGA Museum of Electronic Games & Art e.V., Germany.

WORK IN PROGRESS

Copyright ©2019 - 2024 by Paul Gardner-Stephen, the MEGA Museum of Electronic Games & Art e.V., and contributors.

This reference guide is made available under the GNU Free Documentation License v1.3, or later, if desired. This means that you are free to modify, reproduce and redistribute this reference guide, subject to certain conditions. The full text of the GNU Free Documentation License v1.3 can be found at https://www.gnu.org/licenses/fdl-1.3.en.html.

Implicit in this copyright license, is the permission to duplicate and/or redistribute this document in whole or in part for use in education environments. We want to support the education of future generations, so if you have any worries or concerns, please contact us.

January 16, 2024

# Contents

**1 Introduction** **xi**

Welcome to the MEGA65! . . . . . xiii

Other Books in this series . . . . . xiv

Come Join Us! . . . . . xiv

**I INTRODUCTION** **1**

**2 Welcome, and thank you!** **3**

**3 Other Resources** **7**

**II CROSS-PLATFORM DEVELOPMENT TOOLS** **11**

**4 Emulators** **13**

Using The Xmega65 Emulator . . . . . 15

The keyboard . . . . . 15

Apple MacOS start problems . . . . . 16

Updating settings . . . . . 17

Using the Live ISO image . . . . . 17

Creating a Bootable USB stick or DVD . . . . . 17

Getting Started . . . . . 18
Other Features of the Live ISO . . . . . 19

**5 Data Transfer and Debugging Tools 21**
m65 command line tool . . . . . 23
Screenshots using m65 tool . . . . . 23
Load and run a program on the MEGA65 . . . . . 24
Reconfigure the FPGA to run a different bitstream . . . . . 24
Remote keyboard entry . . . . . 24
Unit testing and logging support . . . . . 25
Using unit tests with C . . . . . 25
Using unit tests with BASIC 65 . . . . . 26
BASIC 65 example . . . . . 27
M65Connect . . . . . 28
mega65_ftp . . . . . 29
TFTP Server . . . . . 30
Converting a BASIC text file listing into a PRG file . . . . . 30

**6 Assemblers 33**

**7 C and C-Like Compilers 37**
MEGA65 libc . . . . . 39

**8 MEGA65 Standard C Library 41**
Structure and Usage . . . . . 43
conio.h . . . . . 43
conionit . . . . . 43
setscreenaddr . . . . . 43
getscreenaddr . . . . . 44
setcolramoffset . . . . . 44

getcolramoffset . . . . . 44
setcharsetaddr . . . . . 44
getcharsetaddr . . . . . 45
clrscr . . . . . 45
getscreensize . . . . . 45
setscreensize . . . . . 45
set16bitcharmode . . . . . 46
sethotregs . . . . . 46
setextendedattrib . . . . . 46
togglecase . . . . . 46
togglecase . . . . . 46
togglecase . . . . . 47
bordercolor . . . . . 47
bgcolor . . . . . 47
textcolor . . . . . 47
revers . . . . . 47
highlight . . . . . 48
blink . . . . . 48
underline . . . . . 48
altpal . . . . . 48
clearattr . . . . . 49
cellcolor . . . . . 49
setpalbank . . . . . 49
setpalbanka . . . . . 49
getpalbank . . . . . 50
getpalbanka . . . . . 50
setmapedpal . . . . . 50
getmapedpal . . . . . 50

setpalentry . . . . . 50
fillrect . . . . . 51
box . . . . . 51
hline . . . . . 52
vline . . . . . 52
gohome . . . . . 52
gotoxy . . . . . 52
gotox . . . . . 53
gotoy . . . . . 53
moveup . . . . . 53
movedown . . . . . 54
moveleft . . . . . 54
moveright . . . . . 54
wherex . . . . . 54
wherey . . . . . 55
pcputc . . . . . 55
pcputsxy . . . . . 55
cputcxy . . . . . 55
pcputs . . . . . 56
cputc . . . . . 56
cputnc . . . . . 56
cputhex . . . . . 56
cputdec . . . . . 57
cputs . . . . . 57
cputsxy . . . . . 57
cputcxy . . . . . 57
cputncxy . . . . . 58
cprintf . . . . . 58

pcprintf . . . . . 59
cgetc . . . . . 59
kbhit . . . . . 59
getkeymodstate . . . . . 60
flushkeybuf . . . . . 60
cinput . . . . . 60
VIC_BASE . . . . . 61

**9 BASIC Tokenisers** **63**

**III NATIVE DEVELOPMENT TOOLS** **67**

**10 C65 Memory Monitor** **69**

**11 MEGA65 Matrix Mode** **71**

**12 Turbo Assembler** **73**

**13 Dealing with C65 and C64-mode** **75**

**IV MEGA65 Operating System** **79**

**14 Developing System Programmes** **81**

Introduction . . . . . 85
Flash Menu . . . . . 85
Format/FDISK Utility . . . . . 86
Keyboard Test Utility . . . . . 87
MEGA65 Configuration Utility . . . . . 87
Freeze Menu . . . . . 87
Freeze Menu Helper Programmes . . . . . 87
Hypervisor . . . . . 88

OpenROM . . . . . 88

**15 MEGA65 Hyppo Services** **89**

Introduction . . . . . 91

General Services . . . . . 95

Drive/Storage Services . . . . . 99

Disk Image Services . . . . . 128

Task and Process Services . . . . . 132

System Partition Services . . . . . 148

Freezer Services . . . . . 149

**V MULTI-MEDIA AND DATA CONVERSION** **151**

**VI GEOS Development** **153**

**VII HARDWARE** **155**

**16 Using Nexys4 boards as a MEGA65** **157**

Building your own MEGA65 Compatible Computer . . . . . 161

Working Nexys4 Boards . . . . . 162

The Nexys4 board . . . . . 162

The Nexys4DDR board . . . . . 162

The Nexys A7 . . . . . 163

Power, Jumpers, Switches and Buttons . . . . . 164

Micro-USB Power . . . . . 165

External Power Supply . . . . . 165

Other Jumpers and Switches . . . . . 166

Connections and Peripherals . . . . . 167

Communicating with your PC . . . . . 167

Onboard buttons . . . . . 168
Keyboard . . . . . 169
Some key mappings with a USB keyboard . . . . . 170
Preparing microSDHC card . . . . . 171
Preparation Steps . . . . . 171
Loading the bitstream from QSPI . . . . . 173
Preparation Steps . . . . . 173
Widget Board . . . . . 173
PMOD-to-Joystick Adapter . . . . . 176
Useful Tips . . . . . 177

**VIII APPENDICES 179**

**A Flashing the FPGAs and CPLDs in the MEGA65 181**
Suggested PC specifications . . . . . 185
Warning . . . . . 186
Installing Vivado . . . . . 186
Installing the FTDI drivers . . . . . 202
Linux drivers . . . . . 203
Windows drivers . . . . . 203
Flashing the main FPGA using Vivado . . . . . 207
Flashing the CPLD in the MEGA65's Keyboard with Lattice Diamond . . . . . 220
Flashing the MAX10 FPGA on the MEGA65's Mainboard with INTEL QUARTUS 227

**B Trouble shooting 233**
Hardware . . . . . 235
No Digital Video Output . . . . . 235
MEGA65 R4, R5, R6 . . . . . 235
No lights when powering on . . . . . 235

Vivado . . . . . 235
RAM requirements . . . . . 236
mega65_ftp . . . . . 237
Missing Library . . . . . 237

**C Schematics** **239**
MEGA65 R3 Schematics . . . . . 241
MEGA65 R2 Schematics . . . . . 266
Nexys Widget Board Schematics . . . . . 289

**D Supporters & Donors** **293**
Organisations . . . . . 295
Contributors . . . . . 296
Supporters . . . . . 297

**INDEX** **307**

# CHAPTER 1

# Introduction

- Welcome to the MEGA65!
- Other Books in this series
- Come Join Us!

# WELCOME TO THE MEGA65!

Congratulations on your purchase of one of the most long-awaited computers in the history of computing! The MEGA65 is community designed, and based on the never-released Commodore® 65[1] computer; a computer designed in 1989 and intended for public release in 1990. Decades have passed, and we have endeavoured to invoke memories of an earlier time when computers were simple and friendly. They were not only simple to operate and understand, but friendly and approachable for new users.

These 1980s computers inspired many of their owners to pursue the exciting and rewarding technology careers they have today. Just imagine the exhilaration these early computing pioneers experienced, as they learned they could use their new computer to solve problems, write a letter, prepare taxes, invent new things, discover how the universe works, and perhaps even play an exciting game or two! We want to re-awaken that same level of excitement (which alas, is no longer found in modern computing), so we have created the **MEGA65**.

The MEGA65 team believes that owning a computer is like owning a home. You don't just use a home; you change things, big and small, to make it your own custom living space. After a while, when you settle in, you may decide to renovate or expand your home to make it more comfortable, or provide more utility. Think of the MEGA65 as your very own "computing home".

This guide will teach you how to do more than just hang pictures on a wall; it will show you how to build your dream home. While you read this user's guide, you will learn how to operate the MEGA65, write programs, add additional software, and extend hardware capabilities. What won't be immediately obvious is that along the journey, you will also learn about the history of computing as you explore the many facets of BASIC version 65 and operating system commands.

Computer graphics and music make computing more fun, and we designed the MEGA65 to be fun! In this user's guide, you will learn how to write programs using the MEGA65's built-in **graphics** and **sound** capabilities. But you don't need to be a programmer to have fun with the MEGA65. Because the MEGA65 includes a complete Commodore® 64™[2], it can also run thousands of existing games, utilities, and business software packages, as well as new programs being written today by Commodore computer enthusiasts. Excitement for the MEGA65 will grow as we all witness the programming marvels our MEGA65 community create, as they (and you!) discover and master the powerful capabilities of this modern Commodore computer recreation. Together, we can build a new "homebrew" community, teeming with software

[1] Commodore is a trademark of C= Holdings
[2] Commodore 64 is a trademark of C= Holdings

and projects that push the MEGA65's capabilities far beyond what anyone thought would be possible.

We welcome you on this journey! Thank you for becoming a part of the MEGA65 community of users, programmers, and enthusiasts!

# OTHER BOOKS IN THIS SERIES

This book is one of several within the MEGA65 documentation suite. The series includes:

- **The MEGA65 User's Guide**
  Provides an introduction to the MEGA65, and a condensed BASIC 65 command reference
- **The MEGA65 BASIC 65 Reference**
  Comprehensive documentation of all BASIC 65 commands, functions and operators
- **The MEGA65 Chipset Reference**
  Detailed documentation about the MEGA65 and C65's custom chips
- **The MEGA65 Developer's Guide**
  Information for developers who wish to write programs for the MEGA65
- **The MEGA65 Complete Compendium**
  (Also known as **The MEGA65 Book**)
  All volumes in a single huge PDF for easy searching. 1200 pages and growing!

# COME JOIN US!

Get involved, learn more about your MEGA65, and join us online at:

- https://mega65.org/chat
- https://mega65.org/forum

# PART I

## INTRODUCTION

CHAPTER 2

# Welcome, and thank you!

I would like to begin by thanking you for your interest in the MEGA65, and in developing software to run on it. The success of the MEGA65 as a fun retro-computing platform depends on there being enough interesting software for people to run on it. This requires the generous work of developers like yourself. So, again, thank you!

But developing for the MEGA65 is not merely a case of giving to the benefit of others. We have tried hard to create a machine that is as fun to program as the Commodore 64, but with some of the barriers and frustrations worn down somewhat. That is, it is our intention that people develop on the MEGA65 because it is fun, and then enjoy the process. That is, the journey should very much be the destination.

Now, in saying that, the MEGA65 is quite new, and we have been flat out creating the machine. This means that some development tools are not as ready or as simple or integrated as we would like. Improving the tools is also an area where we hope that many of you will enjoy contributing, so that developing for the MEGA65 can become accessible to as many people as possible.

If you encounter barriers to your journey of developing for the MEGA65, please do let us know, and consider creating solutions that can be shared with others. This will help everyone to enjoy being an active part of the MEGA65 ecosystem.

Finally, my apologies to anyone who has developed tools or utilities for the MEGA65 that have escaped my memory as I have written this document. If this is the case, please drop me an email, and I'll update this document to include them.

# CHAPTER 3

# Other Resources

This book focusses on the process of developing for the MEGA65. To avoid it being a thousand pages long, it doesn't describe the inner workings of the MEGA65, except where you need to know about them, so that you can engage in the development process. For example, if you wish to know about the capabilities of the various custom chips in the MEGA65, and which registers do which things, please refer to *the MEGA65 Book* online at `https://github.com/mega65/mega65-user-guide`, or consider purchasing one or more of the reference volumes for the MEGA65, from which the MEGA65 Book is formed. For example, the *MEGA65 Chipset Reference Guide* or the *MEGA65 Assembly Language Reference Guide*.

We also hope that you, the developer community, will contribute to those books, or write your own material that can help others in their journies with developing for the MEGA65.

# PART II

## CROSS-PLATFORM DEVELOPMENT TOOLS

# CHAPTER 4

# Emulators

- Using The Xmega65 Emulator
- Using the Live ISO image

At the time of writing, there is only one emulator for the MEGA65, **`xmega65`**; LGB's Xemu emulator suite. The LGB developers work hard to keep up with the development of the MEGA65; however, some MEGA65 emulation may not be accurate but should be sufficient for software development on the MEGA65.

During development, frequently test software on real hardware, such as a MEGA65 or FPGA board capable of running a MEGA65 core.

Download the MEGA65 emulator source code from `https://github.com/lgblgblgb/xemu`.

Download pre-compiled versions from `https://github.lgb.hu/xemu/`. Installers are available for macOS, Windows and Linux.

A live ISO image containing the emulator, documentation, and other tools is available from Forum64.de at `https://www.forum64.de/index.php?thread/104698-xemu-live-system-iso-file/&postID=1549927#post1549936`. Should you choose to use it you may find instructions on page 17.

# USING THE XMEGA65 EMULATOR

## The keyboard

The MEGA65 keyboard layout is based on the Commodore 65, which is unlike a modern PC keyboard. In most cases, the Xmega65 emulator recognizes PC keys as their equivalent MEGA65 keys, such as letters, numbers, and common modifiers like **Shift**. Xmega65 assumes an American/U.S. layout.

Some keys are only present on the MEGA65 keyboard and have no PC equivalents. Xmega65 uses the PC keys **Insert**, **Delete**, **Home**, **End**, **Page Up**, and **Page Down** to represent **+**, **£**, **CLR HOME**, **RUN STOP**, **HELP**, and **RESTORE**, respectively.

On most PC keyboards, this places RUN STOP and RESTORE next to each other, to make that key combination easy to enter.

This diagram shows the PETSCII glyphs as they appear on a MEGA65 keyboard, with the glyph on the left and the Shift glyph on the right. Note that on a PC keyboard, the key for is to the right of the Shift.

Xemu lets you have two joysticks (one at a time) on the numeric keypad. Enter over there is used to toggle ports 1, 2 and to off when pressed repeatedly.

Please be aware the emulator by default catches F9 F10 F11 as shortcuts of its own functions. If you want to use these keys with the emulated MEGA65, you must configure the keyboard mapping. The files are found in the folder for Xemu's settings as explained on page 17. This is done by copying **keymap-default.cfg** into **keymap.cfg** and then removing the following lines:

```
XEMU-EXIT F9
XEMU-FULLSCREEN F11
```

and changing:

```
F9 Unknown
F11 Unknown
```

into:

```
F9 F9
F11 F11
```

With these keys remapped, the full-screen and exit functions are still available from the menu. Right-click (or control-click) anywhere in the window to access it.

## Apple MacOS start problems

If a message **macOS cannot verify that this app is free from malware** pops up use the following procedure when you open the app for the first time:

- Open Finder on your Mac.
- Control-click (or right-click) on the Xmega65 app icon.
- In the context menu, select **Open**.

- The message appears again, but this time there is an **Open** button. Click **Open**. You only need to perform this procedure the first time you run the app. From now on, you can open the app normally.

### Updating settings

You may wish to edit the Xemu settings file to change the keyboard mapping or make a backup of the SD card image file. To access these files from within Xemu, open the menu, select **Debug / Advanced** then **Browse system folder**. Edit the **keymap-default.cfg** and **mega65-default.cfg** files with a text editor. For specific instructions on how to change **keymap-default.cfg** please refer to page 15.

The Xemu system folder location varies by operating system:

- Windows (may be copied into the address bar of the file explorer): **%appdata%\xemu-lgb\mega65**
- Linux: **~/.xemu-lgb/**
- Macintosh: **~/Library/Application\ Support/xemu-lgb/mega65/**

# USING THE LIVE ISO IMAGE

The Live ISO image is the product of a volunteer community; not the MEGA65 team. We include it for your convenience.

### Creating a Bootable USB stick or DVD

There are many ways to create a live ISO image. The method you choose depends on your operating system and whether you wish to install to a USB drive or burn it to

a DVD. Burning to a DVD is straightforward, assuming you own a computer that has a DVD writer. If you wish to create a faster bootable USB drive, try one of the methods below:

If you are using Windows, consider a tool like http://www.isotousb.com/.

On Linux, you can use the instructions at https://fossbytes.com/create-bootable-usb-media-from-iso-ubuntu/.

For Apple Macs, consider these instructions at https://ubuntu.com/tutorials/create-a-usb-stick-on-macos#1-overview.

Similar instructions are available for other popular computers, such as Amigas (https://forum.hyperion-entertainment.com/viewtopic.php?t=3857), or Sun UltraSPARC workstations (https://forums.servethehome.com/index.php?threads/how-to-create-a-bootable-solaris-11-usb.1998/).

Finally, the popular, easy-to-use, and free cross-platform belanaEtcher is available at https://www.balena.io/etcher/.

## Getting Started

To avoid potential copyright issues, the bootable ISO image does not include proprietary ROMs for the MEGA65; such as legacy C65 ROMs. It does include an open-source replacement ROM from our OpenROMs project. This ROM will boot into a BASIC 2 environment that you can use to load and execute many C64 programs as shown in the image below:

If you wish to use a C65 ROM that includes BASIC 10, download the appropriate ROM file and place it on another USB stick named **MEGA65.ROM**. On start-up, the MEGA65 will ask if a ROM has been downloaded; as shown in the image below:

If the Live ISO cannot find a ROM, it will prompt you to download a ROM; as shown below:

## Other Features of the Live ISO

As the previous screen-shots show, the Live ISO provides various and convenient desktop shortcuts. On the left-hand side, there are shortcuts for launching the MEGA65 emulator and the C65 emulator so you can test that programs will run on both platforms. As previously mentioned, both emulators are a work in progress and may not be 100% compatiable.

Another link provides access to the MEGA65 Book. This all-in-one volume, of apporixmately 800 pages, contains the official MEGA65 documentation. The majority of this developer's guide is also present in the MEGA65 Book.

This ISO also includes documentation for the C65 Notepad; a program for the C65 and MEGA65 written by Snoopy (the developer of the Live ISO image). A "read me" file contains further information about the Live ISO.

Finally, on the right-hand side, there are links to download a C65 ROM and to update the MEGA65 Book to the latest version. This will ensure you don't need to create a new bootable image each time a frequent update is made to the MEGA65 Book.

To access all contents of the Live ISO image, use the file explorer.

CHAPTER 5

# Data Transfer and Debugging Tools

- m65 command line tool
- M65Connect
- mega65_ftp
- TFTP Server
- Converting a BASIC text file listing into a PRG file

The key to effective cross-platform development is having quick and easy means to deploy and test software on the MEGA65. This is especially true while the MEGA65 emulator continues to be developed. In fact, even once the MEGA65 emulator is complete, it is unlikely that it will be able to offer full compatibility at full speed, because the MEGA65 is much more demanding to emulate than the C64.

There are a variety of tools that can be used for data transfer and debugging. These typically function using either the MEGA65's serial monitor interface, or via the MEGA65's fast ethernet adapter. The serial monitor interface is available via the UART lines on the JB1 header.

If you do not have access to the serial monitor interface, there are tools being developed for the fast ethernet port that provide some, but not all, of the capabilities of the serial monitor interface. These will be documented as they become available. The remainder of this chapter focusses on methods that access the serial monitor interface.

You can either connect a 3.3V UART adapter to the appropriate lines, or more conveniently, connect a TE-0790-03 JTAG debug module onto this connector. This gives you a USB connection that can be used for injecting software, remote debugging and memory inspection, as well as activating or flashing bitstreams. With this connection, there are the following tools:

# M65 COMMAND LINE TOOL

The https://github.com/mega65/mega65-tools repository contains a number of tools, utilities and example programs. These tools are mainly for Linux but can be used on Windows with Cygwin. One of those is the **m65** command line tool. This is rather a swiss-army knife collection of utilities in one. Common useful functions include:

## Screenshots using m65 tool

To take a screenshot of the MEGA65 use:

```
m65 -S
```

This will create a file called **mega65-screeen-000000.png**, or if that file already exists, the first non-used number will be used in place of **000000**.

Note that this screenshot function works by having **m65** emulate the function of the VIC-IV. Thus while it produces excellent looking digital screenshots, it may not exactly match the real display of the MEGA65. At the time of writing it does not render sprites or bitplanes, only text and bitmap-based video modes.

## Load and run a program on the MEGA65

To load and run a program on the MEGA65, you can use a command like:

```
m65 -F -4 -r foo.prg
```

The **-F** option tells **m65** to reset the MEGA65 before loading the program.

The **-4** option tells **m65** to switch the MEGA65 to C64-mode before loading the program. If this is left off, then it will attempt to load the program in C65-mode.

The **-r** option tells **m65** to run the program immediately after loading.

Note that this command works using the normal BASIC LOAD command, and is thus limited to loading programs into the lower 64KB of RAM

## Reconfigure the FPGA to run a different bitstream

To try out a different MEGA65 bitstream, a command like the following can be used:

```
m65 -b bitstream.bit
```

This will cause the named bitstream to be sent to the FPGA. As the FPGA will be reconfigured by this action, and program currently running will not merely be stopped, but also main memory will be cleared. For models of the MEGA65 that are fitted with 8MB or 16MB of expansion memory, those expansion memories are implemented in external chips, and so the contents of them will not be erased.

For non-MEGA65 bitstreams (such as zxunomega65 and gbc4mega65), use the '-q' argument instead:

```
m65 -q bitstream.bit
```

## Remote keyboard entry

The MEGA65's keyboard interface logic supports the injection of synthetic key events using the registers $D615 - $D617. The **m65** utility uses this to allow remote typing on the MEGA65 in a way that is transparent to software. There are three ways to use this:

```
m65 -t sometext
```

This form types the supplied text, in this case *sometext*, but does not simulate pressing RETURN. If you wish to simulate the pressing of RETURN, use **-T** instead of **-t**, e.g.:

```
m65 -T list
```

This would cause the LIST command to be typed and executed.

Finally, it is possible to begin general remote keyboard control via:

```
m65 -t -
```

In this mode, any key pressed on the keyboard of the computer where **m65** is running will be relayed to the MEGA65. Note that not all special keys are supported, and that there is some latency, so using key repeat can cause unexpected results. But for general remote control, it is a very helpful facility.

## Unit testing and logging support

The **m65** tool includes support to facilitate remote unit testing directly on MEGA65 hardware. When *unit testing mode* is active, **m65** waits for the MEGA65 to send certain byte sequences over the serial interface which signal the current state (started, passed, failed) of a given test. Additionally, it is possible to send log messages from the MEGA65 to the host computer.

Unit testing mode is entered by calling **m65** with the **-u** flag. To run a remote BASIC program in C65-mode and simultaneously put **m65** into *unit testing mode*, the following command can be used:

```
    m65 -Fur attic-ram.prg -w tests.log
```

The **-F** and **-r** options tell **m65** to reset the MEGA65 before loading the program "attic-ram.prg" and then automatically run it. The **-u** option then tells **m65** go into *unit testing mode* instead of exiting after launching the program. The optional **-w** option makes **m65** append the test results to the file "test.log" (creating the file if it doesn't exist).

Please note that **m65** automatically exits from *unit testing mode* if no test state signals were received for over 10 seconds.

Support is provided for sending unit test signals to the host computer from C and BASIC 65 programs:

### Using unit tests with C

The MEGA65 libc contains support for unit testing via functions defined in **tests.h** and **tests.c**.

To signal the start of a test, include **tests.h** and use

```
unit_test_setup("testName",issueNumber);
```

where *"testName"* is a human-readable name of the test (e.g. "VIC-II") and *issueNumber* a reference to the corresponding bug issue (for example, the issue number from github).

After starting a test, it's possible to signal passed tests with the unit_test_ok() function:

```
unit_test_ok();
```

A failed test is signalled with unit_test_fail():

```
unit_test_fail("fail message");
```

Each time the unit_test_ok() or unit_test_fail() functions are called, the *sub issue* of the test (reported on the host computer) is incremented. This makes it easier to combine and identify multiple tests in one file.

You can send arbitrary log messages via unit_test_log():

```
unit_test_log("hello world from mega65!");
```

...and finally, when all is done, the end of unit testing is signalled by the use of

```
unit_test_done();
```

### Using unit tests with BASIC 65

**b65support.bin** is a machine language module providing support for unit testing from BASIC 65, available in the **bin65** folder of the mega65-tools repository. This module works by redirecting the USR vector to perform the functions needed to communicate with the testing host.

In an automated test scenario, you may want to inject the **b65support.bin** binary into MEGA65 RAM by using **m65**:

```
m65 -@ mega65-tools/bin65/b65support.bin@15fe
```

Of course it's also possible to load **b65support.bin** directly from the MEGA65 by mounting the **M65UTILS.D81** image from the freezer and issuing

```
BLOAD "B65SUPPORT.BIN"
```

After loading, **b65support.bin** is initialised with

```
SYS $1600
```

Once initialised, the following functions are provided by **b65support.bin**:

```
A=USR((issueNum))
```

prepares a new test with number <issueNum> and resets subissue number to 0

```
A=USR("=(testName)")
```

sets test name and sends test start signal; for example: **A=USR("=VIC-III")** sets the test name to 'VIC-III' and signals the host computer that the test has started.

```
A=USR("/(logMessage))
```

sends a log message to the host computer

```
A=USR("P")
```

sends the 'passed' signal to the host computer and increases the sub issue number

```
A=USR("F")
```

sends the 'test failed' signal to the host computer and increases the sub issue number

```
A=USR("D")
```

sends the 'test done' signal to the host computer

All calls return the current sub issue number or **?ILLEGAL QUANTITY ERROR** in case of calling an invalid command.

### BASIC 65 example

The following is a complete BASIC 65 example showing how to use **m65**'s unit testing features:

```
100 rem attic ram cache test
110 poke $bffff2,$e0          : rem enable attic ram cache
120 sys $1600                 : rem init test module
130 a=usr(379)                : rem set issue number
140 a=usr("=attic-ram-cache") : rem set test name
150 bank128:poke0,65          : rem just to be sure
160 b0=$8000000 : b1=$8000100 : rem attic ram areas to be tested
170 for r=0 to $ff
180   poke b0+r,0             : rem fill area 1 with 0
190   poke b1+r,$ff           : rem fill area 2 with $ff
200 next r
210 for t=1 to 10             : rem 10 tries
220   poke b0,32              : rem write to b0
230   for x=0 to $ff:t1=b1+x
240     a=peek(b0)            : rem read from b0
250     b=peek(t1):b=peek(t1) : rem read twice from t1
260     ifb()255 thenf=t:t=11:x=256   : rem this shouldn't happen
270   next x
280 next t
290 if f=0 then begin
300   print "no faults detected after";t;"tries."
310   a=usr("p")              : rem signal 'test passed' to host
320 bend : else begin
330   a=usr("f")              : rem signal 'test failed' to host
340   print "hyper ram fault detected after";f;"tries."
350   print "peek($";hex$(t1);") [t1] is";b;"but should be 255"
360 bend
370 a=usr("d")                : rem test done
```

# M65CONNECT

This is a cross-platform graphical tool available for Windows, Linux and MacOSX, which allows access to most of the functions of the **m65** command-line tool, without needing to use a command line, or being able to compile the tool for your preferred operating system.

The repository for M65Connect is: https://github.com/MEGA65/m65connect

The latest binary version is available from https://files.mega65.org.

With the MEGA65 or Nexys FPGA switched off, connect a USB cable from your computer to the MEGA65 or Nexys FPGA board. Run the *M65Connect* executable and follow the prompts to connect. The program will help you identify which USB Serial Port to communicate over.

With this tool you can easily transfer PRG programs and a variety of other files. M65Connect can handle the transfer, switching to C64-mode, and execution of programs.

# MEGA65_FTP

The **mega65_ftp** utility from the https://github.com/mega65/mega65-tools repository is a little misleadingly named: While it is a File Transfer Program, it does not use the File Transfer Protocol (FTP). Rather, it uses the serial monitor interface to take remote control of a MEGA65, and directly access its SD card to enable copying of files between the MEGA65 and the host computer.

Note that it does not perfectly restore the MEGA65's state on exit, and thus should only be used when the MEGA65 is at the READY prompt, so that any running software doesn't go haywire. In particular, you should avoid using it when a sensitive program is running, such as the Freeze Menu, MEGA65 Configuration Utility, or the MEGA65 Format/FDISK utility. (This problem could be solved with a little effort, if someone has the time and interest to fix it).

When run, it provides an FTP-like interface that supports the **get**, **put**, **rename** and **dir** commands. Note that when putting a file, you should make sure that it is given a name that is all capitals and has o DOS-compatible 8.3 character file name. This is due to limitations in both **mega65_ftp** and the MEGA65's Hypervisor's VFAT32 file system code. Again, these problems could be fixed with a modest amount of effort on the part of a motivated member of the community.

Finally, the **mega65_ftp** program is *very* slow to push new files to the MEGA65, typically yielding speeds of around 5KB/sec. This is partly because the serial monitor interface is capable of transferring data at only 40KB/sec (when set to 4,000,000 bits per second), and partly because the remote control process results in a lot of round-trips where helper routines are executed on the MEGA65 to read, write and verify sectors on the SD card. It would be quite feasible to improve this to reach close to 40KB/sec, and potentially faster using either some combination of data compression, de-duplication of identical sectors (especially when uploading disk images) and other techniques. Again, this would be a very welcome contribution that someone in the community could contribute to everyone's benefit.

# TFTP SERVER

Work on a true TFTP server for the MEGA65 that supports fast TFTP transfers over the 100mbit ethernet has begun, and can be used to very quickly read files from the MEGA65. Speeds of close to 1MB/sec are possible, depending on SD card performance. Rather than using DHCP, this utility will respond to *any* IP address that ends in .65. It always uses the MAC address 40:40:40:40:40:40. True DHCP support as well as using the MEGA65's configured ethernet MAC address may be added in the future.

More importantly, support for writing files to the SD card is not yet complete, and is blocked by the need for the implementation of the necessary functions in the MEGA65's Hypervisor for creating and growing files. A particular challenge is enabling the creation of files with contiguous clusters as is required for D81 disk images: If a D81 file is fragmented, then it cannot be mounted, because the mounting mechanism requires a pointer to the contiguous block of the SD card containing the disk image. In the interim, **`mega65_ftp`** can be used as a substitute.

# CONVERTING A BASIC TEXT FILE LISTING INTO A PRG FILE

If you have a untokenised BASIC program in plain text format sourced from somewhere like an internet post, and you wish to try it on the MEGA65 without typing it in, it is possible to convert it to a PRG.

*C64List* is a Windows-based command-line tool that will allow you to make the conversion. Once you have a *.PRG* file, you can use a tool like M65Connect to upload it to the MEGA65 or Nexys FPGA.

C64List is available for download from http://www.commodoreserver.com/Downloads.asp

Ensure you have a program listing saved to a file on your local computer (for example, *program.txt*) encoded as ANSI or UTF8.

Use C64List to convert the file to a PRG file using:

```
C64List program.txt -prg
```

Now you can upload your newly converted program to the MEGA65 with M65Connect or one of the other tools described previously.

It is worth noting that this method will not be 100% effective on listings with special PETSCII characters. Programs with PETSCII will require some editing on the MEGA65 itself before saving to disk.

# CHAPTER 6

# Assemblers

The table below shows an overview of assemblers known to work with MEGA65. For general use we recommend **ACME** as it has good support for the 45GS02 instruction set; is open source; and finally written in C. The latter means that it may be ported to run natively on the MEGA65 in the future.

| Name | 45GS02 | Source | Reference |
|---|---|---|---|
| ACME | yes | C | https://sourceforge.net/projects/acme-crossass |
| KickAss | yes | Java | gitlab.com/jespergravgaard/kickassembler65ce02 |
| LLVM-MOS | yes | C++ | https://llvm-mos.org/wiki/Welcome |
| Ophis | yes | Python | https://github.com/michaelcmartin/Ophis |
| BSA | yes | C | https://github.com/Edilbert/BSA |
| CA65 | no[1] | C | https://github.com/mega65/cc65 |

The **BSA** assembler is currently used to build the **MEGA65.ROM**. Most of this source code is written in the syntax of the ancient **BSO** assembler (Boston Systems Office), which was used in the years 1989 - 1991 by software developers, working on the C65. The **BSA** Assembler has a compatibility mode, which makes it possible to assemble these old source codes with minor or none modifications. The **BSA** Assembler has currently only a description of commands embedded in the C-source of the assembler.

The **LLVM-MOS** project comes with a GNU compatible macro assembler and disassembler that supports all mnemonics of the 45GS02. It can be used as a stand-alone tool with the command **llvm-mc**.

[1] Our fork of CA65 (part of CC65) correctly detects the MEGA65's CPU, but has no explicit support for the processor's features

# CHAPTER 7

# C and C-Like Compilers

- MEGA65 libc

Short answer: CC65 and KickC both work on the MEGA65.

Both CC65 and KickC are known to work on the MEGA65. However, both by default have only a C64 memory model, and use only 6502 opcodes. It would be super for someone to create a C65 memory configuration for CC65, and should not be too hard to do.

CC65 supports overlays, which could be powerfully used with the MEGA65's extra memory to allow programs larger than 64KB. However, this would require writing a suitable loader for such programs, which also does not yet exist.

Similarly, modifying the code generator of CC65 to use 45GS02 features would not be particularly difficult to do, and would help to overcome the otherwise horribly slow and bloated code that CC65 produces. Also adding first-class support for the 45GS02 CPU features in CA65 (or perhaps even better, making CC65 produce ACME compatible assembly output) would be of tremendous advantage, and not particularly hard to do. These would all be great tasks to tackle while you wait for your MEGA65 DevKit to arrive!

An example template for a C program that can be compiled using CC65 and executed on the MEGA65 can be found in the repository `https://github.com/MEGA65/hello-world`. This repository will even download and compile CC65, if you don't already have it installed on your system. This repository should work on Linux and Mac, and on Windows under the Windows Subsystem for Linux (WSL).

The LLVM-MOS project (`https://llvm-mos.org`) has a MEGA65 target that has been tested with C99, C++11, and Rust. The LLVM MOS compiler has support for 45GS02 instructions (`-mcpu=mos45gs02`) and comes with an assembler and disassembler. Precompiled binaries of the SDK are available for Linux, Windows, and macOS.

# MEGA65 LIBC

A C library is being developed for the MEGA65, and which already includes a number of useful features. This library is available from `http://github.com/mega65/mega65-libc`. The procedures, functions and definitions it provides are documented in a separate chapter.

The MEGA65 libc is currently available only for LLVM and CC65, although we would welcome someone maintaining a KickC port of it.

CHAPTER 8

# MEGA65 Standard C Library

- Structure and Usage
- conio.h

A C library is being developed for the MEGA65, and which already includes a number of useful features. This library is available from http://github.com/mega65/mega65-libc. The procedures, functions and definitions it provides are documented in a separate chapter.

# STRUCTURE AND USAGE

The MEGA65 libc is purposely provided in source-form only, and with groups of functions in separate files, and with separate header files for including. The idea is that you include only the header files that you require, and add only the source files required to the list of source files of the program you are compiling. This avoids the risk of the compiler including functions in your compiled program that are never used, and thus wasting precious memory space.

Note that some library source files are written in C, and thus are present as files with a `.c` extension, while others are written in assembly language either for efficiency or out of necessity, and have a `.s` extension.

Typical usage is to either have the mega65-libc source code checked out in an adjacent directory, or within the source directory of your own project. In the latter case, this can be done using the git submodule facility.

The following sections document each of the header files and the corresponding functions that they provide.

# CONIO.H

## conionit

**Description:** Initialises the library internal state

**Syntax:** `void conioinit(void)`

**Notes:** This must be called before using any conio library function.

## setscreenaddr

**Description:** Sets the screen RAM start address

**Syntax:** `void setscreenaddr(long addr);`

**Parameters:** **addr:** The address to set as start of screen RAM

**Notes:** No bounds check is performed on the selected address

# getscreenaddr

**Description:** Returns the screen RAM start address

**Syntax:** `long getscreenaddr(void);`

**Return Value:** The current screen RAM address start address.

# setcolramoffset

**Description:** Sets the color RAM start offset value

**Syntax:** `void setcolramoffset(long offset);`

**Parameters:** **addr:** The offset from the beginning of the color RAM address ($FF80000)

**Notes:** No bounds check is performed on the resulting address. Do not exceed the available Color RAM size

# getcolramoffset

**Description:** Returns the color RAM start offset value

**Syntax:** `long getscreenaddr(void);`

**Return Value:** The current color RAM start offset value.

# setcharsetaddr

**Description:** Sets the character set start address

**Syntax:** `void setcharsetaddr(long addr);`

**Parameters:** **addr:** The address to set as start of character set

**Notes:** No bounds check is performed on the selected address

# getcharsetaddr

**Description:** Returns the current character set start address

**Syntax:** `long getscreenaddr(void);`

**Return Value:** The current character set start address.

# clrscr

**Description:** Clear the text screen.

**Syntax:** `void clrscr(void)`

**Notes:** Color RAM will be cleared with current text color

# getscreensize

**Description:** Returns the dimensions of the text screen

**Syntax:** `void getscreensize(unsigned char* width, unsigned char* height)`

**Parameters:** **width:** Pointer to location where width will be returned

**height:** Pointer to location where height will be returned

# setscreensize

**Description:** Sets the dimensions of the text screen

**Syntax:** `void setscreensize(unsigned char width, unsigned char height)`

**Parameters:** **width:** The width in columns (40 or 80)

**height:** The height in rows (25 or 50)

**Notes:** Currently only 40/80 and 25/50 are accepted. Other values are ignored.

# set16bitcharmode

**Description:** Sets or clear the 16-bit character mode

**Syntax:** `void set16bitcharmode(unsigned char f)`

**Parameters:** **f:** Set true to set the 16-bit character mode

**Notes:** This will trigger a video parameter reset if HOTREG is ENABLED. See sethotregs function.

# sethotregs

**Description:** Sets or clear the hot-register behavior of the VIC-IV chip.

**Syntax:** `void set16bitcharmode(unsigned char f)`

**Parameters:** **f:** Set true to enable the hotreg behavior

**Notes:** When this mode is ENABLED a video mode reset will be triggered when touching $D011, $D016, $D018, $D031 or the VIC-II bank bits of $DD00.

# setextendedattrib

**Description:** Sets or clear the VIC-III extended attributes mode to support blink, underline, bold and highlight.

**Syntax:** `void setextendedattrib(unsigned char f)`

**Parameters:** **f:** Set true to set the extended attributes mode

# togglecase

**Description:** Set lower case character set

**Syntax:** `void setlowercase(void)`

# togglecase

**Description:** Set upper case character set

**Syntax:** `void setuppercase(void)`

# togglecase

**Description:** Toggle the current character set case

**Syntax:** `void togglecase(void)`

# bordercolor

**Description:** Sets the current border color

**Syntax:** `void bordercolor(unsigned char c)`

**Parameters:** **c:** The color to set

# bgcolor

**Description:** Sets the current screen (background) color

**Syntax:** `void bgcolor(unsigned char c)`

**Parameters:** **c:** The color to set

# textcolor

**Description:** Sets the current text color

**Syntax:** `void textcolor(unsigned char c)`

**Parameters:** **c:** The color to set

**Notes:** This function preserves attributes in the upper 4-bits if extended attributes are enabled. See setextendedattrib.

# revers

**Description:** Enable the reverse attribute

**Syntax:** `void revers(unsigned char c)`

**Parameters:** **enable:** 0 to disable, 1 to enable

**Notes:** Extended attributes mode must be active. See setextendedattrib.

# highlight

**Description:** Enable the highlight attribute

**Syntax:** `void highlight(unsigned char c)`

**Parameters:** **enable:** 0 to disable, 1 to enable

**Notes:** Extended attributes mode must be active. See setextendedattrib.

# blink

**Description:** Enable the blink attribute

**Syntax:** `void blink(unsigned char c)`

**Parameters:** **enable:** 0 to disable, 1 to enable

**Notes:** Extended attributes mode must be active. See setextendedattrib.

# underline

**Description:** Enable the underline attribute

**Syntax:** `void underline(unsigned char c)`

**Parameters:** **enable:** 0 to disable, 1 to enable

**Notes:** Extended attributes mode must be active. See setextendedattrib.

# altpal

**Description:** Enable the alternate-palette attribute

**Syntax:** `void altpal(unsigned char c)`

**Parameters:** **enable:** 0 to disable, 1 to enable

**Notes:** Extended attributes mode must be active. See setextendedattrib.

# clearattr

**Description:** Clear all text attributes

**Syntax:** `void clearattr())`

**Notes:** Extended attributes mode must be active. See setextendedattrib.

# cellcolor

**Description:** Sets the color of a character cell

**Syntax:** `void cellcolor(unsigned char x, unsigned char y, unsigned char c)`

**Parameters:** **x:** The cell X-coordinate

**y:** The cell Y-coordinate

**c:** The color to set

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# setpalbank

**Description:** Set current text/bitmap palette bank (BTPALSEL).

**Syntax:** `void setpalbank(unsigned char bank)`

**Parameters:** **bank:** The palette bank to set. Valid values are 0, 1, 2 or 3.

**Notes:** Use setpalbanka to set alternate text/bitmap palette

# setpalbanka

**Description:** Set alternate text/bitmap palette bank.

**Syntax:** `void setpalbanka(unsigned char bank)`

**Parameters:** **bank:** The palette bank to set. Valid values are 0, 1, 2 or 3.

**Notes:** Use setpalbank to set main text/bitmap palette

# getpalbank

**Description:** Get selected text/bitmap palette bank.

**Syntax:** `unsigned char getpalbank(void)`

**Notes:** Use getpalbanka to get alternate text/bitmap selected palette

**Return Value:** The current selected main text/bitmap palette bank.

# getpalbanka

**Description:** Get selected alternate text/bitmap palette bank.

**Syntax:** `unsigned char getpalbanka(void)`

**Notes:** Use getpalbank to get main text/bitmap selected palette

**Return Value:** The current selected alternate text/bitmap palette bank.

# setmapedpal

**Description:** Set maped-in palette bank at $D100-$D3FF.

**Syntax:** `void setmapedpal(unsigned char bank)`

**Parameters:** **bank:** The palette bank to map-in. Valid values are 0, 1, 2 or 3.

# getmapedpal

**Description:** Get maped-in palette bank at $D100-$D3FF.

**Syntax:** `unsigned char getmapedpal(void)`

# setpalentry

**Description:** Set color entry for the maped-in palette

**Syntax:** `void setpalentry(unsigned char c, unsigned char r, unsigned char g, unsigned char b)`

**Parameters:** **c:** The palette entry index (0-255)

**r:** The red component value

**g:** The green component value

**b:** The blue component value

**Notes:** Use setmapedmal to bank-in the palette to modify

# fillrect

**Description:** Fill a rectangular area with character and color value

**Syntax:** `void fillrect(const RECT *rc, unsigned char ch, unsigned char col)`

**Parameters:** **rc:** A RECT structure specifying the box coordinates

**ch:** A char code to fill the rectangle

**col:** The color to fill

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# box

**Description:** Draws a box with graphic characters

**Syntax:** `void box(const RECT *rc, unsigned char color, unsigned char style, unsigned char clear, unsigned char shadow)`

**Parameters:** **rc:** A RECT structure specifying the box coordinates

**color:** The color to use for the graphic characters

**style:** The style for the box borders. Can be set to BOX_STYLE_NONE, BOX_STYLE_ROUNDED, BOX_STYLE_INNER, BOX_STYLE_OUTER, BOX_STYLE_MID

**clear:** Set to 1 to clear the box interior with the selected color

**shadow:** Set to 1 to draw a drop shadow

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# hline

**Description:** Draws an horizontal line.

**Syntax:** `void hline(unsigned char x, unsigned char y, unsigned char len, unsigned char style)`

**Parameters:** **x:** The line start X-coordinate

**y:** The line start Y-coordinate

**len:** The line length

**style:** The style for the line. See HLINE_ constants for available styles.

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# vline

**Description:** Draws a vertical line.

**Syntax:** `void vline(unsigned char x, unsigned char y, unsigned char len, unsigned char style)`

**Parameters:** **x:** The line start X-coordinate

**y:** The line start Y-coordinate

**len:** The line length

**style:** The style for the line. See VLINE_ constants for available styles.

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# gohome

**Description:** Set the current position at home (0,0 coordinate)

**Syntax:** `void gohome(void)`

# gotoxy

**Description:** Set the current position at X,Y coordinates

**Syntax:** `void gotoxy(unsigned char x, unsigned char y)`

**Parameters:** **x:** The new X-coordinate

**y:** The new Y-coordinate

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# gotox

**Description:** Set the current position X-coordinate

**Syntax:** `void gotox(unsigned char x)`

**Parameters:** **x:** The new X-coordinate

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# gotoy

**Description:** Set the current position Y-coordinate

**Syntax:** `void gotoy(unsigned char y)`

**Parameters:** **y:** The new Y-coordinate

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# moveup

**Description:** Move current position up

**Syntax:** `void moveup(unsigned char count)`

**Parameters:** **count:** The number of positions to move

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# movedown

**Description:** Move current position down

**Syntax:** `void movedown(unsigned char count)`

**Parameters:** **count:** The number of positions to move

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# moveleft

**Description:** Move current position left

**Syntax:** `void moveleft(unsigned char count)`

**Parameters:** **count:** The number of positions to move

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# moveright

**Description:** Move current position right

**Syntax:** `void moveright(unsigned char count)`

**Parameters:** **count:** The number of positions to move

**Notes:** No screen bounds checks are performed; out of screen behavior is undefined

# wherex

**Description:** Return the current position X coordinate

**Syntax:** `unsigned char wherex(void)`

**Return Value:** The current position X coordinate

# wherey

**Description:** Return the current position Y coordinate

**Syntax:** `unsigned char wherey(void)`

**Return Value:** The current position Y coordinate

# pcputc

**Description:** Output a single petscii character to screen at current position

**Syntax:** `void cputc(unsigned char c)`

**Parameters:** **c:** The petscii character to output

# pcputsxy

**Description:** Output a petscii string at X,Y coordinates

**Syntax:** `void pcputsxy (unsigned char x, unsigned char y, const unsigned char* s)`

**Parameters:** **x:** The X coordinate where string will be printed

**y:** The Y coordinate where string will be printed

**s:** The petscii string to print

**Notes:** No pointer check is performed. If s is null or invalid, behavior is undefined

# cputcxy

**Description:** Output a single petscii character at X,Y coordinates

**Syntax:** `void pcputcxy (unsigned char x, unsigned char y, unsigned char c)`

**Parameters:** **x:** The X coordinate where character will be printed

**y:** The Y coordinate where character will be printed

**c:** The petscii character to print

# pcputs

**Description:** Output a petscii string at current position

**Syntax:** `void pcputs(const unsigned char* s)`

**Parameters:** **s:** The string to print

**Notes:** No pointer check is performed. If s is null or invalid, behavior is undefined

# cputc

**Description:** Output a single screen code character to screen at current position

**Syntax:** `void cputc(unsigned char c)`

**Parameters:** **c:** The screen code of the character to output

# cputnc

**Description:** Output N copies of a character at current position

**Syntax:** `void cputnc(unsigned char count, unsigned char c)`

**Parameters:** **c:** The screen code of the characters to output

**count:** The count of characters to print

# cputhex

**Description:** Output an hex-formatted number at current position

**Syntax:** `void cputhex(long n, unsigned char prec)`

**Parameters:** **n:** The number to write

**prec:** The precision of the hex number, in digits. Leading zeros will be printed accordingly

**Notes:** The $ symbol will be automatically added at beginning of string

# cputdec

**Description:** Output a decimal number at current position

**Syntax:** `void cputdec(long n, unsigned char padding, unsigned char leadingZ)`

**Parameters:** **n:** The number to write

**padding:** The padding space to add before number

**leadingZ:** The leading zeros to print

# cputs

**Description:** Output screen codes at current position

**Syntax:** `void cputs(const unsigned char* s)`

**Parameters:** **s:** Am array of screen codes to print

**Notes:** This function works with screen codes only. To output ordinary ASCII/PETSCII strings, use the "pcputs" macro. No pointer check is performed. If s is null or invalid, behavior is undefined.

# cputsxy

**Description:** Output multiple screen codes at X,Y coordinates

**Syntax:** `void cputsxy (unsigned char x, unsigned char y, const unsigned char* s)`

**Parameters:** **x:** The X coordinate where string will be printed

**y:** The Y coordinate where string will be printed

**s:** An array of screen codes to print

**Notes:** This function works with screen codes only. To output ordinary ASCII/PETSCII strings, use the "pcputsxy" macro. No pointer check is performed. If s is null or invalid, behavior is undefined.

# cputcxy

**Description:** Output a single character at X,Y coordinates

**Syntax:** `void cputcxy (unsigned char x, unsigned char y, unsigned char c)`

**Parameters:** **x:** The X coordinate where character will be printed

**y:** The Y coordinate where character will be printed

**c:** The screen code of the character to print

# cputncxy

**Description:** Output N copies of a single character at X,Y coordinates

**Syntax:** `void cputncxy (unsigned char x, unsigned char y, unsigned char count, unsigned char c)`

**Parameters:** **x:** The X coordinate where character will be printed

**y:** The Y coordinate where character will be printed

**count:** The number of characters to output

**c:** The screen code of the characters to print

# cprintf

**Description:** Prints formatted output.
Escape strings can be used to modify attributes, move cursor, etc similar to PRINT in CBM BASIC.

**Syntax:** `unsigned char cprintf (const unsigned char* format, ...)`

**Parameters:** **format:** The string to output. The available escape codes are:

**Cursor positioning**
\t Go to next tab position (multiple of 8s)
\r Carriage Return
\n New line

| | | | |
|---|---|---|---|
| `{clr}` | Clear screen | `{home}` | Move cursor to home (top-left) |
| `{d}` | Move cursor down | `{u}` | Move cursor up |
| `{r}` | Move cursor right | `{l}` | Move cursor left |

**Attributes**

| | | | |
|---|---|---|---|
| {rvson} | Reverse attribute ON | {rvsoff} | Reverse attribute OFF |
| {blon} | Blink attribute ON | {bloff} | Blink attribute OFF |
| {ulon} | Underline attribute ON | {uloff} | Underline attribute OFF |

**Colors (default palette)**

| | | | |
|---|---|---|---|
| {blk} | {wht} | {red} | {cyan} |
| {pur} | {grn} | {blu} | {yel} |
| {ora} | {brn} | {pink} | {gray1} |
| {gray2} | {lblu} | {lgrn} | {gray3} |

**Notes:** This function works with screen codes only! To output ordinary ASCII/PETSCII strings, use the "pcprintf" macro. Currently no argument replacement is done with the variable arguments.

# pcprintf

**Description:** Prints formatted petscii string output.

**Syntax:** `see cprintf`

# cgetc

**Description:** Waits until a character is in the keyboard buffer and returns it

**Syntax:** `unsigned char cgetc (void);`

**Return Value:** The last character in the keyboard buffer

**Notes:** Returned values are ASCII character codes

# kbhit

**Description:** Returns the character in the keyboard buffer

**Syntax:** `unsigned char kbhit (void);`

**Return Value:** The character code in the keyboard buffer, 0 otherwise.

**Notes:** Returned values are ASCII character codes

# getkeymodstate

**Description:** Return the key modifiers state.

**Syntax:** `unsigned char getkeymodstate(void)`

**Return Value:** A byte with the key modifier state bits, where bits:

| Bit | Meaning | Constant |
|---|---|---|
| 0 | Right SHIFT State | KEYMOD_RSHIFT |
| 1 | Left SHIFT state | KEYMOD_LSHIFT |
| 2 | CTRL state | KEYMOD_CTRL |
| 3 | MEGA state | KEYMOD_MEGA |
| 4 | ALT state | KEYMOD_ALT |
| 5 | NOSCRL state | KEYMOD_NOSCRL |
| 6 | CAPSLOCK state | KEYMOD_CAPSLOCK |
| 7 | Reserved | - |

# flushkeybuf

**Description:** Flush the keyboard buffer

**Syntax:** `void flushkeybuf(void)`

# cinput

**Description:** Get input from keyboard, printing incoming characters at current position.

**Syntax:** `unsigned char cinput(char* buffer, unsigned char buflen, unsigned char flags)`

**Parameters:** **buffer:** Target character buffer preallocated by caller

**buflen:** Target buffer length in characters, including the null character terminator

**flags:** Flags for input: (default is accept all printable characters)

CINPUT_ACCEPT_NUMERIC
Accepts numeric characters.

CINPUT_ACCEPT_LETTER
Accepts letters.

`CINPUT_ACCEPT_SYM`
Accepts symbols.

`CINPUT_ACCEPT_ALL`
Accepts all. Equals to `CINPUT_ACCEPT_NUMERIC |CINPUT_ACCEPT_LETTER |CINPUT_ACCEPT_SYM`

`CINPUT_ACCEPT_ALPHA`
Accepts alphanumeric characters. Equals to `CINPUT_ACCEPT_NUMERIC |CINPUT_ACCEPT_LETTER`

`CINPUT_NO_AUTOTRANSLATE`
Disables the feature that makes cinput to autodisplay uppercase characters when standard lowercase character set is selected and the user enters letters without the SHIFT key, that would display graphic characters instead of alphabetic ones.

**Return Value:** Count of successfully read characters in buffer

# VIC_BASE

*VIC_BASE* is a pre-processor macro that provides the base address of the VIC-IV chip, i.e., $D000.

*IS_H640* is a pre-processor macro that returns 0 if the current VIC-III/IV video mode is set to 320 pixels accross (40 column mode), and non-zero if it is set to 640 pixels across (80 column mode).

# CHAPTER 9

# BASIC Tokenisers

Various tokenisers for C64 BASIC exist, e.g., https://github.com/catseye/hatoucan, https://www.c64-wiki.com/wiki/C64list, or the **petcat** utility that is part of VICE. If you are using Ubuntu Linux, you can install **petcat** by using the following command:

```
sudo apt-get install vice
```

We recommend **petcat**, because it supports both C64 BASIC 2 and C65 BASIC 10.

Some IDEs offer BASIC 65 tokenisers within them, such as:

**Eleven**

https://files.mega65.org?id=8b189d0b-ea1e-45a7-a4de-87bcb0b11696

**C64 Studio**

https://www.georg-rottensteiner.de/files/C64StudioRelease.zip

**CBM prg Studio**

https://www.ajordison.co.uk

# PART III

## NATIVE DEVELOPMENT TOOLS

# CHAPTER 10

# C65 Memory Monitor

# CHAPTER 11

# MEGA65 Matrix Mode

# CHAPTER 12

# Turbo Assembler

CHAPTER 13

# Dealing with C65 and C64-mode

The MEGA65 normally starts in C65-mode, and there is appeal in making programs that can start directly from C65-mode. However, the different versions of the C65 ROM make this difficult, as there are differences in the memory location of various system variables and ROM routines. If you are developing a program in assembly language or C, it is typically easier to develop it for C64-mode.

But that leaves you with the problem of how to allow it to be started from C65-mode. Our solution to this is the **c65toc65wrapper.asm** program. This short program can be found in the **src/utilities** directory of the https://github.com/mega65/mega65-tools repository. When compiled, it produces a short program that you can pre-pend to your C64-mode oriented MEGA65 program, and have it automatically detect C64 or C65-mode, switch to C64-mode if required, and set your program going. It also switches the CPU to 40MHz. The only assumption it makes, is that your program should be started using **SYS 2061**

This program is particularly handy when using exomizer, as by having the CPU set to 40MHz before depacking allows depacking to happen almost instantly. This is important on the MEGA65, as loading a program often takes only a fraction of a second, but depacking an exomized program at 1MHz can take several seconds.

Finally, you can use this approach to create bootable disks for the MEGA65, by placing the file onto the disk with the name **AUTOBOOT.C65**. This allows you to use C64-mode programs to autoboot on the MEGA65, which is itself very convenient.

# PART IV

## MEGA65 Operating System

# CHAPTER 14

# Developing System Programmes

- Introduction
- Flash Menu
- Format/FDISK Utility
- Keyboard Test Utility
- MEGA65 Configuration Utility
- Freeze Menu
- Freeze Menu Helper Programmes

- Hypervisor
- OpenROM

# INTRODUCTION

The MEGA65 has a number of system programs and utilities that are used at various times to perform various functions. This includes the utilities accessible via the Utility Menu , the Freeze Menu and its own helper programs, as well as the Flash Menu .

A number of these system programs are pre-loaded into the MEGA65 bitstream, while others live on the SD card. For those that are pre-loaded into the MEGA65 bitstream, this works by having areas of pre-initialised memory, that contain the appropriate program. For example, the utilities accessible via the Utility Menu are all located in the colour RAM, while the Flash Menu is located at $50000 - $57FFF.

In one sense, the easiest way to test new versions of these utilities is to generate a new bitstream with the updated versions. However, synthesising a new bitstream is very time consuming, typically taking an hour on a reasonably fast computer. Therefore this chapter explains the procedure for loading an alternate version of each of these system programs, as well as providing some useful information about these programs, how they operate, and the environment in which they operate compared with normal C64 or C65-mode programs.

# FLASH MENU

The flash menu is located in pre-initialised RAM at $50000 - $57FFF. It is executed during the first boot each time the MEGA65 is switched on. It is unusual in that it executes in the hypervisor context. This is so that it has access to the QSPI flash, which is not available outside of Hypervisor Mode, so that user programs cannot corrupt the cores stored in the flash.

It is also important to note that the flash menu program must fit *entirely* below $8000 when loaded *and* executing, as the Hypervisor is still mapped at $8000 - $BFFF, and can easily be corrupted by an ill behaved flash menu program. In this regard, the flash menu can be regarded as an extension of the hypervisor that is discarded after the first boot. This is unlike all other system programs, that operate in a dedicated memory context, from where the Hypervisor is safe from corruption. It also means that you can't crunch the flash menu to make it fit, as it would overwrite the Hypervisor during decrunching.

Also, as the flash menu is executed very early in the boot process, only the pre-included OpenROM ROM image is available. Thus you must ensure that your flash menu program is compatible with that ROM.

The Hypervisor maintains a flag that indicates whether the flash menu has been executed or not. This flag is updated at the point where the Hypervisor exits to user

mode for the first time, since after that point, the contents of $50000 – $57FFF can no longer be trusted to contain the flash menu. This means that if you wish to have the Hypervisor run a new version of the flash menu that you have loaded, you must prevent the Hypervisor from exiting to user mode first.

The easiest way to achieve this is to hold the ALT key down while powering on the MEGA65. This will cause the Hypervisor to display the Utility Menu, rather than exiting to user mode. It is safe at this time to use the `m65` utility to load the replacement flash menu program using a command similar to the following:

```
m65 -@ newflashmenu.prg@50000
```

That command would load the file `newflashmenu.prg` at memory location $50000.

After that, you can simply press the reset button on the side of the MEGA65 while holding NO SCROLL down, and it will boot again, and because it never left Hypervisor Mode during the previous boot cycle, it will run your updated flash menu program.

It should also be possible to completely automate this process, by first using `m65 -b` to load a new bitstream, thus simulating a cold boot, and then quickly calling `m65` again to simulate depressing the ALT key (or herhaps simply halting the processor), then `m65 -@ ...` and finally `m65 -F` to reset the machine. Writing a script or utility that correctly implements this automation is left as an exercise for the reader.

# FORMAT/FDISK UTILITY

The Format/FDISK utility is accessed as part of the Utility Menu system. These utilities are compiled, crunched and linked using the `utilpacker` program. If you have checked out the mega65-core source repository, you can re-build the colour RAM image by using:

```
make bin/COLOURRAM.BIN
```

You will of course need to first have modified the Format/FDISK utility, which is normally located in the `src/mega65-freeze-menu` subdirectory.

You need to then load this modified colour RAM image into the running machine. Similar to when updating the flash menu, the Hypervisor will only present the utility menu on the first boot, before exiting to user mode for the first time, because it cannot otherwise be sure that the colour RAM contains the valid utility programs.

So as for the flash menu, you would power the MEGA65 off, and then holding the ALT key down, you switch the MEGA65 back on, so that it displays the utility menu. At this point you can use the following command to load your modified `COLOURRAM.BIN` file:

```
m65 -c COLOURRAM.BIN
```

You can now hold ALT down, and press the reset button on the left-hand side of the MEGA65, which should again present the utility menu, but this time with your modified format/fdisk utility in place.

# KEYBOARD TEST UTILITY

The process for updating the Keyboard test utility is essentially the same as for the format/FDISK utility, as it lives in the colour RAM

# MEGA65 CONFIGURATION UTILITY

The process for updating the MEGA65 Configuration utility is essentially the same as for the format/FDISK utility, as it lives in the colour RAM

# FREEZE MENU

The Freeze Menu is a normal program, which is stored in `FREEZER.M65` on the SD card's FAT32 file system.

To updated the Freeze Menu, simply use the `m65ftp` utility or some other means to upload your updated FREEZER.M65 file to the SD card's FAT32 file system. The format of the program is simply a C64-mode PRG file, just renamed to FREEZER.M65.

# FREEZE MENU HELPER PROGRAMMES

The Freeze Menu helper programs are updated in the same way as the Freeze Menu itself.

# HYPERVISOR

The Hypervisor is normally built as HICKUP.M65, a 16KB file that contains the complete Hypervisor program. MEGA65 bitstreams contain a pre-build version located at $FFF8000 - $FFFBFFF. Updated versions of the Hypervisor can be tested using two main approaches:

- 1. Place the updated HICKUP.M65 file on the FAT32 file system of the SD card, and then power the MEGA65 off and on. This works because the Hypervisor contains code that checks for an updated version of itself, and if found, loads it. However this approach is problematic in that if you install a newer bitstream, it will still downgrade the Hypervisor to whatever version is found in the HICKUP.M65 file on the SD card. This method is only recommended for developers who have a need to test their modified Hypervisor code from a cold start. Even then, it is recommended to remove the HICKUP.M65 file immediately after testing to avoid unexpected down-grading in the future.
- 2. Use the m65 command's `-k` option to replace the Hypervisor in place, and then reset the MEGA65 using the reset button on the left-hand side of the case. This should be done when the Hypervisor is *not* active, so that corruption of current execution cannot occur. However, it must also occur before any ROM has been loaded to replace the default OpenROM image. This is because the Hypervisor will attempt to call into the ROM on first-boot in prepration for calling the flash menu, and assumes that the OpenROM is present, because it uses a special OpenROM-specific call to initialise parts of the system state for the flash menu. This is best done by using a command like `m65 -k bin/HICKUP.M65 -R bin/MEGA65.ROM` to load both a new Hypervisor program and re-load an OpenROM image.

# OPENROM

To load a new version of a ROM, there are several options, including replacing both the Hypervisor and ROM at the same time, as described above. However, typically the easiest is to copy the new ROM onto the FAT32 filesystem of the SD card as either MEGA65.ROM, or MEGA65$x$.ROM, where $x$ is replaced by a digit between 0 and 9. When reseting the MEGA65, MEGA65.ROM will then be loaded as normal, or if a digit between 0 and 9 is held down on the keyboard while resetting, the Hypervisor will instead load MEGA65$x$.ROM, where $x$ is the number being held down on the keyboard.

# CHAPTER 15

# MEGA65 Hyppo Services

- Introduction
- General Services
- Drive/Storage Services
- Disk Image Services
- Task and Process Services
- System Partition Services
- Freezer Services

# INTRODUCTION

A part of the MEGA65 is the system program called Hyppo that:

- Boots the MEGA65.
- Loads the ROMs and other files from the SD card.
- Makes memory banks 2 and 3 ROM-like by protecting them from being written to.
- Virtualises the floppy disk controller so you can use disk images.
- Launches various utilities like the freezer and the Matrix Mode Debugger.
- Provides services specific to the MEGA65 that you can use in your programs.

If you know about hypervisors and virtual machines, Hyppo is a very limited hypervisor. Don't expect to be able to run multiple virtual machines concurrently with full isolation. Hyppo runs things that are more akin to the task and processes of a modern operating system than the virtual machines of a hypervisor as you might know it.

Hyppo provides 3 operating modes.

- **The C64-like operating mode** runs C64 programs and MEGA65 programs that run in the MEGA65's C64 mode. When you boot with pressed or use the **GO64** command, Hyppo starts a process in the C64-like operating mode to run BASIC 2 or the MEGA65 program.
- **The C65-like operating mode** is the MEGA65's normal operating mode. This is where regular MEGA65 program run, including BASIC 65 programs.
- **The MEGA65 operating mode** runs the MEGA65's system programs like the freezer, the configuration utility and the Matrix Mode Debugger. Maybe surprisingly, normal MEGA65 programs do not run in the MEGA65 operating mode. They run in the C65-like operating mode. The MEGA65 operating mode is designed solely for the MEGA65 and does not attempt to be compatible with or even be similar to previous systems.

Unlike on the C128, it is possible for a program to effectively change the operating mode while it's is running, by simply enabling or disabling the various hardware features.

Hyppo provides very limited virtualisation of the MEGA65's hardware. It can virtualise the floppy controller. There are plans to virtualise the serial bus so the MEGA65 can use disk images for units like the 1541.

There are some parts of the hardware that only Hyppo can access. It is the only component that can directly access the internal and external SD cards. You need to use Hyppo's services if you want to access the files and directories on the SD cards from within your programs.

# Terminology

When you start to learn about Hyppo, there can be some terminology that might be confusing if you already know about other parts of the MEGA65.

On the SD card there is likely to be a file called HICKUP.M65. This file updates Hyppo to new versions without having to install an upgraded core. You might find occasions where Hyppo might be called Hickup because of this strong association.

There are 3 distinct disk operating systems in the MEGA65.

- Inside Hyppo is Hyppo DOS, or HDOS for short. HDOS is for accessing the FAT32 file system on the SD cards. HDOS does not know anything about Commodore file systems. It can attach an image of a Commodore file system, but it does not understand what is inside the image.
- Inside the KERNAL is CBDOS. CBDOS is for accessing 1581-like file systems. CBDOS uses the 45IO27 multi-function I/O controller to access the sectors of a physical disk. CBDOS does not know anything about SD cards and the FAT32 file system on them. Hyppo virtualises part of the 45IO27 so CBDOS can access disk images like they're physical disks.
- The external disk units attached to the serial bus each have their own DOS. They are used for accessing the file systems on their respective physical disks.

The word drive means different things for each of these DOS's.

- The drives in Hyppo are the partitions of the internal and external SD cards. When the MEGA65 boots, Hyppo assigns numbers to the partitions it can read.
- The drives in CBDOS are the physical disk drives attached to the 45IO27 multi-function I/O controller — such as the internal disk drive — or the disk images attached to the virtualised 45IO27. The CBDOS drives are normally seen as units 8 and 9.
- The drives in an external unit attached to the serial bus are the disk drives inside that unit.

# Versions

This chapter describes the services available in Hyppo 1.2.

New Hyppo services may become available and existing Hyppo services may change or be deprecated. A robust program will use the hyppo_getversion service to check whether it is compatible with the Hyppo in the MEGA65 it's running on.

# Using

When you want to use a Hyppo service, you don't use JSR. This is because Hyppo exists in a space that's separate from regular code. In order to access it, the CPU needs to switch into its hypervisor mode.

At addresses $D640 – $D67F are a set of hypervisor traps. Writing to these addresses are not like writing to other addresses. Instead of writing to memory or I/O, the CPU switches into the hypervisor mode and starts a Hyppo service. How the CPU does this is described in *the MEGA65 Book*, 45GS02 Microprocessor (Appendix J).

Which Hyppo service starts depends on what value from the A register you write and which trap you write to. Each of the services described in this chapter tells you what value to write and which trap to use. You have to use the A register when triggering a trap. Writing the same value from another register won't work.

When the Hyppo service finishes, the CPU will switch back to your program. Except for the registers a service uses to return values, the registers are otherwise preserved.

**Important** The CPU may or may not execute the next byte in your program after the Hyppo service finishes. Put a CLV instruction after the STA. The CPU executing the CLV or not shouldn't matter to your program. If your program does rely on the V flag, you can use the NOP instruction instead. When you use NOP you must be mindful of when the CPU interprets the NOP as a prefix for the following instruction. For this reason, you should prefer using CLV over NOP.

# Errors

If the service was successful, it will set the C flag.

If the service was unsuccessful, it will clear the C flag and put an error code in the A register. There is a table of error codes in the description for hyppo_geterrorcode.

# Examples

The examples use the ACME assembler. The ACME assembler is not required. The Hyppo services can be used with any assembler.

The examples are often not complete. They assume an error handler called error is defined somewhere. They also assume a transfer area has been defined somewhere. hyppo_setup_transfer_area and hyppo_setname show how to define a transfer area.

# GENERAL SERVICES

## hyppo_geterrorcode

**Trap:** LDA #$38 : STA $D640 : CLV

**Service:** Returns the current error code from Hyppo.

**Precondition:** The previous service used cleared the C flag.

**Outputs:** **A** The error code of the previously failed service.

**History:** Available since Hyppo 1.2

**Remarks:** The error code is only valid if the previous Hyppo service cleared the C flag. If the C flag was set there was no error and the Hyppo error code is undefined.

The meanings here are generic. See the sections for the services for more specific meanings.

**Error codes:** This is possibly not an exhaustive list.

| Hex | Dec | Name | General meaning |
|---|---|---|---|
| $01 | 1 | partition not interesting | The partition is not of a supported type. |
| $02 | 2 | bad signature | The signature bytes at the end of a partition table or of the first sector of a partition were missing or incorrect. |
| $03 | 3 | is small FAT | This is partition is FAT12 or FAT16 partition. Only FAT32 is supported. |
| $04 | 4 | too many reserved clusters | The partition has more than 65,535 reserved sectors. |
| $05 | 5 | not two FATs | The partition does not have exactly two copies of the FAT structure. |
| $06 | 6 | too few clusters | The partition contains too few clusters. |
| $07 | 7 | read timeout | It took to long to read from the SD card. |
| $08 | 8 | partition error | An unspecified error occurred while handling a partition. |
| $10 | 16 | invalid address | An invalid address was supplied in an argument. |

| Hex | Dec | Name | General meaning |
|---|---|---|---|
| $11 | 17 | illegal value | An illegal value was supplied in an argument. |
| $20 | 32 | read error | An unspecified error occurred while reading. |
| $21 | 33 | write error | An unspecified error occurred while writing. |
| $80 | 128 | no such drive | The supplied Hyppo drive number does not exist. |
| $81 | 129 | name too long | The supplied filename was too long. |
| $82 | 130 | not implemented | The Hyppo service is not implemented. |
| $83 | 131 | file too long | The file is larger than 16MB. |
| $84 | 132 | too many open files | All of the file descriptors are in use. |
| $85 | 133 | invalid cluster | The supplied cluster number is invalid. |
| $86 | 134 | is a directory | An attempt was made to operate on a directory, where a normal file was expected. |
| $87 | 135 | not a directory | An attempt was made to operate on a normal file, where a directory was expected. |
| $88 | 136 | file not found | The file could not be located in the current directory of the current drive. |
| $89 | 137 | invalid file descriptor | An invalid or closed file descriptor was supplied. |
| $8A | 138 | image wrong length | The disk image file has the wrong length. |
| $8B | 139 | image fragmented | The disk image is not stored contiguously on the SD card. |
| $8C | 140 | no space | The SD card has no free space for the requested operation. |
| $8D | 141 | file exists | A file already exists with the given name. |
| $8E | 142 | directory full | The directory cannot accommodate any more entries. |
| $FF | 255 | eof | The end of a file or directory was encountered. |
| $FF | 255 | no such trap | There is no Hyppo service available for the trap. The program may be incompatible with this version of Hyppo. |

# hyppo_getversion

**Trap:** LDA #$00 : STA $D640 : CLV

**Service:** Returns the version of Hyppo **A.X** and HDOS **Y.Z**.

**Outputs:** **A** The major version number of Hyppo

**X** The minor version number of Hyppo

**Y** The major version number of HDOS

**Z** The minor version number of HDOS

**History:** Available since Hyppo 1.2

**Remarks:** The HDOS in Hyppo is not related to the CBDOS inside the KERNAL or the DOS in the disk drive units attached to the serial port.

**Example:** Tests if Hyppo's version is $\geq$ 1.2 and $<$ 2.0.

```
; Get the version numbers
LDA #$00 : STA $D640 : CLV
; Test if the major version number of Hyppo in A is 1
CMP #1 : BNE incompatible
; Test if the minor version number of Hyppo in X is 2 or more
TXA : CMP #2 : BMI incompatible
; This Hyppo version is compatible
```

# hyppo_setup_transfer_area

**Trap:** LDA #$3A : STA $D640 : CLV

**Service:** Sets up the area Hyppo uses to transfer data to and from your program.

**Inputs:** **Y** The MSB of the transfer area's address

**Errors:** **$10 invalid address** The transfer area address in Y > $7E

**History:** Available since Hyppo 1.2

**Remarks:** The transfer area must be between $0000 and $7E00. It must also begin on a page boundary. The LSB of its address must be $00.

The transfer area is 256 bytes long for most services.

The transfer area is indicated using the CPU's current memory mapping at the time that a service is used. However, it is good practice to always place it in the bottom 32KB of bank 0.

**Example:** Reserves 256 bytes on a page boundary and sets it up as the transfer area.

```
;-----------------------------------------------------------------------------
; Somewhere in the program's data area

    ; Align with the new page boundary
    !align 255, 0

transferarea:
    ; Reserve 256 bytes
    !skip 256

;-----------------------------------------------------------------------------
; Elsewhere in the progrma's code

setuptransferarea:
    ; Set the transferarea as the Hyppo transfer area
    LDY #>transferarea : LDA #$3A : STA $D640 : CLV : BCC error
```

# DRIVE/STORAGE SERVICES

In Hyppo, drives are the partitions of the internal and external SD cards. They are not the drive 0 and drive 1 of the F011 floppy controller. They are also not the drive 0 and drive 1 of dual-drive units attached to the serial bus.

## hyppo_chdir

**Trap:** LDA #$0C : STA $D640 : CLV

**Service:** Changes the current working directory.

**Preconditions:** The FAT dir entry for the directory you want to change to has been found. hyppo_findfile is typically used to find a FAT dir entry. hyppo_findfirst , hyppo_findnext and hyppo_readdir can also be used.

**Errors:** **$87 not a directory** The FAT dir entry last found isn't for a directory. Bit 4 of the FAT dir entry's attribute byte is set for directories.

**History:** Available since Hyppo 1.2

**Remarks:** You can move up to the parent directory by finding the .. FAT dir entry.

You cannot move up or down more than one directory at a time.

Use hyppo_cdrootdir to directly change back to the root directory.

**Example:** Changes to an arbitrary path on the current drive. Call with Y:X being the address of the path. The last character of each component in the path needs to have bit 7 set. The whole path is terminated with a $00. For example, to change into DIR1 and then DIR2 the path would be `!text "DIR", '1'+$80, "DIR" '2'+$80, 0`.

If successful, returns with the C flag set. If some part of the path doesn't exist, returns with the C flag cleared and the current working directory will be whatever directory was last successfully navigated to.

```
    !addr pathptr = $C4
chdirpath:
    STX pathptr : STY pathptr+1 ; Set pathptr to Y:Z
@chdirpath10:
```

... continues on the next page ...

```
    JSR @copycomponent ; Get the next component of the path
    BEQ @chdirpath20 ; Stop at a 0 byte in the path
    JSR @trychdir ; Try to change into the directory
    BCC @chdirpath30 ; Stop if we cannot change into the directory
    BRA @chdirpath10 ; Repeat with the next component
@chdirpath20: SEC
@chdirpath30: RTS
;-----------------------------------------------------------------------------------------
@trychdir:
    ; Set the Hyppo filename from transferbuffer
    LDY #>transferbuffer : LDA #$2E : STA $D640 : CLV : BCC @trychdir10
    ; Find the FAT dir entry
    LDA #$34 : STA $D640 : CLV : BCC @trychdir10
    ; Chdir into the directory
    LDA #$0C : STA $D640 : CLV
@trychdir10:
    RTS
;-----------------------------------------------------------------------------------------
    ; Copy the next component of the path into transferbuffer
@copycomponent:
    LDY #0
@copycomponent10:
    LDA (pathptr),Y
    BEQ @copycomponent20
    TAX
    AND #$7F : STA transferbuffer,Y
    INY
    TXA : BPL @copycomponent10
    ; Add a 0 terminating byte to transferbuffer
    LDA #0 : STA transferbuffer,Y
    ; Move pathptr
    TYA
    CLC : ADC pathptr : STA pathptr : LDA #0 : ADC pathptr+1 : STA pathptr+1
@copycomponent20:
    RTS
```

# hyppo_cdrootdir

**Trap:** LDA #$3C : STA $D640 : CLV

**Service:** Change drive by calling hyppo_selectdrive and set the current directory to the root directory.

**Inputs:** **X** The drive number to become the new current drive

**Precondition:** None.

**Outputs:** **A** Any error code that can be returned by hyppo_selectdrive

**History:** Available since Hyppo 1.16

# hyppo_closeall

**Trap:** LDA #$22 : STA $D640 : CLV

**Service:** Closes all the file descriptors.

**Postconditions:** Using any file descriptor with hyppo_closedir or hyppo_closefile succeeds.

Using any file descriptor with hyppo_readdir or hyppo_readfile fails.

hyppo_opendir and hyppo_openfile reuse the file descriptor.

**History:** Available since Hyppo 1.2

**Remarks:** You can also close individual file descriptors using hyppo_closedir or hyppo_closefile .

# hyppo_closedir

**Trap:** LDA #$16 : STA $D640 : CLV

**Service:** Closes a file descriptor for a directory.

**Preconditions:** The file descriptor given in the X register was opened using hyppo_opendir .

**Inputs:** **X** The file descriptor for the directory

**Postconditions:** Using the file descriptor again with hyppo_closedir succeeds.

Using the file descriptor again with hyppo_readdir fails.

hyppo_opendir and hyppo_openfile reuse the file descriptor.

**History:** Available since Hyppo 1.2

**Remarks:** You can also close all the open file descriptors using hyppo_closeall .

**Example:** See the example in hyppo_opendir .

# hyppo_closefile

**Trap:** LDA #$20 : STA $D640 : CLV

**Service:** Closes a file descriptor for a file.

**Preconditions:** The file descriptor given in the X register was opened using hyppo_openfile .

**Inputs:** **X** The file descriptor for the file

**Postconditions:** Using the file descriptor again with hyppo_closefile succeeds.

Using the file descriptor again with hyppo_readfile fails.

hyppo_opendir and hyppo_openfile reuse the file descriptor.

**History:** Available since Hyppo 1.2

**Remarks:** You can also close all the open file descriptors using hyppo_closeall .

# hyppo_filedate

**Trap:** LDA #$2C : STA $D640 : CLV

**Service:** Sets time stamp of a file.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_findfile

**Trap:** LDA #$34 : STA $D640 : CLV

**Service:** Finds the first file whose filename matches the current Hyppo filename.

**Preconditions:** The current Hyppo filename has been set using hyppo_setname .

**Postconditions:** No additional file descriptors are open.

**Errors:** **$88 file not found** A matching file was not found in the current directory of the current drive.

**History:** Available since Hyppo 1.2

**Remarks:** Hyppo will only find files whose long filename is all in uppercase. Hyppo converts the current filename to ASCII uppercase before trying to match it. Bytes $61 – $7B change to $41 – $5A.

Hyppo does not yet support the wildcard characters * and ?. Support for that is planned in a future version.

This only finds the first matching file. You can find multiple matches by using hyppo_findfirst and hyppo_findnext .

**Example:** See the example in hyppo_openfile .

# hyppo_findfirst

**Trap:** LDA #$30 : STA $D640 : CLV

**Service:** Finds the first file whose filename matches the current Hyppo filename.

**Preconditions:** The current Hyppo filename has been set using hyppo_setname .

**Outputs:** **A** The file descriptor for reading the current working directory. You might be responsible for closing this file descriptor using hyppo_closedir . See the remarks.

**Postconditions:** hyppo_findnext find the next matching file or fails with a file not found error.

**Side effects:** Sets the current file descriptor.

**Errors:** **$88 file not found** A matching file was not found in the current directory of the current drive.

**History:** Available since Hyppo 1.2

**Remarks:** If Hyppo finds an initial matching file, it will set the C flag and return a file descriptor in the A register. This is a file descriptor for reading the current working directory. You are responsible for closing it using hyppo_closedir . It's a standard directory file descriptor. You can use hyppo_readdir to read the FAT dir entries after the file that was found.

If Hyppo doesn't find any matching files, it will fail with a file not found error. In this case Hyppo will have already closed the file descriptor and you don't have to close it.

Hyppo will only find files whose long filename is all in uppercase. Hyppo converts the current filename to ASCII uppercase before trying to match it. Bytes $61 - $7B change to $41 - $5A.

Hyppo does not yet support the wildcard characters * and ?. Support for that is planned in a future version.

If you are only interested in the first match, you can use hyppo_findfile instead. hyppo_findfile always closes the file descriptor for you. But you can't use it to find multiple matching files.

**Example:** See the example in hyppo_findnext .

# hyppo_findnext

**Trap:** LDA #$32 : STA $D640 : CLV

**Service:** Finds a subsequent file whose filename matches the current Hyppo filename.

**Preconditions:** The current Hyppo filename has been set using hyppo_setname .

The first matching file has already been found successfully using hyppo_findfirst .

**Postconditions:** Using hyppo_findnext again finds the next matching file or fails with a file not found error.

**Errors:** **$88 file not found** A subsequent matching file was not found in the current directory of the current drive.

**History:** Available since Hyppo 1.2

**Remarks:** If Hyppo doesn't find a subsequent matching file, it will fail with a file not found error. Hyppo will also close the file descriptor it output in hyppo_findfirst .

If you don't exhaust the search by using hyppo_findnext until it fails with a file not found error, you are required to close the file descriptor yourself using hyppo_closedir .

**Example:** Returns with X register containing the number of files matching the current Hyppo filename in the current working directory of the current drive. While a directory can in theory have multiple files with an indentical name, this example will be more useful once Hyppo supports * and ? wildcards.

```
    ; Assume the current Hyppo filename has already been set.
    ; Reset the count in X.
    LDX #0
    ; Try to find the first matching file.
    LDA #$30
@10:
    STA $D640 : CLV : BCC @20
    ; Increment the count in X
    INX
    ; Try to find the next matching file.
    LDA #$32
    BRA @10
```

... continues on the next page ...

```
@20:
    ; If the error code in A is $88 there are no more matching files,
    ; otherwise an error occurred.
    CMP #$88 : BNE error
    ; No need to close the file handle because the search exhausted.
```

# hyppo_fstat

**Trap:** LDA #$28 : STA $D640 : CLV

**Service:** Returns information about a file.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_getcurrentdrive

**Trap:** LDA #$04 : STA $D640 : CLV

**Service:** Returns the number of the currently selected drive (SD card partition).

**Outputs:** **A** The current drive number

**History:** Available since Hyppo 1.2

**Remarks:** hyppo_selectdrive changes the current drive number. hyppo_cdrootdir can also change it.

**Example:** Prints the number of the currently selected drive in the top-left of the screen. This example assumes that there aren't more than 10 drives (drives 0 to 9). It also assumes the screen memory hasn't been moved from $800.

```
; Get the current drive
LDA #$04 : STA $D640 : CLV : BCC error

; Convert the drive number in A into a screen code
CLC : ADC #$30

; Put the screen code into the top-left of screen memory
STA $0800
```

# hyppo_getcwd

**Trap:** LDA #$0A : STA $D640 : CLV

**Service:** Returns information on the currently selected directory or sub-directory.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_getdefaultdrive

**Trap:** LDA #$02 : STA $D640 : CLV

**Service:** Returns the drive number (SD card partition) Hyppo selected while booting.

**Outputs:** **A** The default drive number

**History:** Available since Hyppo 1.2

**Example:** Selects the default drive.

```
; Get the default drive
LDA #$02 : STA $D640 : CLV : BCC @error

; Transfer the drive number in A to X
TAX

; Select the default drive
LDA #$06 : STA $D640 : CLV : BCC @error
```

# hyppo_getdrivesize

**Trap:** LDA #$08 : STA $D640 : CLV

**Service:** Returns information on the size of the currently selected drive (SD card partition).

**Remarks:** **NOT IMPLEMENTED**

# hyppo_loadfile

**Trap:** LDA #$36 : STA $D640 : CLV

**Service:** Loads a file into chip memory.

**Preconditions:** The name of the file to load has been set using hyppo_setname .

**Inputs:** **X** The LSB of the address to start loading from

**Y** The middle byte of the address to start loading from

**Z** The MSB of the address to start loading from

**Postconditions:** No additional file descriptors are open.

**Errors:** **$84 too many open files** hyppo_loadfile uses one file descriptor internally, but all the file descriptors are in use. Close some or all of the file descriptors using hyppo_closedir , hyppo_closefile or hyppo_closeall .

**$88 file not found** The file was not found in the current directory of the current drive.

**History:** Available since Hyppo 1.2

**Remarks:** This service can load files up to 16MB in size into the first 16MB of chip memory. Chip memory is the 384KB or more of memory inside the CPU module.

Loading will start at 28-bit address $00ZZYYXX. If loading tries to go beyond $00FFFFFF, it wraps around and continue at $00000000.

You can use hyppo_loadfile_attic to load a file into hyper memory. The hyper memory is the 8MB or more of memory in the external RAM chips.

**Example:** Loads a file into memory starting at $48000.

```
; Assume the current Hyppo filename has already been set.
; No need to find the file first.
LDZ #$04 ; Most significant byte
LDY #$80 ; Middle byte
LDX #$00 ; Least significant byte
LDA #$36 : STA $D640 : CLV : BCC error
```

# hyppo_loadfile_attic

**Trap:** LDA #$3E : STA $D640 : CLV

**Service:** Loads a file into hyper memory.

**Preconditions:** The name of the file to load has been set using hyppo_setname .

**Inputs:** **X** The LSB of the address to start loading from

**Y** The middle byte of the address to start loading from

**Z** The MSB of the address to start loading from

**Postconditions:** No additional file descriptors are open.

**Errors:** **$84 too many open files** hos_loadfile_attic uses one file descriptor internally, but all the file descriptors are in use. Close some or all of the file descriptors using hyppo_closedir , hyppo_closefile or hyppo_closeall .

**$88 file not found** The file was not found in the current directory of the current drive.

**History:** Available since Hyppo 1.2

**Remarks:** This service can load files up to 16MB in size into the first 16MB of hyper memory. Hyper memory is the 8MB or more of memory in the external RAM chips.

Loading will start at 28-bit address $08ZZYYXX. If loading tries to go beyond $08FFFFFF, the loading will wrap around and continue at $08000000.

You can use hyppo_loadfile to load a file into chip memory. The chip memory is the 384KB or more of memory inside the CPU module.

# hyppo_mkdir

**Trap:** LDA #$0E : STA $D640 : CLV

**Service:** Creates a sub-directory.

**Errors:** **$8D file exists** A sub-directory or file already exists with the current Hyppo filename in the current working directory of the current drive.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_mkfile

**Trap:** LDA #$1E : STA $D640 : CLV

**Service:** Creates a file.

**Errors:** **$8D file exists** A sub-directory or file already exists with the current Hyppo filename in the current working directory of the current drive.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_opendir

**Trap:** LDA #$12 : STA $D640 : CLV

**Service:** Opens the current working directory for reading the file entries in it.

**Preconditions:** The drive and directory you want to read have already been set up using hyppo_selectdrive and hyppo_chdir if necessary.

**Outputs:** **A** The file descriptor for reading the directory. You are responsible for closing this file descriptor using hyppo_closedir .

**Postconditions:** hyppo_readdir reads the first FAT dir entry in the directory.

**Errors:** **$84 too many open files** All the file descriptors are in use. hyppo_opendir and hyppo_openfile share the same very small pool of file descriptors. Close some or all of the file descriptors using hyppo_closedir , hyppo_closefile or hyppo_closeall .

**$87 not a directory** The FAT dir entry last found is for a file. Use hyppo_openfile for files.

**History:** Available since Hyppo 1.2

**Example:** Calls processdirentry for each FAT dir entry in the current working directory. processdirentry is assumed to be defined elsewhere.

```
    ; Open the current working directory
    LDA #$12 : STA $D640 : CLV : BCC error
    ; Transfer the directory file descriptor into X
    TAX
    ; Set Y to the MSB of the transfer area
    LDY #>transferarea
@10:
    ; Read the directory entry
    LDA #$14 : STA $D640 : CLV : BCC @20
    ; Call processdirentry (assumed to be defined elsewhere)
    PHX : PHY : JSR processdirentry : PLY : PLX
    BRA @10
@20:
    ; If the error code in A is $85 we have reached the end of the
    ; directory otherwise there's been an error
    CMP #$85 : BNE error
    ; Close the directory file descriptor in X
    LDA #$16 : STA $D640 : CLV : BCC error
```

# hyppo_openfile

**Trap:** LDA #$18 : STA $D640 : CLV

**Service:** Opens a file on a drive.

**Preconditions:** The file has already been found. Files can be found using hyppo_findfile , hyppo_findfirst and hyppo_findnext . hyppo_readdir can also be used to find a file.

**Outputs:** **A** The file descriptor for accessing the file. You are responsible for closing this file descriptor using hyppo_closefile .

**Postconditions:** Using hyppo_readfile with this file descriptor reads the first sector of the file.

**Side effects:** Sets the current file to the newly opened file. hyppo_readfile reads from the current file.

**Errors:** **$84 too many open files** All the file descriptors are in use. hyppo_opendir and hyppo_openfile share the same very small pool of file descriptors. Close some or all of the file descriptors using hyppo_closedir , hyppo_closefile or hyppo_closeall .

**$86 is a directory** The FAT dir entry last found is for a directory. Use hyppo_opendir for directories.

**History:** Available since Hyppo 1.2

**Remarks:** You cannot use this to open a file inside a disk image. To do that you use hyppo_d81attach0 or hyppo_d81attach1 to attach the disk image and then use either use the KERNAL to read the file or program the virtualised F011 floppy controller.

**Example:** Finds and opens a file.

```
; Assume the current Hyppo filename has already been set.
; Find the file
LDA #$34 : STA $D640 : CLV : BCC error
; Open the file
LDA #$18 : STA $D640 : CLV : BCC error
```

# hyppo_readdir

**Trap:** LDA #$14 : STA $D640 : CLV

**Service:** Reads the next FAT dir entry into a destination area.

**Preconditions:** The file descriptor given in the X register was opened using hyppo_opendir and hyppo_closedir hasn't since been used to close it.

The destination area is on a page boundary between $0000 and $7E00 and is at least 87 bytes.

**Inputs:** **X** The file descriptor for the directory.

**Y** The MSB of the destination area.

**Outputs:** Starting at $YY00, the FAT dir entry has this structure.

| Offset | Type | Description |
|---|---|---|
| $00 | asciiz | The long file name |
| $40 | byte | The length of long file name |
| $41 | ascii | The "8.3" file name. The name part is padded with spaces to make it exactly 8 bytes. The 3 bytes of the extension follow. There is no . between the name and the extension. There is no NULL byte. |
| $4E | dword | The cluster number where the file begins. For sub-directories, this is where the FAT dir entries start for that sub-directory. |
| $52 | dword | The length of file in bytes. |
| $56 | byte | The type and attribute bits. |

This is what the bits in the last byte mean. Bits 6 and 7 are undefined.

| Bit | Meaning if bit is set |
|---|---|
| 0 | Read only |
| 1 | Hidden |
| 2 | System |
| 3 | Volume label |
| 4 | Sub-directory |
| 5 | Archive |

**Postconditions:** Using hyppo_readdir again reads the next FAT dir entry in the directory.

**Errors:** **$08 partition error** An unspecified error occurred while handling the currently selected partition.

**$10 invalid address** The Y register is > $7E.

**$85 invalid cluster** An attempt was made to read past the end of the directory.

**Remarks:** If the long file name in the FAT dir entry is too long to copy into the destination area, Hyppo skips the entry entirely.

The file names in FAT are encoded as UTF-16. Hyppo only reads the LSB of each 16-bit character. Hyppo does not convert file names into PETSCII.

See hyppo_setup_transfer_area for more details about the value for the Y register.

**History:** Available since Hyppo 1.2

**Example:** See the example in hyppo_opendir .

# hyppo_readfile

**Trap:** LDA #$1A : STA $D640 : CLV

**Service:** Reads the next sector of the current file into the sector buffer.

**Preconditions:** There is a current file open. Files can be opened with hyppo_openfile .

**Outputs:** **X** The LSB of the number of bytes read

**Y** The MSB of the number of bytes read

**Postconditions:** The next call to hyppo_readfile will read the next sector of the current file or signal the end of the file.

**Errors:** **$89 invalid file descriptor** There is no current file.

**History:** Available since Hyppo 1.2

**Remarks:** To access the data, you need to either:

- map the sector buffer into the 16-bit address space;
- use an enhanced DMA transfer to copy the sector buffer at $FFD6E00 – $FFD6FFF into a buffer already mapping into the 16-bit address space; or
- use 32-bit load instructions to access the sector buffer directly.

If a full sector was read, Y:X will be $0200. For the last sector of the file, Y:X may be less than that. Any bytes in the sector buffer after Y:X are undefined and will not necessarily be zero.

If you read past the end of the last sector, Y:X will be $0000, the A register will be $FF and the C flag will be set.

While multiple files can be opened simultaneously, only the current file can be read. The current file is often the last file opened, but not always.

**Example:** Maps the sector buffer to $DE00 and then reads each sector of the file calling proccesssector for each sector read. proccesssector is assumed to be defined elsewhere.

```
; Assume the file is already open.
; Unmap the colour RAM from $DC00 because that will prevent us
; from mapping in the sector buffer
LDA $D030 : PHA : AND #%11111110 : STA $D030
```

... continues on the next page ...

```
@10:
    ; Read the next sector
    LDA #$1A : STA $D640 : CLV : BCC @20
    ; Map the sector buffer to $DE00
    LDA #$81 : STA $D680
    ; Call processsector (assumed to be defined elsewhere)
    JSR processsector
    ; Unmap the sector buffer from $DE00
    LDA #$82 : STA $D680
    BRA @10
@20:
    ; If the error code in A is $FF we have reached the end of the file
    ; otherwise there's been an error
    CMP #$FF : BNE error
    ; Map the colour RAM at $DC00 if it was previously mapped
    PLA : STA $D030
```

# hyppo_rename

**Trap:** LDA #$2A : STA $D640 : CLV

**Service:** Renames a file or sub-directory.

**Errors:** **$8D file exists** A sub-directory or file already exists with the current Hyppo filename in the current working directory of the current drive.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_rmdir

**Trap:** LDA #$10 : STA $D640 : CLV

**Service:** Removes a sub-directory.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_rmfile

**Trap:** LDA #$26 : STA $D640 : CLV

**Service:** Removes a files.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_seekfile

**Trap:** LDA #$24 : STA $D640 : CLV

**Service:** Seeks to a given sector in a file.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_selectdrive

**Trap:** LDA #$06 : STA $D640 : CLV

**Service:** Sets the currently selected drive (SD card partition).

**Preconditions:** Hyppo has assigned a drive number to the SD card partition.

**Inputs:** **X** The drive number to become the new current drive

**Postconditions:** hyppo_getcurrentdrive returns the value that was in the X register.

Hyppo services operate on the newly selected drive.

**Errors:** **$80 no such drive** The drive in the X register does not exist. Hyppo only assigns drive numbers to the SD card partitions it can read.

**History:** Available since Hyppo 1.2

**Example:** Tests if drive 2 exists by trying to select it. Returns with the C flag set if drive 2 exists.

```
doesdrive2exist:
    ; Preserve the current drive so we can restore it later
    LDA #$04 : STA $D640 : CLV : BCC error
    PHA
    ; Try to select drive 2
    LDX #2 : LDA #$06 : STA $D640 : CLV : BCC @10
    ; Restore the previously selected drive
    PLX
    LDA #$06 : STA $D640 : CLV : BCC error
    ; The C flag was already set by the Hyppo service
    RTS

@10:
    ; If the error code in A is $80, the drive doesn't exist; otherwise
    ; some other kind of error occurred
    CMP #$80
    BNE error
    ; Forget about the current drive we preserved because it wasn't
    ; changed
    PLX
    ; Clear the C flag because the drive doesn't exist
    CLC
    RTS
```

# hyppo_setname

**Trap:** LDA #$2E : STA $D640 : CLV

**Service:** Sets the current Hyppo filename.

**Preconditions:** The filename is stored in ASCII and ends with a $00 byte.

The filename starts on a page boundary between $0000 and $7E00 and is less than 63 characters, excluding the $00 byte.

**Inputs:** **Y** The MSB of the filename address.

**Postconditions:** Hyppo has copied the filename into it's own data area.

The hyppo_find* and hyppo_load* services use this filename.

**Side effects:** Sets the transfer area to $YY00.

**Errors:** **$10 invalid address** The Y register is > $7E.

**$81 name too long** The filename is longer than 63 characters.

**History:** Available since Hyppo 1.2

**Remarks:** The filename must be between $0000 and $7E00. It must also begin on a page boundary. That is, its address must end with $00. The current memory mapping is used. However, it is good practice to place it in the bottom 32KB of bank 0.

The filenames in FAT are encoded in UTF-16. Hyppo only reads the LSB of each 16-bit character. Hyppo does not convert between ASCII and PETSCII.

Hyppo accesses the files in the FAT file system on the internal and external SD cards. It does not access files on disks in floppy drives or in disk images.

**Example:** Set the current Hyppo filename to GAME.MAP.

```
; Somewhere in the program's data area
    !align 255, 0 ; Align with the next page boundary
filename:
    !text "GAME.MAP", 0 ; Must end with a 0 byte

; Elsewhere in the progrma's code
setfilename:
    LDY #>filename : LDA #$2E : STA $D640 : CLV : BCC error
```

# hyppo_writefile

**Trap:** LDA #$1C : STA $D640 : CLV

**Service:** Writes the sector buffer to the current file.

**Remarks:** **NOT IMPLEMENTED**

# DISK IMAGE SERVICES

The 45IO27 multi-function I/O controller includes a F011-compatible floppy controller. The internal floppy drive is attached to this as drive 0.

Hyppo can virtualise the F011 floppy controller so that disk images can be attached instead of floppy drives. Once a disk image is attached, Hyppo traps the F011's I/O registers and emulates the commands on the disk image.

You can use BASIC, the KERNAL and the F011 I/O registers to operate on a disk image just as you would a physical disk. The virtualisation does not behave the same as a floppy drive in all cases. If you intend for your program to work with both disk images and physical disks, be sure to test it with both.

## hyppo_d81attach0

**Trap:** LDA #$40 : STA $D640 : CLV

**Service:** Attach a D81 disk image to virtualised F011 drive 0.

**Preconditions:** The current Hyppo filename has been set using hyppo_setname .

**Errors:** **$88 file not found** The disk image file was not found in the current directory of the current drive.

**History:** Available since Hyppo 1.2

**Remarks:** Unless it's been changed, drive 0 of the virtualised F011 floppy controller is unit 8.

**Example:** Attaches the disk image DISK2.D81 to virtualised F011 drive 0.

```
; Somewhere in the program's data area
    !align 255, 0 ; Align with the next page boundary
disk2name:
    !text "DISK2.D81", 0 ; Must end with a 0 byte

; Elsewhere in the progrma's code
attachdisk2:
    ; Set the Hyppo filename to DISK2.D81
    LDY #>disk2name : LDA #$2E : STA $D640 : CLV : BCC error
    ; Attach the disk image
    LDA #$40 : STA $D640 : CLV : BCC error
```

# hyppo_d81attach1

**Trap:** LDA #$46 : STA $D640 : CLV

**Service:** Attach a D81 disk image to virtualised F011 drive 1.

**Preconditions:** The current Hyppo filename has been set using hyppo_setname .

**History:** Available since Hyppo 1.2

**Remarks:** Unless it's been changed, drive 1 of the virtualised F011 floppy controller is unit 9.

# hyppo_d81detach

**Trap:** LDA #$42 : STA $D640 : CLV

**Service:** Detaches any disk images from virtualised F011 drives 0 and 1.

**History:** Available since Hyppo 1.2

# hyppo_d81write_en

**Trap:** LDA #$44 : STA $D640 : CLV

**Service:** Enables writing to any disk images attached to virtualised F011 drives 0 and 1.

**History:** Available since Hyppo 1.2

# TASK AND PROCESS SERVICES

## hyppo_create_task_c64

**Trap:** LDA #$66 : STA $D640 : CLV

**Service:** Creates a Hyppo task in the C64-like operating mode.

**Remarks:** **NOT IMPLEMENTED**

## hyppo_create_task_c65

**Trap:** LDA #$68 : STA $D640 : CLV

**Service:** Creates a Hyppo task in the C65-like operating mode.

**Remarks:** **NOT IMPLEMENTED**

## hyppo_create_task_native

**Trap:** LDA #$62 : STA $D640 : CLV

**Service:** Creates a Hyppo task in the MEGA65 operating mode.

**Remarks:** **NOT IMPLEMENTED**

## hyppo_exit_and_switch_to_task

**Trap:** LDA #$6A : STA $D640 : CLV

**Service:** Exits the current Hyppo task and switches context to another Hyppo task.

**Remarks:** **NOT IMPLEMENTED**

## hyppo_exit_task

**Trap:** LDA #$6E : STA $D640 : CLV

**Service:** Exits the current Hyppo task.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_get_mapping

**Trap:** LDA #$74 : STA $D640 : CLV

**Service:** Copies the current 45GS02 memory mapping into a destination area.

**Preconditions:** The destination area starts on a page boundary between $0000 and $7E00 and is at least 6 bytes.

**Inputs:** **Y** The MSB of the destination area.

**Outputs:** Starting at $YY00, the current mapping info has this structure.

| Offset | Type | Description |
|---|---|---|
| 0 | word | MAPLO |
| 2 | word | MAPHI |
| 4 | byte | The megabyte offset for MAPLO |
| 5 | byte | The megabyte offset for MAPHI |

**Errors:** **$10 invalid address** The Y register is > $7E.

**History:** Available since Hyppo 1.2

**Remarks:** MAPLO is the mapping for $0000 - $7FFF.

MAPHI is the mapping for $8000 - $FFFF.

See *the MEGA65 Book*, 45GS02 Microprocessor (Appendix J) for more information on MEGA65 memory mapping and banking.

# hyppo_get_proc_desc

**Trap:** LDA #$48 : STA $D640 : CLV

**Service:** Copies the current task block into a destination area.

**Preconditions:** The destination area starts on a page boundary between $0000 and $7E00 and is at least 256 bytes.

**Inputs:** **Y** The MSB of the destination area.

**Outputs:** Starting at $YY00, the current task block has this structure.

| Offset | Type | Description |
|---|---|---|
| $00 | byte | The ID of the current task. |
| $01 | text | The name of the current task. A maximum of 16 characters. Padded with $00 bytes. If it's 16 characters, there are no trailing $00 bytes. |
| $11 | byte | Flags for the D81 disk image attached to drive 0 of the virtualised F011 floppy controller. |
| $12 | byte | Same as above but for drive 1. |
| $13 | byte | The length of the D81 disk image filename attached to drive 0. |
| $14 | byte | Same as above but for drive 1. |
| $15 | text | The filename of the D81 disk image attached to drive 0. A maximum of 32 characters. Padded with $20 bytes. There is no trailing $00 byte. |
| $35 | text | Same as above but for drive 1. |
| $55 | | The meaning of these bytes are undefined and subject to change. |
| $80 | | File descriptor 0. |
| $A0 | | File descriptor 1. |
| $C0 | | File descriptor 2. |
| $E0 | | File descriptor 3. |

Each of the file descriptors has this structure.

| Offset | Type | Description |
|---|---|---|
| $00 | byte | The number of the SD card partition where the file resides. $FF means the file descriptor is closed. |
| $01 | dword | The cluster where the file starts |
| $05 | dword | The current cluster |
| $09 | byte | The current sector within the current cluster |
| $0A | dword | The length of the file |
| $0E | dword | The current position within the file's buffer |
| $12 | dword | The cluster of the directory in which the file resides |
| $16 | word | The index of the file within its directory |
| $18 | dword | The absolute 32-bit address of the file's buffer |
| $1C | word | The number of bytes used in the file's buffer |
| $1E | word | The current offset within the file's buffer |

**Errors:** **$10 invalid address** The Y register is > $7E.

**History:** Available since Hyppo 1.2

# hyppo_gettasklist

**Trap:** LDA #$50 : STA $D640 : CLV

**Service:** Gets the list of tasks in Hyppo.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_load_into_task

**Trap:** LDA #$64 : STA $D640 : CLV

**Service:** Loads a file from an SD card partition into the memory of a Hyppo task.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_readoutoftask

**Trap:** LDA #$58 : STA $D640 : CLV

**Service:** Reads from the memory of another Hyppo task.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_receivemessage

**Trap:** LDA #$54 : STA $D640 : CLV

**Service:** Receives messages sent from other Hyppo tasks.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_reset

**Trap:** LDA #$7E : STA $D640 : CLV

**Service:** Warm boots the MEGA65.

**History:** Available since Hyppo 1.2

# hyppo_rom_writeenable

**Trap:** LDA #$02 : STA $D641 : CLV

**Service:** Changes $20000 – $3FFFF to behave like RAM by disabling the write-protection.

**History:** Available since Hyppo 1.2

**Remarks:** $20000 – $3FFFF normally has the KERNAL, BASIC, CBDOS, and font ROMs.

hyppo_rom_writeprotect enables the write-protection and blocks writes. hyppo_toggle_rom_writeprotect toggles the write-protection.

# hyppo_rom_writeprotect

**Trap:** LDA #$00 : STA $D641 : CLV

**Service:** Changes $20000 - $3FFFF to behave like ROM by enabling the write-protection.

**History:** Available since Hyppo 1.2

**Remarks:** $20000 - $3FFFF normally has the KERNAL, BASIC, CBDOS, and font ROMs.

hyppo_rom_writeenable disables the write-protection and allows writes. hyppo_toggle_rom_writeprotect toggles the write-protection.

# hyppo_sendmessage

**Trap:** LDA #$52 : STA $D640 : CLV

**Service:** Sends a message to another Hyppo task.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_serial_monitor_wait_and_write

**Trap:** LDA #$xx : STA $D643 : CLV

**Service:** Waits for the serial monitor or Matrix Mode Debugger to be ready to receive and then writes a character to it.

**Inputs:** **A** The ASCII character to write.

**History:** Available since Hyppo 1.2

**Remarks:** The service waits for the serial monitor to be ready to receive. This could slow down or hang your program. If you don't want this and you are happy for the character to be lost if the serial monitor is not ready to receive, use hyppo_serial_monitor_write .

# hyppo_serial_monitor_write

**Trap:** LDA #$7C : STA $D640 : CLV

**Service:** Writes a character to the serial monitor or the Matrix Mode Debugger.

**Preconditions:** The serial monitor is ready to receive.

**Inputs:** **Y** The ASCII character to write.

**History:** Available since Hyppo 1.2

**Remarks:** The character will be lost if the serial monitor is not ready to receive, If you don't want the character to be lost, use hyppo_serial_monitor_wait_and_write .

# hyppo_set_mapping

**Trap:** LDA #$76 : STA $D640 : CLV

**Service:** Copies the source area into the current 45GS02 memory mapping.

**Preconditions:** The source area starts on a page boundary between $0000 and $7E00 and is at least 6 bytes.

**Inputs:** **Y** The MSB of the source area.

Starting at $YY00, the current mapping info has this structure.

| Offset | Type | Description |
|---|---|---|
| 0 | word | MAPLO |
| 2 | word | MAPHI |
| 4 | byte | The megabyte offset for MAPLO |
| 5 | byte | The megabyte offset for MAPHI |

**Postconditions:** The CPU continues execution with the new memory mapping.

**Errors:** **$10 invalid address** The Y register is > $7E.

**History:** Available since Hyppo 1.2

**Remarks:** You must take care when changing the memory mapping. Hyppo will not take any steps to ensure the instructions after the STA are executed regardless of the mapping. If you change the mapping of the block where the program counter points to, the CPU will resume with the instructions in the newly mapped block.

MAPLO is the mapping for $0000 - $7FFF.

MAPHI is the mapping for $8000 - $FFFF.

See *the MEGA65 Book*, 45GS02 Microprocessor (Appendix J) for more information on MEGA65 memory mapping and banking.

# hyppo_switch_to_task

**Trap:** LDA #$6C : STA $D640 : CLV

**Service:** Switches context to another Hyppo task.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_terminateothertask

**Trap:** LDA #$60 : STA $D640 : CLV

**Service:** Terminates another Hyppo task.

**Remarks:** **NOT IMPLEMENTED**

# hyppo_toggle_force_4502

**Trap:** LDA #$72 : STA $D640 : CLV

**Service:** Toggles the CPU personality between 45GS02 and 6502.

**Outputs:** **A** If bit 5 is set, the CPU is in the 45GS02 personality. If bit 5 is clear, the CPU is in the 6502 personality.

**History:** Available since Hyppo 1.2

**Remarks:** The others bits of the A register are undefined. Do not expect them to be zero.

In the 6502 personality, none of the new opcodes of the 65C02, 65CE02, 4510 or 45GS02 are available. These are replaced with the original — and often strange — behaviour of the undefined opcodes of the 6502.

**Warning** This feature is incomplete and untested. Most undocumented 6502 opcodes do not operate correctly when the 6502 personality is enabled.

**Example:** Enables the 45GS02 personality regardless of the CPU's current personality.

```
@10
    ; Toggle the CPU personality
    LDA #$72 : STA $D640 : CLV : BCC error
    ; Toogle again if bit 5 of A is clear
    AND #%00100000 : BEQ @10
```

# hyppo_toggle_rom_writeprotect

**Trap:** LDA #$70 : STA $D640 : CLV

**Service:** Toggles the write-protection for $20000 - $3FFFF.

**Outputs:** **A** If bit 2 is set, $20000 - $3FFFF cannot be written to.

**History:** Available since Hyppo 1.2

**Remarks:** The others bits of the A register are undefined. Do not expect them to be zero.

If you simply want to disable or enable the protection, you can use hyppo_rom_writeenable and hyppo_rom_writeprotect .

# hyppo_writeintotask

**Trap:** LDA #$56 : STA $D640 : CLV

**Service:** Writes into the memory of another Hyppo task.

**Remarks:** **NOT IMPLEMENTED**

# SYSTEM PARTITION SERVICES

## hyppo_configsector_apply

**Trap:** LDA #$04 : STA $D642 : CLV

**Service:** Applies the system configuration sector currently loaded into memory.

**History:** Available since Hyppo 1.2

## hyppo_configsector_read

**Trap:** LDA #$00 : STA $D642 : CLV

**Service:** Reads the system configuration sector into memory.

**History:** Available since Hyppo 1.2

## hyppo_configsector_write

**Trap:** LDA #$02 : STA $D642 : CLV

**Service:** Writes the system configuration sector from memory.

**History:** Available since Hyppo 1.2

## hyppo_dmagic_autoset

**Trap:** LDA #$06 : STA $D642 : CLV

**Service:** Sets the DMAgic revision based on the loaded ROM.

**History:** Available since Hyppo 1.2

# FREEZER SERVICES

## hyppo_freeze_self

**Trap:** LDA #$xx : STA $D67F : CLV

**Service:** Launches the freezer.

**History:** Available since Hyppo 1.2

## hyppo_get_slot_count

**Trap:** LDA #$16 : STA $D642 : CLV

**Service:** Gets the number of freeze slots.

**History:** Available since Hyppo 1.2

## hyppo_locate_freeze_slot

**Trap:** LDA #$10 : STA $D642 : CLV

**Service:** Locates the first sector of a freeze slot.

**History:** Available since Hyppo 1.2

## hyppo_read_freeze_region_list

**Trap:** LDA #$14 : STA $D642 : CLV

**Service:** Reads the freeze region list.

**History:** Available since Hyppo 1.2

## hyppo_unfreeze_from_slot

**Trap:** LDA #$12 : STA $D642 : CLV

**Service:** Unfreezes from a freeze slot.

**History:** Available since Hyppo 1.2

# PART V

## MULTI-MEDIA AND DATA CONVERSION

# PART VI

## GEOS Development

# PART VII

## HARDWARE

# CHAPTER 16

# Using Nexys4 boards as a MEGA65

- Building your own MEGA65 Compatible Computer
- Working Nexys4 Boards
- Power, Jumpers, Switches and Buttons
- Keyboard
- Preparing microSDHC card
- Loading the bitstream from QSPI
- Widget Board

- PMOD-to-Joystick Adapter
- Useful Tips

# BUILDING YOUR OWN MEGA65 COMPATIBLE COMPUTER

You can build your own MEGA65-compatible computer by using either a Nexys4DDR (aka. Nexys A7) or the older Nexys4 (Non-DDR) FPGA development boards. This appendix describes the process to set up a Nexys4DDR (Nexys A7) board for this purpose (which is the newer, preferred board). The older non-DDR Nexys4 board is also supported, and the instructions are the same, except that you must use a bitstream designed for that board. Using a Nexys4DDR bitstream on a non-DDR Nexys4 board, or vice versa, may cause irreparable damage to your board, so make sure you have the correct bitstream to suit your board.

DISCLAIMER: M.E.G.A cannot take any responsibility for any damage that may occur to your Nexys4DDR/NexysA7/Nexys4 boards.

# WORKING NEXYS4 BOARDS

There are currently 3 Nexys FPGA boards which can be setup as a MEGA65:

## The Nexys4 board

No longer manufactured but still available for sale on some websites with old stock.

Documentation:

- https://reference.digilentinc.com/reference/programmable-logic/nexys-4/reference-manual
- https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4/nexys4_rm.pdf

## The Nexys4DDR board

No longer manufactured but still available for sale on some websites with old stock.

Documentation:

- https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual
- https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4-ddr/nexys4ddr_rm.pdf

# The Nexys A7

This is the re-branded version of the above Nexys4 DDR board:

Documentation:

- https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual
- https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-a7/nexys-a7_rm.pdf

# POWER, JUMPERS, SWITCHES AND BUTTONS

This top-down picture highlights the key jumper positions of interest on the Nexys4 board:

The Nexys4 boards can be powered in two ways: using an external power supply, or from a standard USB port.

# Micro-USB Power

Connect your micro-usb cable to a USB port on a USB charger or PC to provide power. Connect the other end to the Nexys4's micro-usb connector. Place the JP3 jumper on pins 1 and 2 to select USB power. Use the switch to power up the Nexys4.

# External Power Supply

The MEGA65 core can consume a lot of power, and a standard USB port could potentionally be too little for the Nexys4 board. In particular, writing to the SD card might hang or perform odd behaviour. Therefore you should consider a 5V power supply.

Digilent sell a power supply for the Nexys4 board, and we recommend you use this to ensure you avoid the risk of damage to your Nexys4 board. The chosen power supply should be center positive, 2.1mm internal diameter plug, and should deliver 4.5VDC to 5.5VDC rated at least 1 Amp.

Connect the power supply cable to the supply plug of the Nexys4. Place the JP3 jumper on pins 2 and 3 to select WALL power. Use the switch to power up the Nexys4.

# Other Jumpers and Switches

For your initial set up, we'd suggest you set the following jumpers on your Nexys4 board to these positions:

- JP1 - USB/SD
- JP2 - SD

This will assure that the bitstream files will get loaded from your SD card on start-up.

At some later stage, you may prefer to load the bitstream from the on-board QSPI flash, and at that point, you can revisit your JP1 jumper setting and adjust it to the QSPI position.

All 16 switches on the lower edge of the board must be set to the off position.

# Connections and Peripherals

A USB keyboard can be connected to the USB port. Only a keyboard that lacks a USB hub will work with the Nexys4 board. Generally, extremely cheap keyboards will work, while more expensive keyboards tend to have a USB hub integrated, and will not work. You may need to try several keyboards before you find one that works.

You can connect a VGA monitor to the VGA port.

The mono audio-out jack can be connected to the line-in of an amplifier.

# Communicating with your PC

There may be occasions where you wish to communicate with your Nexys4 board from your PC, in order to perform activities such as:

- Flash your QSPI flash chip via Vivado

- Upload bitstream files directly from your PC (via m65 tool)
- Make use of support tools such as M65Connect, m65, mega65_ftp, m65dbg, etc

On such occasions, you will need to connect your micro-usb cable up to your PC.

# Onboard buttons

The "CPU RESET" button will reset the MEGA65 when pressed, while the "PROG" button will cause the FPGA itself to reload the MEGA65 core. The main difference between the two is that CPU RESET is faster, and does not clear the contents of memory, while the FPGA button is slower, and does reset the contents of memory.

Two of the five buttons in the cross arrangement can also be used: BTND acts as though you have pressed RESTORE, while BTNC will trigger an IRQ, as though the IRQ line had been pulled to ground.

# KEYBOARD

The keyboard layout is positional rather than logical. This means that keys in similar positions to the keys on a C65 keyboard will have similar function. This relationship assumes that your USB keyboard uses a US keyboard layout.

To help you locate what the various MEGA65 keys are mapped to, the MEGA65 has a built-in virtual keyboard test feature. This can be accessed in two ways.

The easiest way is to keep ALT held down in while switching on the Nexys4, or resetting the Nexys4 with the "PROG" button. The configure menu will be presented and by pressing 3, the virtual keyboard will be presented on a black background.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RUN STOP | | ESC | ALT | ASC DIN | NO SCRL | | F1 F2 | F3 F4 | F5 F6 | F7 F8 | | F9 F10 | F11 F12 | F13 F14 | HELP | |
| ~ _ | ! 1 | " 2 | # 3 | $ 4 | % 5 | & 6 | ´ 7 | ( 8 | ) 9 | ` 0 | + | - | GBP | CLR HOME | INST DEL | |
| TAB | Q | W | E | R | T | Y | U | I | O | P | @ | * | PI ~^ | RESTORE | | |
| CTRL | SHFT LOCK | A | S | D | F | G | H | J | K | L | [ {: | ] }; | = | RETURN | | |
| M= | SHIFT | Z | X | C | V | B | N | M | < , | > . | ? / | SHIFT | UP | | | |
| | | SPACE | | | | | | | | | | | LEFT | DOWN | RGHT | |

Pressing a key on the USB keyboard will show the highlighted key on the virtual keyboard to help you identify the key mapping.

The other way to access the virtual keyboard is from within the MEGA65. Hold M and press TAB to access the Matrix Mode Debugger. From here, enter the following:

```
s ffd3615 ff
```

This will open a semi-transparent virtual keyboard at the top of the screen. Alternatively:

```
s ffd3615 ff ff
```

This will open a semi-transparent virtual keyboard in the centre of the screen.

Hold M and press TAB to exit Matrix Mode Debugger and return to the MEGA65.

# Some key mappings with a USB keyboard

RESTORE is mapped to the PAGE UP key.

RUN STOP is mapped to ESC.

# PREPARING MICROSDHC CARD

The MEGA65 requires an SDHC card of between 4GB and 64GB capacity. Some SDXC cards may work, however, this is not officially supported.

Preparation steps for the Nexys4 board's SD card share much in common with the steps needed for real MEGA65 hardware, and as such, it is worth having a look over *the MEGA65 Book*, Configuring Your MEGA65 (section 4) if you ever need details.

So in this section, we'll provide more details on the distinctive steps, and be more brief on the common steps.

One point of distinction between the Nexys board and the real MEGA65 hardware is that the latter already has a default bitstream/core provided, which permits you to format your SD card in the specific style required by the MEGA65.

For Nexys4 board owners however, you have no such default bitstream, so see the **MEGA65 Book** for more details on where the appropriate "nexys4.bit" or "nexys4ddr-widget.bit" files for your device can be downloaded from.

## Preparation Steps

The steps are:

- Format the SD card in a convenient computer using the FAT32 file-system. The MEGA65 and Nexys4 boards do not understand other file systems, especially the exFAT file system.
- Copy your bitstream file (with name ending in ".bit") onto the SD card.
- Insert the SD card into the SD card slot on the under-side of the Nexys4 board.
- Switch on the Nexys4 board.
- Enter the Utility Menu by holding ALT down on the USB keyboard you have connected to the Nexys4 board.
- Enter the FDISK/FORMAT tool by pressing 2 when the option appears on the MEGA65 boot screen.
- Follow the prompts in the FDISK/FORMAT program to again format the SD card for use by the MEGA65.

  The FDISK tool will partition your SD card into two partitions and format them.

  - One is type $41 = MEGA65 System Partition, where the save slots, configuration data and other files live.
    (This partition is invisible in i.e. Win PCs).
  - The other partition with type $0C = VFAT32, where KERNAL, support files, games, and so on, will be copied to later.
    (This partition is visible on i.e. Win PCs).
- Once formatting is complete, switch off the Nexys4 board and remove the microSDHC card from the Nexys board and put it back into your PC
- This time, copy the following items onto the SD card:
  - The bitstream file
  - The extracted files from within either the "**SD essentials.rar**" or "**SD essentialsNoROM.rar**" file that you downloaded from the MEGA65 filehost. (See *the MEGA65 Book*, **??** (**??**) for more details).
  - If you have sourced your own preferred ROM file (e.g. "**911001.BIN**"), copy it onto the SD card also, and rename it to "**MEGA65.ROM**" (uppercase is essential).
  - Any .D81 disk image files you wish to make use of.
    * Note that if a file named MEGA65.D81 is added to the SD card, it will be mounted automatically on startup.
    * Make sure that all .D81 files have names that fit the old DOS 8.3 character limit, and are upper case. This restriction will be removed in a future release.
- Remove the SD card and reinsert it into your Nexys4 board.
- Power the Nexys4 board back on. The MEGA65 should boot within 15 seconds.
- On first start up, you will find yourself at the on-boarding screen, of which more details can be found in *the MEGA65 Book*, Configuring Your MEGA65 (section 4)
  .

Congratulations. Your MEGA65 has been set up and is ready to use.

Please note that the above method of copying the bitstream file to the SD card means that the bitstream is loaded into the Nexys FPGA each time on boot - which takes around 13 seconds for the system to start. The bitstream can also be flashed using Vivado software into the QSPI flash to deliver a boot up time of 0.3 seconds.

For more detailed information on preparing and configuring your MEGA65, please refer to *the MEGA65 Book*, Configuring Your MEGA65 (section 4) chapter.

# LOADING THE BITSTREAM FROM QSPI

While loading the bitstream from the SD card is the suggested (and well-trodden) path this document has chosen, of late, more nexys4 users have been exploring the alternative pathway of loading the bitstream from the QSPI flash. Some potential reasons they have chosen this pathway are:

- Faster loading times (0.3 seconds versus 13 seconds)
- Some people were interested in the possibility of flashing multiple cores onto their QSPI (via steps described in the *the MEGA65 Book*, How a MEGA65 Can Be Upgraded (section 5) Chapter)
- Some people have experienced niggling issues with the SD card pathway, such as:
  - System unable to reboot from on-boarding screen
  - System unable to reboot from freeze-menu after switching between PAL/NTSC

In time, if this proves to be a more popular pathway, we can revise our documentation here to suit it. Here are some steps in brief.

## Preparation Steps

For users that want to try this pathway, you will need to adjust the JP1 jumper setting to use QSPI and then follow the steps in the Flashing the FPGAs and CPLDs in the MEGA65 chapter in relation to Installing Vivado and Flashing the main FPGA using Vivado.

Be forewarned that the installation of Vivado is a lengthy process (both in terms of download time, and installation time).

Once you have flashed Slot0 of your QSPI chip via Vivado, you can then follow the steps described in *the MEGA65 Book*, Configuring Your MEGA65 (section 4) to perform the custom SD card formatting, installing of ROM and support files and on-boarding.

# WIDGET BOARD

For Nexys board owners (all models), you may be interested in adding a widget board to your nexys device, in order to allow you to connect to:

- a genuine C64 or C65 keyboard

- 2 DB-9 joysticks
- Paddles or a 1351 Mouse
- Cartridges (not functioning as yet)

The widget board connects to your nexys board via its PMOD connectors.

It presently is only available as an unpopulated PCB, purchasable from here:

- https://oshpark.com/shared_projects/Y37xg9N7

The firmware, pcb diagram, schematic diagram and bill of materials can be found within the following github project:

- https://github.com/sy2002/DM65PIC

You will need to purchases parts from the bill of materials separately and populate the board yourself.

You may find this forum64.de thread of interest, if you would like to read on the experiences of others that undertook this process, or if you have questions of your own to ask:

- https://www.forum64.de/index.php?thread/90465-mega-65-ports-add-on-card-still-available-if-where

The pcb (c65Keyb.brd) and schematic (c65Keyb.sch) files can be found within the 'eagle/' subfolder of the DM65PIC project, and can be viewed via the free tool 'AutoDesk Eagle', available here:

- https://www.autodesk.com/products/eagle/free-download

Here are some photos of the widget board in use:

CONTROL
RUN STOP
SHIFT LOCK
SHIFT
SHIFT
RETURN
RESTORE
CLR HOME
INST DEL
CRSR
CRSR

For convenience, the pcb and schematics diagrams for the widget board have also been provided in Chapter/Appendix C on page 289.

Some additional backstory notes for the board are:

- The widget board is originally meant to sit inside a c65 case
- Expansion port is not usable and a bit needs to be cut out for c64 case
- Would be reasonable to adapt it for c64 case use (which definitely will be the more regular use)

# PMOD-TO-JOYSTICK ADAPTER

As an alternate (and cheaper) option for those that just want to add a DB9 joystick port via the PMOD connectors, a user from the community, TheChief, has devised a means to do so. More information can be found in the following Discord thread:

- https://discord.com/channels/719326990221574164/903079038015389716/929369283119685643

| Joystick action | PMOD-JA Connections | DB9 Connections |
|---|---|---|
| Fire | jahi(9) | pin 6 |
| Up | jalo(1) | pin 1 |
| Left | jalo(2) | pin 3 |
| Down | jahi(7) | pin 2 |
| Right | jahi(8) | pin 4 |

# USEFUL TIPS

The following are some useful tips for getting familiar with the MEGA65:

- Press & hold [Commodore] (or the Commodore key if using a Commodore 64 or 65 keyboard) during boot to start up in C64-mode instead of C65-mode
- Press & hold [RUN STOP] during boot to enter the machine language monitor, instead of starting BASIC.
- Press [RESTORE] for approximately 1/2 - 1 second to enter the MEGA65 Freeze Menu. From this menu you have convenient tools to change the CPU speed, switch between PAL & NTSC video mode, change Audio settings, manage freeze-states, select D81 disk images, examine and modify memory of the frozen program, among other features. This is in many ways the heart of the MEGA65, so it is well worth exploring and getting familiar with.
- Type `POKE0,65` in C64-mode to switch the CPU to full speed (40MHz). Some software may behave incorrectly in this mode, while other software will work very well, and run many times faster than on a C64.
- Type `POKE0,64` in C64-mode to switch the CPU to 1MHz.
- Type `SYS58552` in C64-mode to switch to C65-mode.
- Type `GO64` in C65-mode and confirm, by pressing `Y`, to switch to C64-mode, which is the same as on a C128.
- The C65 ROM makes device 8 the default, so you can normally leave off the **,8** from the end of LOAD and SAVE commands.
- Pressing [SHIFT] + [RUN STOP] from either C64 or C65-mode will attempt to boot from disk.

Have fun! The MEGA65 has been lovingly crafted over many years for your enjoyment. We hope you have as much fun using it as we have had creating it!

The MEGA Museum of Electronic Games & Art welcomes your feedback, suggestions and contributions to this open-source digital heritage preservation project.

# PART VIII

## APPENDICES

# APPENDIX A

# Flashing the FPGAs and CPLDs in the MEGA65

- Suggested PC specifications
- Warning
- Installing Vivado
- Installing the FTDI drivers
- Flashing the main FPGA using Vivado
- Flashing the CPLD in the MEGA65's Keyboard with Lattice Diamond

- **Flashing the MAX10 FPGA on the MEGA65's Mainboard with INTEL QUARTUS**

The MEGA65 is an open-source and open-hardware computer. This means you are free, not only to write programs that run on the MEGA65 as a finished computer, but also to use the re-programmable chips in the MEGA65 to turn it into all sorts of other things.

If you just want to install an upgrade core for the MEGA65, or a core that lets you use your MEGA65 as another type of computer, you probably want to look in *the MEGA65 Book*, How a MEGA65 Can Be Upgraded (section 5) instead.

This chapter is more intended for people who want to help develop cores for the MEGA65. This chapter may also be of interest to Nexys4 board owners that are interested booting their devices from the on-board QSPI flash memory chip (rather than a bitstream file on the SD card). This will require flashing an .mcs file onto their board's QSPI chip, so as to provide an initial bistream in the 'Slot 0' position.

These re-programmable chips are called Field Programmable Gate Arrays (FPGAs) or Complex Programmable Logic Devices (CPLDs), and can implement a wide variety of circuits. They are normally programmed using a language like VHDL or Verilog. These are languages that are not commonly encountered by most people. They are also quite different in some ways to "normal" programming languages, and it can take a while to understand how they work. But with some effort and perseverance, exciting things can be created with them.

# SUGGESTED PC SPECIFICATIONS

Be prepared to install many gigabytes of software on a Linux or Windows PC, before you will be able to write programs for the FPGAs and CPLDs in the MEGA65. Also, "compiling" complex designs can take up to several hours, depending on the speed and memory capacity of your computer. We recommend a computer with at least 12GB RAM (preferably 16GB) if you want to write programs for FPGAs and CPLDs. On the other hand, if all you want to do is load programs onto your MEGA65's FPGAs and CPLDs that other people have written, then most computers running a recent version of Windows or Linux should be able to cope.

- OS: Linux or Windows
- CPU Speed: As fast as you can get your hands on!
- Number of cores: Ideally, 8 or more, as the free license of Vivado can make use of a max of 8 cores.
- Hard disk space: Have about 70GB or more. The exact amount used depends on how many components within Vivado you install (bear in mind that the full install file is about 50GB in itself)

- Memory: minimum of 12GB (ideally, have more, to play it safe)

# WARNING

Before we go any further, we do have to provide a warning about reprogramming the FPGAs and CPLDs in the MEGA65. Re-programming the MEGA65 FPGA can potentially cause damage, or leave your MEGA65 in an unresponsive state from which it is very difficult to recover, i.e., "bricked". Therefore if you choose to open your MEGA65 and reprogram any of the FPGAs it contains, it is no longer possible to guarantee its correct operation. Therefore, we cannot reasonably honour the warranty of the device as a computer. You have been warned!

# INSTALLING VIVADO

Installation of Vivado is required to flash the QSPI flash memory within your MEGA65 target device, whether it be a MEGA65 R2/R3/R3A/R4, Nexys4/Nexys4DDR/NexysA7, MEGAphone or other.

Vivado is also the tool used to perform compilation (synthesis, as it is preferably called) of FPGA bitstreams.

To get started, connect to `https://www.xilinx.com/support/download.html`

Select 2020.2 version

NOTE : Some users still have success with using older versions, as the main aim here is to install a version that supports the FPGA of your target hardware.

I.e., the Artix7 100T (for Nexys and R2) or 200T (R3/R3A/R4).

Click on Xilinx Unified Installer 2020.2: Windows Self Extracting Web Installer EXE - 248.44MB

You will be asked to create an account in order to sign in and be able to download the installation program.

Your credentials will also be requested when doing the installation.

After having signed in, you have to provide some personal information and then click on Download

Execute the installer as Administrator (Xilinx_Unified_2020.2_1118_1232_Win64.exe).

Click on Allow Access.

Click on Next.

Enter your credentials and click on Next.

Select Vivado and click on Next.

Select "Vivado HL WebPACK" and click on "Next"

We'd suggest selecting only the "**7 Series**" devices, as our chosen FPGA is within this series, and de-selecting the other series will save you about 6GB in download size. Then click on "Next"

Warning: As stated, disconnect any USB cable that would be connected to your PC from the Nexys board.

Agree with all the End User Licence Agreement and Terms and conditions and click on "Next".

Choose the location where you want to install the software and click on "Next".

Warning : You are about to download 20GB of software and you need 70GB to perform the installation.

Click on "Yes"

Click on "Install"

Wait for the installation to complete. At the very end of the installation you will be asked if you want to install Xilinx device software.

Click on "Install"

Let the installation complete.

The installation is completed. Click on "OK"

You end up with the following icons on your desktop:

Note: Ubuntu users might need to install one missing dependency with: `sudo apt install libtinfo5`

Launch Vivado 2020.2

Click on "Help"->"Obtain a licence Key"

This launches the Vivado licence manager

Select "Get Free ISE WebPACK, ISE/Vivado IP or PetaLinux Licenses"

Click on "Connect Now"

Connect with the user account you have created to be able to download the Vivado software. If you were not already connected to Xilinx website, this will take you to the main webpage. Go back in the licence manager (which is not closed)

Click again on "Connect Now" (ensure "Get Free ISE WebPACK, ISE/Vivado IP or PetaLinux Licenses" is still selected)

You then register your personal information on the Vivado website, and click on "Next".

Select "ISE WebPACK Licence" and "Vivado Design Suite: HL WebPACK 2015 and Earlier License"

Then click on "Generate Node-Locked Licence"

| Product | Type | License | Available Seats | Status | Subscription End Date |
|---|---|---|---|---|---|
| Xilinx add-on for Matlab and Simulink, 90 Day Evaluation | Certificate - Evaluation | Node | 1/1 | Current | 90 days |
| ISE Embedded Edition License | Certificate - No Charge | Node | 1/1 | Current | None |
| Vivado Design Suite: 30-Day Evaluation License | Certificate - Evaluation | Node | 1/1 | Current | 30 days |
| SDSoC Environment, 60 Day Evaluation License | Certificate - Evaluation | Node | 1/1 | Current | 60 days |
| SDAccel OpenCL Development Environment: 30 Day Node Locked Evaluation Lice... | Certificate - Evaluation | Node | 1/1 | Current | 30 days |
| Vivado Design Suite: HL WebPACK 2015 and Earlier License | Certificate - No Charge | Node | 1/1 | Current | None |
| ISE WebPACK License | Certificate - No Charge | Node | 1/1 | Current | None |
| Xilinx MicroBlaze/All Programmable SoC Software Development Kit – Standalone | Certificate - No Charge | Node | 1/1 | Current | None |
| PetaLinux Tools License | Certificate - Evaluation | Node | 1/1 | Current | 365 days |
| Vivado HLS Evaluation License | Certificate - Evaluation | Node | 1/1 | Current | 30 days |

Click on "Next"

Click on "Next"

Check your email box : You should have received an email from Xilinx, Inc. with a licence file attached and named "Xilinc.lic".

Retrieve this file on your PC and keep it in safe place.

Congratulations

Your new license file has been successfully generated and e-mailed to [redacted] You can also view the license file under the Manage Licenses tab.

**Please add this sender (xilinx.notification@entitlenow.com) to your address book.**

**License File Details**
Node License
Host ID: ANY

**Products**
ISE WebPACK License (No Charge): 1 seats

Go back to the licence manager (which is still running).

Set "Load License" and click on "Copy License"

Browse to the location where you saved "Xilinc.lic" file, select it and click on "Open".

Click on "OK" and close the Vivado licence manager.

Your Vivado software is registered and you can now use it.

# INSTALLING THE FTDI DRIVERS

The FTDI drivers are needed in order for your PC to communicate with the hardware's JTAG port and serial comms port (note that the single physical USB connection made to your PC actually provides these two ports).

# Linux drivers

Some Linux users have reported that they have found the FTDI drivers to be installed within their Linux distributions out-of-the-box, while others have found they needed to run this extra command after installing Vivado:

```
cd /opt/Xilinx/Vivado/2018.3/data/xicom/cable_drivers/lin64/install_script/install_driv
sudo ./install_drivers
```

# Windows drivers

Download the following archive to install the drivers:

- https://www.ftdichip.com/Drivers/CDM/CDM21228_Setup.zip

Unzip the file **CDM21228_Setup.zip**, you get the file **CDM21228_Setup.exe**.

Warning:
Before installing the drivers, it is imperative to switch off the Nexys4 board and to disconnect the USB cable from the PC.

Review the devices already installed before the installation:

Run the file CDM21228_Setup.exe as administrator:

Confirm that you want to run the program.

Click on "Extract".

Click on "Next >"

Accept the agreement and click on "Next >".

The installation of the drivers starts.

Click on "Finish".

Connect the USB cable to a USB port on the PC without turning on the Nexys4 board.

Connecting the USB cable triggers the appearance of new devices.

- An additional COM port has been installed: This is the COM port that will be used to communicate with the Nexys4 board.
- An additional USB composite device has been installed.

- Two USB serial converter devices have been installed.

At this point the Nexys4 board has still not been powered up.

For more information about the installed drivers, you can download the corresponding documentation:

- https://ftdichip.com/wp-content/uploads/2020/08/AN_396-FTDI-Drivers-Installation-Guide-for-Windows-10.pdf
- https://ftdichip.com/wp-content/uploads/2021/01/AN_119_FTDI_Drivers_Installation_Guide_for_Windows7.pdf

# FLASHING THE MAIN FPGA USING VIVADO

Firstly, to clarify that when we say 'flashing the FPGA', in reality, what we mean is that we are flashing the QSPI flash memory chip that the FPGA makes use of upon startup in order to quickly load the bitstream from.

The diagram below shows two common pathways that the FPGA can load bitstreams at startup:

- We can first flash a bitstream/core-file onto the QSPI flash memory chip, and the FPGA can load this quickly at power-up. Flashing the QSPI is quite slow, but the reward of a fast boot-up time is an advantage.

- Nexys board users can drop a bitstream file onto our SD card and let the FPGA load it (somewhat more slowly) from there at power-up. This allows them to swap/upgrade bitstreams quickly without the need for a TE0790-03 JTAG programming module. This feature is planned to be added to the MEGA65 later, though.

In this section, we describe the pathway that makes use of the QSPI.

Many of the following steps in this section are applicable not only to MEGA65 R2/R3/R3A/R4 owners, but Nexys4 board owners too. There are a few points of distinction along the way that readers will be made aware of.

If you choose to proceed, you will need a functioning installation of Xilinx's Vivado software, and the FTDI drivers installed, as described in the earlier sections.

You will also need to download or build an .mcs bitstream file (and optional .prm checksum verification file) that you intend to flash onto the QSPI chip via Vivado. See the **MEGA65 Book** for more details on where such files can be downloaded.

For MEGA65 R2/R3/R3A/R4 owners:

You will need a TE0790-03 JTAG programming module. It is also necessary to have dip-switches 1 and 3 in the ON position and dip-switches 2 and 4 in the OFF position on the TE-0790. With your MEGA65 disconnected from the power, the TE-0790 must be installed on the JB1 connector which is located between the floppy data cable and the audio jack. The gold-plated hole of the TE-0790 must line up with the screw hole below. The mini-USB cable will then connect on the side towards the 3.5" floppy drive. The following image shows the correct position: The TE0790 is surrounded by the yellow box, and the dip-switches by the red box. Dip-switch 1 is the one nearest the floppy data cable.

For Nexys4 board owners:

Simply connect your micro-usb cable between your Nexys4 board and your PC via the port labeled 'PROG UART' (J6), as shown:

Also, set J1 jumper to the QSPI position:

- Connect your non-8-bit computer to the FPGA programming device using the appropriate USB cable.
- Switch the MEGA65 computer ON.
- Open Vivado.

Step 1a: Create a new Vivado project with "File", "Project", "New...".
NOTE: On future occasions that you need to flash the QSPI, just re-open this project (no need to create a new project each time).

Step 1b: The 'New Project' wizard appears. Click on "Next":

Step 1c: Name your project and choose the location you like, then click on "Next":

Step 1d: Keep the default selected options and click on "Next":

Step 1e: Do not add any sources, keep the default selected options and click on "Next":

Step 1f: Keep the default selected options and click on "Next":

Step 1g: Click on "Finish":

Step 2: In the left column, select "Open Hardware Manager" at the very bottom.

Step 3: Connect to the FPGA:
Under "Open Hardware Manager", choose "Open Target", then "Auto Connect".

Step 4: Wait a moment, "Connecting to server..." should automatically close without dropping an error to the console.

Step 5: Under "Open Hardware Manager", choose "Add Configuration Memory Device", then:

- For MEGA65R3/R3A/R4: "xc7a200t_0"
- For Nexys4 and MEGA65R2: "xc7a100t_0".

Step 6a: Select Memory Part:

In the newly opened dialogue:

- For MEGA65R2/R3: type "S25fl256s" (without quotes), then select "s25fl256sxxxxxxx0-spi-x1_x2_x4" (the upper one) and click "OK".
- For MEGA65R3A/R4: Vivado cannot flash the larger flash part on these boards. Use the flash menu on the MEGA65.
- For Nexys4: type "S25fl128s" (without quotes), then select "s25fl128sxxxxxxx0-spi-x1_x2_x4" (the upper one) and click "OK".

Step 6b: Click on "OK" to confirm you want to program the configuration memory device now.

Step 6c: If you do not see such a popup, or wish to reprogram the QSPI on a future occasion, in "Hardware" window, right click on the memory configuration and select "Program Configuration Memory Device":

Step 7: Set programming options:
In the next dialogue, set the "Configuration file" to the path of your ".mcs" bitstream file. You can also optionally set the "PRM file" field to the path of your ".prm" file. Leave all other parameters as they are (see screenshot below).

Step 8: Patiently wait for the programming to finish. This can take several minutes as the Vivado software erases and then reprograms the flash memory that is used to initialise the FPGA on power-up.

Step 9: If your screen looks like the screenshot below, your new bitstream has been successfully flashed into Slot0 of your QSPI flash memory!

Step 10: If you want to reflash the FPGA, you might find the "Add Configuration Memory Device" option in step 5 greyed out. Instead, select "s25fl256sxxxxxxx0-spi-x1_x2_x4" in the "Hardware" window, press right mouse button and select "Program Configuration Memory Device" to flash.

# FLASHING THE CPLD IN THE MEGA65'S KEYBOARD WITH LATTICE DIAMOND

If you choose to proceed, you will need a TE0790-03 JTAG programming module and a functioning installation of Lattice Diamond Programmer software. This can be done on either Windows or Linux, but in both cases you will need to install any necessary USB drivers. It is also necessary to have dip-switches 1 and 3 in the ON position and dip-switches 2 and 4 in the OFF position on the TE-0790. With your MEGA65 disconnected from the power, the TE-0790 must be installed on the JB1 connector, which is located between the floppy data cable and the audio jack. The gold-plated hole of the TE-0790 must line up with the screw hole below. The mini-USB cable will then connect on the side towards the 3.5" floppy drive. The following image shows the correct position: The TE0790 is surrounded by the yellow box, and the dip-switches by the red box. Dip-switch 1 is the one nearest the floppy data cable.

FLOPPY
TE0790-03
S2
S1
J14
J17
J21
U32
U33
S3
J16
J13
P1
P2
U7
U6
U9
J4
J0
D3
D2
D1
D0

On the PCB of a R2/R3/R3A/R4 MEGA65 mainboard, dip switch 1 (the one nearest to the user sitting in front of the machine) must be in the ON position. The other switches must be OFF. The keyboard will go into "ambulance mode" (blue flashing lights) when set correctly.

Connect your non-8-bit computer to the FPGA programming device using a mini-USB cable. Switch the MEGA65 computer ON. Open the Diamond Programmer which can be downloaded from the Internet.

Step 1: Open DIAMOND PROGRAMMER:
Select "Create a new project from a JTAG scan". If entry under "Cable:" is empty, click "Detect Cable".

Step 2: Create a new project:
If dialog "Programmer: Multiple Cables Detected" appears, select the first entry ("Location 0000") and click "OK".

Step 3: Select cable:
You have now created a new project which should display "MachXO2" under "Device Family" and "LCMXO2-1200HC" under "Device"

Step 4: New Diamond Programmer project:
Choose "File" then "Open File" to load the Diamond Pprogrammer project with the MEGA65 keyboard firmware update.

Step 5: Open project:
Navigate into the folder with the extracted MEGA65 keyboard firmware files you have received and select the file ending with ".xcf".

Step 6: Select project file:
Click the three dots under "File Name" to set the correct path and find the file ending with ".jed".

Step 7: Choose correct path of .jed file:
Select the file ending with ".jed" and click "OK".

Step 8: Select .jed file:
Click on the icon with the green arrow facing down "PROGRAM", which looks similar to the Diamond Programmer program icon.

Step 9: Select cable:
After a moment the Output window should display "INFO - Operation: successful." and the "Status" cell should go green (does not always happen).

Step 10: Operation successful:
You have now successfully flashed the MEGA65 keyboard. If you wish you can now save the project for later use.

# FLASHING THE MAX10 FPGA ON THE MEGA65'S MAINBOARD WITH INTEL QUARTUS

If you choose to proceed, you will need a TEI0004 - Arrow USB Programmer2 module with TEI0004 driver installed and a functioning installation of Quartus Prime Programmer Lite Edition. This can be done on either Windows or Linux, but in both cases you will need to install any necessary USB drivers. With your MEGA65 disconnected from the power, the TEI0004 must be installed on the J17 connector, which is located between the floppy data cable and the ARTIX 7 FPGA on the Mainboard. The micro-USB port of the TEI0004 must face in the opposite direction of the HDMI and LAN sockets, towards the trap door. The following image shows the correct position.

On the PCB R2/R3/R3A/R4 MEGA65 mainboard, all dip switches must be in the OFF position. The main FPGA of the MEGA65 must not contain a valid bitstream. The easiest way to do this is if you have a TE0790 JTAG adapter: you can then use the m65 tool from a connected computer to begin sending a bitstream via JTAG, and then using control-C to abort the m65 tool before it can finish loading the bitstream.

MOULDS TEAM
MOULDS TEAM UG
Am Rinkenbühl 15
64807 Dieburg
7499011222A
NR2112
41220AC
DVI
VGA
ETHERNET
FLOPPY
J17
A26572
TE1000-02
J21
SD
J6
J3
S3
S5
U1
U3
U16
U18
U19
U10
J10
J9
D3
D2
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
D16
D17
D18
D19
D20
D21
D22
D23
D24
D25
D26
D27
D28
D29
D30
D31
D32
D33
D34
D35
D36
D37
D38
D39
D40
D41
D42
D43
D44

Connect your non-8-bit computer to the FPGA programming device using a micro-USB cable. Open Quartus Prime Programmer Lite Edition, which can be downloaded from the Internet.

Step 1: Open Quartus Prime Programmer Lite Edition:
Click the "Hardware Setup" button in the top left corner of the Quartus Prime Programmer window.

Step 2: Enter Hardware Setup:
In the newly appeared window under "Currently selected hardware" choose "Arrow-USB-Blaster". If "Arrow-USB-Blaster" does not appear, verify cable and drivers being correctly installed.

Step 3: Select Arrow USB-Blaster:
Click the "Add File" button from the left row and choose the latest ".pof" file. Then click "Open".

Step 4: Select Programming File:
Tick at least the three boxes under "Program/Configure". Also enabling all boxes under "Verify" and "Blank-Check" will make the process more reliable.

Step 5: Select Program/Configure Options:

While keeping the Reset-Button pressed, switch the MEGA65 computer ON. The keyboard will go into "ambulance mode" (blue flashing lights). If it does not, the main FPGA is not empty - restart the whole process.

Now click on "Start" in the left row of buttons. The progress bar in the top right corner should quickly go to 100 percent and turn green. You have now successfully updated your MAX10 FPGA.

If you receive an error message instead, make sure the main FPGA bitstream has been erased and that you did not release the reset-button on the MEGA65 beforehand. Switch off the MEGA65 and restart this step.

Step 6: Programming successful:

# APPENDIX B

# Trouble shooting

- Hardware
- Vivado
- mega65_ftp

# HARDWARE

## No Digital Video Output

**MEGA65 R4, R5, R6**

The digital video port has improved back-power isolation, to correct the "No lights when powering on" problem. A side effect of this is that some digital video cables incorrectly use only the shell of the connector to provide the ground signal, instead of using the ground signals from the pins of the connector. Long-story-short: If you don't get an image on the digital video port, you can try one of the following solutions:

- Try different digital video cables you have. This problem only occurs with certain brands of cables, especially thinner ones (see the image in Figure **??**), that we strongly suspect are missing the internal ground wires to save money. This is the best solution, as it is actually the cable that is the cause of the fault, by being non-compliant. Even if you use one of the following work-arounds, you should still go and find a better HDMI cable that solves the problem at the root-cause.
- Connect a VGA cable to the MEGA65 and to the same monitor. This will provide a ground connection through the VGA connector, and seemingly magically allow the HDMI to work.
- Connect a TE0790 JTAG adapter. One user reported that this also magically worked-around the problem.
- Bridge the digital video connector shell to the shell of the internal full-size SD card shell.
- Connect an splitter or repeater to the digital video port.

## No lights when powering on

If there are occasions when your MEGA65 display any lights when powering on, they relate to having certain Digital Video devices plugged in while the MEGA65 is off, that don't provide enough power for the keyboard's CPLD to be properly powered on, but enough to stop it properly resetting when the MEGA65 powers on. Removing the Digital Video cable and switching the machine off and on again fixes the issue.

# VIVADO

Figure B.1: Example of bad (top) and good (bottom) digital video cables. Note how the bad cable is suspiciously thinner than the good one. This is a sign that it likely is missing dedicated ground wires, and will cause problems on the MEGA65 R4 and later revisions. Any "normal thickness" cable should work fine, though. You don't need to buy one of those over-priced "oxygen-free unobtanium-plated hand-assembled in zero-gravity by world-famous artisans" cables. Those cables offer about as much practical benefit sprinkling some glitter or drawing a unicorn on one end, but with much less class.

# RAM requirements

```
INFO: [Synth 8-256] done synthesizing module 'ram32x1024' [/home/....]
INFO: [Synth 8-256] synthesizing module 'charrom' [/home/....]
   /opt/Xilinx/Vivado/2019.2/bin/loader: line 280: 2317 killed
   WARNING: [Vivado 12-8222] Failed run(s) : 'synth\_1'
      ERROR: Application Exception: failed to launch run 'impl\_1'
          due to failures in the following run(s):
      synth\_1
      These failed run(s) need to be reset prior to launching 'impl\
          _1' again.
```

This error is due to Vivado crashing because the machine doesn't have enough RAM for Vivado to run. Vivado requires at least 4GB to synthesise the MEGA65 target, but 8GB is better.

# MEGA65_FTP

## Missing Library

```
/usr/bin/ld: cannot find -lncurses
collect2: error: ld returned 1 exit status
Makefile:474: recipe for target 'bin/mega65_ftp' failed
make: *** [bin/mega65_ftp] Error 1
```

This error occurs when the ncurses library is missing from the computer when building the mega65_ftp program. To rectify this issue you will need to ensure that you install this dependency.

```
sudo apt-get install libncurses5-dev libncursesw5-dev
```

# APPENDIX C

# Schematics

- MEGA65 R3 Schematics
- MEGA65 R2 Schematics
- Nexys Widget Board Schematics

# MEGA65 R3 SCHEMATICS

U_HDMI
HDMI.SchDoc

U_Ethernet
Ethernet.SchDoc

U_LED_SW_BUT
LED_SW_BUT.SchDoc

U_SYS_MAX10_CTRL
SYS_MAX10_CTRL.SchDoc

U_SOUND
SOUND.SchDoc

U_B13
B13.SchDoc

U_B14
B14.SchDoc

U_B15
B15.SchDoc

U_B16
B16.SchDoc

U_B34
B34.SchDoc

U_FPGA-CFG
FPGA-CFG.SchDoc

U_FPGA-MGT
FPGA-MGT.SchDoc

U_FPGA-PWR
FPGA-PWR.SchDoc

U_JOY
JOY.SchDoc

U_KEYBOARD
KEYBOARD.SchDoc

U_POWER
POWER.SchDoc

U_PowerMain
PowerMain.SchDoc

U_VGA
VGA.SchDoc

U_EXT_HEADER
EXT_HEADER.SchDoc

U_EXP_Slot
EXP_Slot.SchDoc

U_Floppy
Floppy.SchDoc

PM1 FIDU-DOT - small
PM2 FIDU-DOT - small
PM3 FIDU-DOT - small
PM4 FIDU-DOT - small
PM5 FIDU-DOT - small
PM6 FIDU-DOT - small

Serial1
Serial
Serialnumber 6,3 x 6.3mm

LOGO1
TE Logo PRINT Layer
LOGO PRINT

GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm

| | | | |
|---|---|---|---|
| Design drawn by: | IG | Title: | MEGA65 |
| Checked by: | MR | | |
| Assembly variant: | [No Variations] | A4 | Number: TE0765 [No Variations] Rev. 03 |
| Created by: | VariantCreatedBy | | |
| Modified by: | VariantModifiedBy | Date: 2020-06-29 | Copyright: Trenz Electronic GmbH Page 1 of 23 |
| Modified at: | VariantDateModification | Filename: | TE0765.SchDoc |

trenz electronic

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 2 of 23 |
| Filename: EXT_HEADER.SchDoc | | | |

| | | | |
|---|---|---|---|
| Title: MEGA65 | | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 3 of 23 |
| Filename: EXP_Slot.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 4 of 23 |
| Filename: B13.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 5 of 23 |
| Filename: B14.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 6 of 23 |
| Filename: B15.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 7 of 23 |
| Filename: B16.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 8 of 23 |
| Filename: B34.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 9 of 23 |
| Filename: FPGA-CFG.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 10 of 23 |
| Filename: FPGA-MGT.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 11 of 23 |
| Filename: FPGA-PWR.SchDoc | | | |

U32A
BANK 1A
BANK 1B
10M08SAU169C8G
U32B
BANK 2
U32C
BANK 3
U32E
BANK 6
U32D
BANK 5
U32F
BANK 8
U32G
J21
Pin Header 2.54 2Row 14Pins
LED Green LTST-C191KGKT
LED Red LTST-C191KRKT
3.3V_REF
MPZ1608S221A
M10_nCONF
M10_CONF_DONE
M10_nSTATUS
Title:
MEGA65
A4
Number:
TE0765
[No Variations]
Rev.
03
Date:
2020-06-29
Copyright: Trenz Electronic GmbH
Page 12 of 23
Filename:
SYS_MAX10_CTRL.SchDoc
trenz electronic

IEC - 1541 style Diskdrive
J6
61PC6F
SER_SRQ
SER_DATA
SER_RESET
SER_CLK
SER_ATN
GND
U15
U16
U27
U20
U21
U28
U22
U23
U24
U25
U26
NC7SZ126P5X
F_SER_CLK_EN
F_SER_CLK_O
F_SER_DATA_EN
F_SER_DATA_O
F_SER_SRQ_EN
F_SER_SRQ_O
SER_CLK
SER_DATA
SER_SRQ
JA_RIGHT
JA_LEFT
JA_DOWN
JA_FIRE
JA_UP
F_SER_CLK_I
F_SER_DATA_I
F_SER_SRQ_I
FA_RIGHT
FA_LEFT
FA_DOWN
FA_FIRE
FA_UP
R87
R168
R170
4K7
5V
3.3V
C179
10µF
C180
10µF
10V
J7
J3
JA_AY
JA_AX
JA_RIGHT
JA_LEFT
JA_DOWN
JA_FIRE
JA_UP
JB_AY
JB_AX
JB_RIGHT
JB_LEFT
JB_DOWN
JB_FIRE
JB_UP
R164
R165
R166
R167
R169
R171
R172
R173
R174
R175
R86
1M
5%
C135
100pF
1000V
X7R
C45
10µF
C72
10µF
5V_JOY
DB-9-M UNC 4-40 rivet, w/o bolts
U33
MP5010BDQ-LF-Z
VCC
NC
Enable/Fault
dv/dt
GND
SOURCE
I-Limit
R127
4K7
5V_JOY_PG
T2
2N7002,215
R122
DNP
R123
200K
1%
R125
200K
1%
C73
1µF
50V
X7R
R124
22R
1%
T1
2N7002,215
EN_5V_JOY_N
trenz electronic
Title:
MEGA65
A4
Number:
TE0765
[No Variations]
Rev.
03
Date:
2020-06-29
Copyright: Trenz Electronic GmbH
Page 13 of 23
Filename:
JOY.SchDoc

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| | A4 | Number: TE0765 [No Variations] | Rev. 03 |
| | Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | Page 14 of 23 |
| | Filename: KEYBOARD.SchDoc | | |

trenz electronic

| | | |
|---|---|---|
| Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | Page 15 of 23 |
| Filename: Floppy.SchDoc | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 16 of 23 |
| Filename: VGA.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 17 of 23 |
| Filename: HDMI.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 18 of 23 |
| Filename: Ethernet.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 19 of 23 |
| Filename: LED_SW_BUT.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 20 of 23 |
| Filename: PowerMain.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH / TT | | Page 21 of 23 |
| Filename: POWER.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 22 of 23 |
| Filename: SOUND.SchDoc | | | |

REVISION HISTORY

| REV | Description | |
|---|---|---|
| -03 | 1. Added a VRP resistor on bank 65;<br>2. LDO U33 is changed on ADP7102ACPZ;<br>3. Signal FPGA IO0 is connected on AE18 pin of FPGA;<br>4. Signal DBG_LED3 is connected on AD18 pin of FPGA;<br>5. Signal MIO13_25 connected to J1 pin 33 instead MIO25.<br>6. Resistor R84 is removed;<br>7. LED D1 moved on edge of PCB;<br>8. Added THT testpoints J4 on CPLD_JTAGEN, R76 was removed;<br>9. Signals B49_XX_X are renamed in B88_XX_X;<br>10. C241 is changed on 1nF;<br>11. Length of CLK signals on RFADC and RFDAC are adjusted;<br>12. Wrong connection on U8 is fixed (PCB);<br>13. Wrong connection PGOOD1 pin of U7 is fixed; | IG |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0835 [No Variations] | | Rev. 03 |
| Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | | Page 23 of 23 |
| Filename: Revision_Changes.SchDoc | | | |

208,05
195,30
179,51
159,03
131,20
102,50
85,00
85,00
26,00
20,00
7,50
20,00
5,5
4,889mm
25,00
34,94
59,50
31,50
23,00
32
214,00
10,50
POWER
DISK
VGA
HDMI
ETHERNET
micro SD
AUDIO
CARD
MOULDS TEAM
MOULDS TEAM UG
Am Rinkenbühl 15
76870 Kandel
KEYBOARD
JOYSTICK B
JOYSTICK A
MEGA65
SPK R
SPK L
RST
GROVE
SD
TLOP65-03

# MEGA65 R2 SCHEMATICS

U_HDMI
HDMI.SchDoc

U_Ethernet
Ethernet.SchDoc

U_LED_SW_BUT
LED_SW_BUT.SchDoc

U_SYS_MAX10_CTRL
SYS_MAX10_CTRL.SchDoc

U_SOUND
SOUND.SchDoc

U_B13
B13.SchDoc

U_B14
B14.SchDoc

U_B15
B15.SchDoc

U_B16
B16.SchDoc

U_B34
B34.SchDoc

U_FPGA-CFG
FPGA-CFG.SchDoc

U_FPGA-MGT
FPGA-MGT.SchDoc

U_FPGA-PWR
FPGA-PWR.SchDoc

U_JOY
JOY.SchDoc

U_KEYBOARD
KEYBOARD.SchDoc

U_POWER
POWER.SchDoc

U_PowerMain
PowerMain.SchDoc

U_VGA
VGA.SchDoc

U_EXT_HEADER
EXT_HEADER.SchDoc

U_EXP_Slot
EXP_Slot.SchDoc

U_Floppy
Floppy.SchDoc

PM1 FIDU-DOT - small
PM2 FIDU-DOT - small
PM3 FIDU-DOT - small
PM4 FIDU-DOT - small
PM5 FIDU-DOT - small
PM6 FIDU-DOT - small

Serial1
Serial
Serialnumber 6,3 x 6.3mm

S/N1
1 0R 2
Serialnumber

GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| | A4 | Number: TE0765 Default | Rev. 02 |
| | Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | Page1 of 23 |
| | Filename: TE0765.SchDoc | | |

Grove CONN
D1
WE-TVS-824013
GROVE_SCL0
GROVE_SDA0
3.3V
GND
J18
110990037
R128
R129
4K7
CPLD CONFIG
J17
SMD-254-9132-14-10
CPLD_JTAG
M_TCK
M_TDO
M_TMS
M_TDI
R156
C137
470nF
TE0790-Base SMT
JB1
TE_JTAG
TE_TCK
TE_TDO
TE_TDI
TE_TMS
TE_UART_RX
TE_UART_TX
R157
C138
R158
10K
PMOD CONN
D2
D3
B35_L2_N
B35_L2_P
B35_L1_P
B35_L1_N
B35_L5_N
B35_L5_P
B35_L3_P
B35_L3_N
P1
PMod 2x6 SMD Host Socket 90°
PMOD1_VCC
C145
10µF
10V
C146
470nF
P2
PMod 2x6 SMD Host Socket 90°
PMOD2_VCC
C162
C163
D4
D11
U35
AP2196SG-13
C164
R146
R147
10K
PMOD2_FLG
PMOD1_FLG
PMOD2_EN
PMOD1_EN
IN
OUT1
OUT2
FLG1
FLG2
EN1
EN2
trenz electronic
Title:
MEGA65
A4
Number:
TE0765
Default
Rev.
02
Date:
2019-03-11
Copyright: Trenz Electronic GmbH
Page2 of 23
Filename:
EXT_HEADER.SchDoc

Title: MEGA65

A4 | Number: TE0765 Default | Rev. 02

Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | Page3 of 23

Filename: EXP_Slot.SchDoc

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page4 of 23 |
| Filename: B13.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page5 of 23 |
| Filename: B14.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page6 of 23 |
| Filename: B15.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page 7 of 23 |
| Filename: B16.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page8 of 23 |
| Filename: B34.SchDoc | | | |

U1G
BANK0
XC7A100T-2FGG484C
VCCO_0
VCCO_0
VCCBATT_0
CCLK_0
M0_0
M1_0
M2_0
DONE_0
PROGRAM_B_0
CFGBVS_0
TDI_0
TDO_0
TCK_0
TMS_0
INIT_B_0
VCCADC_0
VREFP_0
VREFN_0
VP_0
VN_0
DXP_0
DXN_0
GNDADC_0
C170
4.7µF
SPI-SCK
BOOTMODE = MASTER SPI
FPGA_DONE
FPGA_PROG_B
FPGA_TDI
FPGA_TDO
FPGA_TCK
FPGA_TMS
FPGA_INIT
FPGA_JTAG
R68
4K87
1%
R3
1K
R8
4K87
1%
CFG
V_P
V_N
C122
10nF
16V
X7R
AVCC
AGND
GND
3.3V
SPIFLASH
U5A
S25FL256SAGBHI20
CS
CLK
DI/IO0
DO/IO1
HOLD/RESET/IO3
WP/IO2
VCC
GND
SPI-CS
SPI-SCK
SPI-DQ0
SPI-DQ3
SPI-DQ2
SPI-DQ1
R7
4K87
1%
C14
100nF
L3
BKP0603HS121-T
L2
BKP0603HS121-T
1.8V
C23
470nF
U5B
S25FL256SAGBHI20
NC
S2
SPUJ191500
R134
49R9
C75
10nF
16V
X7R
RST_BTN
R145
4K87
1%
R148
4K87
1%
CPLD_JTAGEN/RESET_BTN
T6
2N7002,215
J20
JUMPER2.54-2
trenz electronic
Title:
MEGA65
A4
Number:
TE0765
Default
Rev.
02
Date:
2019-03-11
Copyright: Trenz Electronic GmbH
Page9 of 23
Filename:
FPGA-CFG.SchDoc

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page10 of 23 |
| Filename: FPGA-MGT.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page 11 of 23 |
| Filename: FPGA-PWR.SchDoc | | | |

U32A
BANK 1A
BANK 1B
10M08SAU169C8G
U32B
BANK 2
U32C
BANK 3
U32E
BANK 6
U32D
BANK 5
U32F
BANK 8
U32G
J21
Pin Header 2.54 2Row 14Pins
LED Green LTST-C191KGKT
LED Red LTST-C191KRKT
MPZ1608S221A
3.3V_REF
CPLD_ADC1
CPLD_ADC2
CPLD_ADC3
CPLD_JTAGEN/RESET_BTN
M_TCK
M_TMS
M_TDO
M_TDI
CPLD_CLK
HDMI_PD
VDAC_PSAVE_N
DBG_UART_TX
DBG_UART_RX
FPGA_TX
K_TDO
K_IO2
K_IO3
FPGA_RESET_N
5V_JOY_PG
EN_5V_JOY_N
FPGA_PROG_B
K_IO1
CPLD_DBG10
FPGA_TCK
CPLD_DBG11
CPLD_DBG9
CPLD_DBG8
CPLD_DBG5
CPLD_DBG7
FPGA_INIT
FPGA_DONE
CPLD_DBG4
CPLD_DBG6
FPGA_TDO
FPGA_TDI
K_TCK
K_TDI
CPLD_DBG3
CPLD_DBG1
FPGA_RX
K_JTAGEN
FPGA_TMS
CPLD_DBG0
CPLD_DBG2
K_TMS
KB_IO2
KB_TCK
KB_IO3
KB_TMS
KB_JTAGEN
KB_TDO
KB_IO1
LED_R
CPLD_CFG3
PMOD2_EN
CPLD_CFG1
LED_G
CPLD_CFG0
KB_TDI
PMOD1_FLG
PMOD1_EN
PMOD2_FLG
CPLD_CFG2
TE_TCK
TE_TDO
TE_TDI
TE_TMS
TE_UART_TX
M10_NSTATUS
M10_CONF_DONE
M_TX
M_RX
TE_UART_RX
M10_NCONF
M10_CONF_DONE
M10_NSTATUS
Bank1
Bank2
Bank3
Bank5
Bank6
Bank8
trenz electronic
Title:
MEGA65
A4
Number:
TE0765
Default
Rev.
02
Date:
2019-03-11
Copyright: Trenz Electronic GmbH
Page12 of 23
Filename:
SYS_MAX10_CTRL.SchDoc

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page14 of 23 |
| Filename: SOUND.SchDoc | | | |

| | | | |
|---|---|---|---|
| Title: MEGA65 | | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page15 of 23 |
| Filename: JOY.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page16 of 23 |
| Filename: KEYBOARD.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page17 of 23 |
| Filename: Floppy.SchDoc | | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page 18 of 23 |
| Filename: VGA.SchDoc | | | |

U6A
Video
HDMI
PixelC
Blue
Green
Red
VSYNC
HSYNC
CLK
DE
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
D16
D17
D18
D19
D20
D21
D22
D23
D24
D25
D26
D27
D28
D29
D30
D31
D32
D33
D34
D35
TXC+
TXC–
TX0+
TX0–
TX1+
TX1–
TX2+
TX2–
HPD
CEC
CEC_CLK
DDCSCL
DDCSDA
HEAC+
HEAC-
SCL
SDA
INT
HDMI_VS
HDMI_HS
HDMI_CLK
HDMI_DE
HDMI_D0
HDMI_D1
HDMI_D2
HDMI_D3
HDMI_D4
HDMI_D5
HDMI_D6
HDMI_D7
HDMI_D8
HDMI_D9
HDMI_D10
HDMI_D11
HDMI_D12
HDMI_D13
HDMI_D14
HDMI_D15
HDMI_D16
HDMI_D17
HDMI_D18
HDMI_D19
HDMI_D20
HDMI_D21
HDMI_D22
HDMI_D23
HDMI_D24
HDMI_D25
HDMI_D26
HDMI_D27
HDMI_D28
HDMI_D29
HDMI_D30
HDMI_D31
HDMI_D32
HDMI_D33
HDMI_D34
HDMI_D35
HDMI_TXC_P
HDMI_TXC_N
HDMI_TX0_P
HDMI_TX0_N
HDMI_TX1_P
HDMI_TX1_N
HDMI_TX2_P
HDMI_TX2_N
HPD_A
CEC_A
CEC_CLK
SCL_A
SDA_A
HDMI_SCL
HDMI_SDA
HDMI_INT
ADV7511KSTZ
GND
C94
1µF
6.3V
C95
100nF
DVDD_HDMI
R22
49R9
R23
49R9
C78
1µF
6.3V
C79
1µF
6.3V
1%
2K R46
3.3V
2K R47
2K R48
J4
Data2+
Data2 Shield
Data2-
Data1+
Data1 Shield
Data1-
Data0+
Data0 Shield
Data0-
Clock+
Clock Shield
Clock-
CEC
NC
SCL
SDA
DDC/CEC GND
+5V
Hot Plug Detect
FRAME
FRAME
FRAME
FRAME
HDMI Connector
HDMI_TX2_P
HDMI_TX2_N
HDMI_TX1_P
HDMI_TX1_N
HDMI_TX0_P
HDMI_TX0_N
HDMI_TXC_P
HDMI_TXC_N
CEC_B
SCL_B
SDA_B
5V_HDMI
HPD_B
U6B
Audio
SPDIF
SPDIF_OUT
MCLK
SCLK
LRCLK
DSD_CLK
PD
I2S0
I2S1
I2S2
I2S3
DSD0
DSD1
DSD2
DSD3
DSD4
DSD5
HDMI_SPDIF
HDMI_SPDIFOUT
HDMI_PD
ADV7511KSTZ
L1
MPZ1608S221A
C80
10µF
C81
100nF
C82
100nF
C83
100nF
C84
100nF
C85
10nF
C86
10nF
C87
10nF
C88
10nF
L8
MPZ1608S221A
C89
10µF
C96
100nF
C97
100nF
C101
100nF
C103
100nF
C104
10nF
C105
10nF
AVDD_HDMI
L9
MPZ1608S221A
1.8V
C107
10µF
C108
100nF
C109
100nF
C110
10nF
C112
10nF
PLVDD_HDMI
L6
MPZ1608S221A
3.3V
C113
10µF
C114
10µF
C115
100nF
C116
10nF
MVDD_HDMI
U6C
Power
DVDD
PVDD
PLVDD
BGVDD
AVDD
MVDD
R_EXT
R14
887R
DVDD_HDMI
AVDD_HDMI
PLVDD_HDMI
MVDD_HDMI
ADV7511KSTZ
U10
CEC_A
SCL_A
SDA_A
HPD_A
LS_OE
GND
CEC_B
SCL_B
SDA_B
HPD_B
VCC5V
CT_HPD
VCCA
D2+
D2-
D1+
D1-
D0+
D0-
CLK+
CLK-
5V_OUT
HDMI_TX2_P
HDMI_TX2_N
HDMI_TX1_P
HDMI_TX1_N
HDMI_TX0_P
HDMI_TX0_N
HDMI_TXC_P
HDMI_TXC_N
5V_HDMI
TPD12S016PWR
C106
100nF
3.3V
5V
CT_HPD
C117
100nF
C171
100nF
trenz electronic
Title:
MEGA65
A4
Number:
TE0765
Default
Rev.
02
Date:
2019-03-11
Copyright: Trenz Electronic GmbH
Page19 of 23
Filename:
HDMI.SchDoc

trenz electronic

| | | | |
|---|---|---|---|
| Title: MEGA65 | | | |
| A4 | Nummer: TE0765 Default | | Rev. 02 |
| Datum: 2019-03-11 | Zeichner: Trenz Electronic GmbH | | Blatt 20 von 23 |
| Filename: Ethernet.SchDoc | | | |

http://linux-sunxi.org/MicroSD_Breakout

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | | Page21 of 23 |
| Filename: LED_SW_BUT.SchDoc | | | |

| | | |
|---|---|---|
| Title: MEGA65 | | |
| A4 | Number: TE0765 Default | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH | Page22 of 23 |
| Filename: PowerMain.SchDoc | | |

| | | | |
|---|---|---|---|
| trenz electronic | Title: MEGA65 | | |
| A4 | Number: TE0765 Default | | Rev. 02 |
| Date: 2019-03-11 | Copyright: Trenz Electronic GmbH / TT | | Page 23 of 23 |
| Filename: POWER.SchDoc | | | |

# NEXYS WIDGET BOARD SCHEMATICS

| | |
|---|---|
| TITLE: c65Keyb | |
| Document Number: | REV: |
| Date: 6/05/2022 10:40 AM | Sheet: 1/2 |

## Nexys 4 Input

## Output Driver

## Expansion Slot

| | |
|---|---|
| TITLE: c65Keyb | |
| Document Number: | REV: |
| Date: 6/05/2022 10:40 AM | Sheet: 2/2 |

Bottom

# APPENDIX D

## Supporters & Donors

- Organisations
- Contributors
- Supporters

The MEGA65 would not have been possible to create without the generous support of many organisations and individuals.

We are still compiling these lists, so apologies if we haven't included you yet. If you know anyone we have left out, please let us know, so that we can recognise the contribution of everyone who has made the MEGA65 possible, and into the great retro-computing project that it has become.

# ORGANISATIONS

**The MEGA Museum of Electronic Games & Art e.V. Germany**
EVERYTHING

**Trenz Electronic, Germany**
MOTHERBOARD MANUFACTURING SALES

**Hintsteiner, Austria**
CASE

**GMK, Germany**
KEYBOARD

**KEVAG Telekom, Germany**
WEB HOSTING

# CONTRIBUTORS

**Andreas Liebeskind**
*(libi in paradize)*
CFO MEGA eV

**Thomas Hertzler**
*(grumpyninja)*
USA spokesman

**Russell Peake**
*(rdpeake)*
Bug herding

**Alexander Nik Petra**
*(n0d)*
Early case design

**Ralph Egas**
*(0-limits)*
Business advisor

**Lucas Moss**
MEGAphone PCB design

**Daren Klamer**
*(Impakt)*
Manual proof-reading

**Daniël Mantione**
*(dmantione)*
C64 hardware guru

**Dr. Canan Hastik**
*(indica)*
Chairwoman MEGA eV

**Simon Jameson**
*(Shallan)*
Platform enhancements

**Stephan Kleinert**
*(ubik)*
Destroyer of BASIC 10

**Wayne Johns**
*(sausage)*
Manual additions

**L. Kleiss**
*(LAK132)*
MegaWAT presentation software

**Maurice van Gils**
*(Maurice)*
BASIC 65 example programs

**Andrew Owen**
*(Cheveron)*
Keyboard, Sinclair support

**Adam Barnes**
*(amb5l)*
HDMI expert and board revision

# SUPPORTERS

@11110110100
3c74ce64
8-Bit Classics
Aaron Smith
Achim Mrotzek
Adolf Nefischer
Adrian Esdaile
Adrien Guichard
Ahmed Kablaoui
Alan Bastian Witkowski
Alan Field
Alastair Paulin-Campbell
Alberto Mercuri
Alexander Haering
Alexander Kaufmann
Alexander Niedermeier
Alexander Soppart
Alfonso Ardire
Amiga On The Lake
André Kudra
André Simeit
André Wösten
Andrea Farolfi
Andrea Minutello
Andreas Behr
Andreas Freier
Andreas Grabski
Andreas Millinger
Andreas Nopper
Andreas Ochs
Andreas Wendel Manufaktur
Andreas Zschunke
Andrew Bingham
Andrew Dixon
Andrew Mondt
Andrzej Hłuchyj
Andrzej Sawiniec
Andrzej Śliwa
Anthony W. Leal
Arkadiusz Bronowicki
Arkadiusz Kwasny
Arnaud Léandre
Arne Drews
Arne Neumann
Arne Richard Tyarks
Axel Klahr
Balaz Ondrej
Barry Thompson
Bartol Filipovic
Benjamin Maas
Bernard Alaiz
Bernhard Zorn
Bieno Marti-Braitmaier
Bigby
Bill LaGrue
Bjoerg Stojalowski
Björn Johannesson
Bjørn Melbøe
Bo Goeran Kvamme
Boerge Noest
Bolko Beutner
Brett Hallen
Brian Gajewski
Brian Green
Brian Juul Nielsen
Brian Reiter
Bryan Pope
Burkhard Franke
Byron Goodman
Cameron Roberton (KONG)
Carl Angervall
Carl Danowski
Carl Stock
Carl Wall
Carlo Pastore
Carlos Silva
Carsten Sørensen
Cenk Miroglu Miroglu
Chang sik Park
Charles A. Hutchins Jr.
Chris Guthrey
Chris Hooper
Chris Stringer
Christian Boettcher
Christian Eick
Christian Gleinser
Christian Gräfe
Christian Heffner
Christian Kersting
Christian Schiller
Christian Streck
Christian Weyer
Christian Wyk
Christoph Haug
Christoph Huck
Christoph Pross
Christopher Christopher
Christopher Kalk
Christopher Kohlert
Christopher Nelson
Christopher Taylor
Christopher Whillock
Claudio Piccinini
Claus Skrepek
Collen Blijenberg
Constantine Lignos
Crnjaninja
Daniel Auger
Daniel Julien
Daniel Lobitz
Daniel O'Connor
Daniel Teicher
Daniel Tootill
Daniel Wedin
Daniele Benetti
Daniele Gaetano Capursi
Dariusz Szczesniak
Darrell Westbury
David Asenjo Raposo
David Dillard
David Gorgon
David Norwood
David Raulo
David Ross
de voughn accooe
Dean Scully
Dennis Jeschke
Dennis Schaffers
Dennis Schierholz

Dennis Schneck
denti
Dick van Ginkel
Diego Barzon
Dierk Schneider
Dietmar Krueger
Dietmar Schinnerl
Dirk Becker
Dirk Wouters
Domingo Fivoli
DonChaos
Donn Lasher
Douglas Johnson
Dr. Leopold Winter
Dusan Sobotka
Earl Woodman
Ed Reilly
Edoardo Auteri
Eduardo Gallardo
Eduardo Luis Arana
Eirik Juliussen Olsen
Emilio Monelli
EP Technical Services
Epic Sound
Erasmus Kuhlmann
ergoGnomik
Eric Hilaire
Eric Hildebrandt
Eric Hill
Eric Jutrzenka
Erwin Reichel
Espen Skog
Evangelos Mpouras
Ewan Curtis
Fabio Zanicotti
Fabrizio Di Dio
Fabrizio Lodi
FARA Gießen GmbH
FeralChild
First Choice Auto's
Florian Rienhardt
Forum64. de
Francesco Baldassarri
Frank Fechner
Frank Glaush
Frank Gulasch
Frank Haaland
Frank Hempel
Frank Koschel
Frank Linhares
Frank Sleeuwaert
Frank Wolf
FranticFreddie
Fredrik Ramsberg
Fridun Nazaradeh
Friedel Kropp
Garrick West
Gary Lake-Schaal
Gary Pearson
Gavin Jones
Geir Sigmund Straume
Gerd Mitlaender
Giampietro Albiero
Giancarlo Valente
Gianluca Girelli
Giovanni Medina
Glen Fraser
Glen R Perye III
Glenn Main
Gordon Rimac
GRANT BYERS
Grant Louth
Gregor Bubek
Gregor Gramlich
Guido Ling
Guido von Gösseln
Guillaume Serge
Gunnar Hemmerling
Günter Hummel
Guy Simmons
Guybrush Threepwood
Hakan Blomqvist
Hans Pronk
Hans-Jörg Nett
Hans-Martin Zedlitz
Harald Dosch
Harri Salokorpi
Harry Culpan
Harry Venema
Heath Gallimore
Heinz Roesner
Heinz Stampfli
Helge Förster
Hendrik Fensch
Henning Harperath
Henri Parfait
Henrik Kühn
Holger Burmester
Holger Sturk
Howard Knibbs
Hubert de Hollain
Huberto Kusters
Hugo Maria Gerardus v.d. Aa
Humberto Castaneda
Ian Cross
IDE64 Staff
Igor Ianov
Immo Beutler
Ingo Katte
Ingo Keck
Insanely Interested Publishing
IT-Dienstleistungen Obsieger
Ivan Elwood
Jaap HUIJSMAN
Jace Courville
Jack Wattenhofer
Jakob Schönpflug
Jakub Tyszko
James Hart
James Marshburn
James McClanahan
James Sutcliffe
Jan Bitruff
Jan Hildebrandt
Jan Iemhoff
Jan Kösters
Jan Peter Borsje
Jan Schulze
Jan Stoltenberg-Lerche
Janne Tompuri
Jannis Schulte
Jari Loukasmäki
Jason Smith
Javier Gonzalez Gonzalez
Jean-Paul Lauque
Jeffrey van der Schilden
Jens Schneider
Jens-Uwe Wessling

Jesse DiSimone
Jett Adams
Johan Arneklev
Johan Berntsson
Johan Svensson
Johannes Fitz
John Cook
John Deane
John Dupuis
John Nagi
John Rorland
John Sargeant
John Traeholt
Jon Sandelin
Jonas Bernemann
Jonathan Prosise
Joost Honig
Jordi Pakey-Rodriguez
Jöre Weber
Jörg Jungermann
Jörg Schaeffer
Jörg Weese
Josef Hesse
Josef Soucek
Josef Stohwasser
Joseph Clifford
Joseph Gerth
Jovan Crnjanin
Juan Pablo Schisano
Juan S. Cardona Iguina
JudgeBeeb
Juliussen Olsen
Juna Luis Fernandez Garcia
Jürgen Endras
Jürgen Herm Stapelberg
Jyrki Laurila
Kai Pernau
Kalle Pöyhönen
Karl Lamford
Karl-Heinz Blum
Karsten Engstler
Karsten Westebbe
katarakt
Keith McComb
Kenneth Dyke
Kenneth Joensson
Kevin Edwards
Kevin Thomasson
Kim Jorgensen
Kim Rene Jensen
Kimmo Hamalainen
Konrad Buryło
Kosmas Einbrodt
Kurt Klemm
Lachlan Glaskin
Large bits collider
Lars Becker
Lars Edelmann
Lars Slivsgaard
Lasse Lambrecht
Lau Olivier
Lee Chatt
Loan Leray
Lorenzo Quadri
Lorenzo Travagli
Lorin Millsap
Lothar James Foss
Lothar Serra Mari
Luca Papinutti
Ludek Smetana
Lukas Burger
Lutz-Peter Buchholz
Luuk Spaetgens
Mad Web Skills
MaDCz
Magnus Wiklander
Maik Diekmann
Malte Mundt
Manfred Wittemann
Manuel Beckmann
Manzano Mérida
Marc "3D-vice" Schmitt
Marc Bartel
Marc Jensen
Marc Schmidt
Marc Theunissen
Marc Tutor
Marc Wink
Marcel Buchtmann
Marcel Kante
Marco Beckers
Marco Cappellari
Marco Rivela
Marco van de Water
Marcus Gerards
Marcus Herbert
Marcus Linkert
Marek Pernicky
Mario Esposito
Mario Fetka
Mario Teschke
Mariusz Tymków
Mark Adams
Mark Anderson
Mark Green
Mark Hucker
Mark Leitiger
Mark Spezzano
Mark Watkin
Marko Rizvic
Markus Bieler
Markus Bonet
Markus Dauberschmidt
Markus Fehr
Markus Fuchs
Markus Guenther-Hirn
Markus Liukka
Markus Merz
Markus Roesgen
Markus Uttenweiler
Martin Bauhuber
Martin Benke
Martin Gendera
Martin Groß
Martin Gutenbrunner
Martin Johansen
Martin Marbach
Martin Sonnleitner
Martin Steffen
Marvin Hardy
Massimo Villani
Mathias Dellacherie
Mathieu Chouinard
Matthew Adams
Matthew Browne
Matthew Carnevale
Matthew Palmer
Matthew Santos

Matthias Barthel
Matthias Dolenc
Matthias Fischer
Matthias Frey
Matthias Grandis
Matthias Guth
Matthias Lampe
Matthias Meier
Matthias Mueller
Matthias Nofer
Matthias Schonder
Maurice Al-Khaliedy
Max Ihlenfeldt
Meeso Kim
Michael Dailly
Michael Dötsch
Michael Dreßel
Michael Fichtner
Michael Fong
Michael Geoffrey Stone
Michael Gertner
Michael Grün
Michael Habel
Michael Härtig
Michael Haynes
Michael J Burkett
Michael Jensen
Michael Jurisch
Michael Kappelgaard
Michael Kleinschmidt
Michael Lorenz
Michael Mayerhofer
Michael Nurney
Michael Rasmussen
Michael Richmond
Michael Sachse
Michael Sarbak
Michael Schneider
Michael Scholz
Michael Timm
Michael Traynor
Michael Whipp
Michal Ursiny
Michele Chiti
Michele Perini
Michele Porcu
Miguel Angel Rodriguez Jodar
Mikael Lund
Mike Betz
Mike Kastrantas
Mike Pikowski
Mikko Hämäläinen
Mikko Suontausta
Mirko Roller
Miroslav Karkus
Morgan Antonsson
Moritz
Morten Nielsen
MUBIQUO APPS,SL
Myles Cameron-Smith
Neil Moore
Nelson
neoman
Nicholas Melnick
Nikolaj Brinch Jørgensen
Nils Andreas
Nils Eilers
Nils Hammerich
Nils77
Norah Smith
Norman King
Normen Zoch
Olaf Grunert
Ole Eitels
Oliver Boerner
Oliver Brüggmann
Oliver Graf
Oliver Smith
Olivier Bori
ONEPSI LLC
oRdYNe
Osaühing Trioflex
OSHA-PROS USA
Padawer
Patrick Becher
Patrick Bürckstümmer
Patrick de Zoete
Patrick Toal
Patrick Vogt
Paul Alexander Warren
Paul Gerhardt (KONG)
Paul Jackson
Paul Johnson
Paul Kuhnast (mindrail)
Paul Massay
Paul Westlake
Paul Woegerer
Pauline Brasch
Paulo Apolonia
Pete Collin
Pete of Retrohax.net
Peter Eliades
Peter Gries
Peter Habura
Peter Herklotz
Peter Huyoff
Peter Knörzer
Peter Leswell
Peter Weile
Petri Alvinen
Philip Marien
Philip Timmermann
Philipp Rudin
Pierre Kressmann
Pieter Labie
Piotr Kmiecik
Power-on.at
Przemysław Safonow
Que Labs
R Welbourn
R-Flux
Rafał Michno
Rainer Kappler
Rainer Kopp
Rainer Weninger
Ralf Griewel
Ralf Pöscha
Ralf Reinhardt
Ralf Schenden
Ralf Smolarek
Ralf Zenker
Ralph Bauer
Ralph Wernecke
Rédl Károly
Reiner Lanowski
Remi Veilleux
Riccardo Bianchi
Richard Englert

Richard Good
Richard Menedetter
Richard Sopuch
Rick Reynolds
Rico Gruninger
Rob Dean
Robert Bernardo
Robert Eaglestone
Robert Grasböck
Robert Miles
Robert Schwan
Robert Shively
Robert Tangmar
Robert Trangmar
Rodney Xerri
Roger Olsen
Roger Pugh
Roland Attila Kett
Roland Evers
Roland Schatz
Rolf Hass
Ronald Cooper
Ronald Hunn
Ronny Hamida
Ronny Preiß
Roy van Zundert
Rüdiger Wohlfromm
Ruediger Schlenter
Rutger WIllemsen
Sampo Peltonen
Sarmad Gilani
SAS74
Sascha Hesse
Scott Halman
Scott Hollier
Scott Robison
Sebastian Baranski
Sebastian Bölling
Sebastian Felzmann
Sebastian Lipp
Sebastian Rakel
Şemseddin Moldibi
Seth Morabito
Shawn McKee
Siegfried Hartmann
Sigurbjorn Larusson
Sigurdur Finnsson
Simon Lawrence
Simon Wolf
spreen.digital
Stefan Haberl
Stefan Kramperth
Stefan Richter
Stefan Schultze
Stefan Sonnek
Stefan Theil
Stefan Vrampe
Stefano Canali
Stefano Mozzi
Steffen Reiersen
Stephan Bielmann
Stephen Jones
Stephen Kew
Steve Gray
Steve Kurlin
Steve Lemieux
Steven Combs
Stewart Dunn
Stuart Marsh
Sven Neumann
Sven Stache
Sven Sternberger
Sven Wiegand
Szabolcs Bence
Tantrumedia Limited
Techvana Operations Ltd.
Teddy Turmeaux
Teemu Korvenpää
The Games Foundation
Thierry Supplisson
Thieu-Duy Thai
Thomas Bierschenk
Thomas Edmister
Thomas Frauenknecht
Thomas Gitzen
Thomas Gruber
Thomas Haidler
Thomas Jager
Thomas Karlsen
Thomas Laskowski
Thomas Marschall
Thomas Niemann
Thomas Scheelen
Thomas Schilling
Thomas Tahsin-Bey
Thomas Walter
Thomas Wirtzmann
Thorsten Knoll
Thorsten Nolte
Tim Krome
Tim Waite
Timo Weirich
Timothy Blanks
Timothy Henson
Timothy Prater
Tobias Butter
Tobias Heim
Tobias Köck
Tobias Lüthi
Tommi Vasarainen
Toni Ammer
Tore Olsen
Torleif Strand
Torsten Schröder
Tuan Nguyen
Uffe Jakobsen
Ulrich Hintermeier
Ulrich Nieland
Ulrik Kruse
Ursula Förstle
Uwe Anfang
Uwe Boschanski
Vedran Vrbanc
Verm Project
Wayne Rittimann
Wayne Sander
Wayne Steele
Who Knows
Winfried Falkenhahn
Wolfgang Becker
Wolfgang Stabla
Worblehat
www.patop69.net
Yan B
Zoltan Markus
Zsolt Zsila
Zytex Online Store

# Bibliography

[1] L. Soares and M. Stumm, "Flexsc: Flexible system call scheduling with exceptionless system calls." in *Osdi*, vol. 10, 2010, pp. 1-8.

[2] N. Montfort, P. Baudoin, J. Bell, I. Bogost, J. Douglass, M. C. Marino, M. Mateas, C. Reas, M. Sample, and N. Vawter, *10 PRINT CHR $(205.5+ RND (1));: GOTO 10*. MIT Press, 2012.

[3] Actraiser, "Vic-ii for beginners: Screen modes, cheaper by the dozen," 2013. [Online]. Available: http://dustlayer.com/vic-ii/2013/4/26/vic-ii-for-beginners-screen-modes-cheaper-by-the-dozen

# INDEX

altpal, 48
Apple, 16

bgcolor, 47
blink, 48
bordercolor, 47
box, 51

cellcolor, 49
cgetc, 59
cinput, 60
clearattr, 49
clrscr, 45
conionit, 43
copyright, ii
cprintf, 58
cputc, 56
cputcxy, 55, 57
cputdec, 57
cputhex, 56
cputnc, 56
cputncxy, 58
cputs, 57
cputsxy, 57

emulator, 15

fillrect, 51
Flash Menu, 85
flushkeybuf, 60
Freeze Menu, 85

getcharsetaddr, 45
getcolramoffset, 44
getkeymodstate, 60
getmapedpal, 50
getpalbank, 50
getpalbanka, 50
getscreenaddr, 44
getscreensize, 45
gohome, 52
gotox, 53
gotoxy, 52
gotoy, 53

highlight, 48
hline, 52
Hyppo Error Codes, 93, 95
    $01, 95
    $02, 95
    $03, 95
    $04, 95
    $05, 95
    $06, 95
    $07, 95
    $08, 95, 120
    $10, 95, 98, 120, 126, 133, 135, 143
    $11, 96
    $20, 96
    $21, 96
    $80, 96, 125
    $81, 96, 126
    $82, 96
    $83, 96
    $84, 96, 115, 116, 118, 119
    $85, 96, 121
    $86, 96, 119
    $87, 96, 99, 118
    $88, 96, 106–108, 115, 116, 128
    $89, 96, 122
    $8A, 96
    $8B, 96
    $8C, 96
    $8D, 96, 117, 124
    $8E, 96
    $FF, 96
Hyppo Move to Root Directory, 101
Hyppo Services, 91
    $D640 $00, 97
    $D640 $02, 113
    $D640 $04, 111
    $D640 $06, 125
    $D640 $08, 114
    $D640 $0A, 112

$D640 $0C, 99
$D640 $0E, 117
$D640 $10, 124
$D640 $12, 118
$D640 $14, 120
$D640 $16, 103
$D640 $18, 119
$D640 $1A, 122
$D640 $1C, 127
$D640 $1E, 117
$D640 $20, 104
$D640 $22, 102
$D640 $24, 124
$D640 $26, 124
$D640 $28, 110
$D640 $2A, 124
$D640 $2C, 105
$D640 $2E, 126
$D640 $30, 107
$D640 $32, 108
$D640 $34, 106
$D640 $36, 115
$D640 $38, 95
$D640 $3A, 98
$D640 $3C, 101
$D640 $3E, 116
$D640 $40, 128
$D640 $42, 130
$D640 $44, 131
$D640 $46, 129
$D640 $48, 134
$D640 $50, 136
$D640 $52, 140
$D640 $54, 136
$D640 $56, 147
$D640 $58, 136
$D640 $60, 144
$D640 $62, 132
$D640 $64, 136
$D640 $66, 132
$D640 $68, 132
$D640 $6A, 132
$D640 $6C, 144
$D640 $6E, 132
$D640 $70, 146
$D640 $72, 145
$D640 $74, 133
$D640 $76, 143
$D640 $7C, 142
$D640 $7E, 137
$D641 $00, 139
$D641 $02, 138
$D642 $00, 148
$D642 $02, 148
$D642 $04, 148
$D642 $06, 148
$D642 $10, 149
$D642 $12, 149
$D642 $14, 149
$D642 $16, 149
$D643 $xx, 141
$D67F $xx, 149

kbhit, 59
keyboard, 15

macOS, 16
MEGA Flash, 85
movedown, 54
moveleft, 54
moveright, 54
moveup, 53

OpenROMs, 18

pcprintf, 59
pcputc, 55
pcputs, 56
pcputsxy, 55

Registers
$D640, 91
$D641, 91
$D642, 91
$D643, 91

$D644, 91
$D645, 91
$D646, 91
$D647, 91
$D648, 91
$D649, 91
$D64A, 91
$D64B, 91
$D64C, 91
$D64D, 91
$D64E, 91
$D64F, 91
$D650, 91
$D651, 91
$D652, 91
$D653, 91
$D654, 91
$D655, 91
$D656, 91
$D657, 91
$D658, 91
$D659, 91
$D65A, 91
$D65B, 91
$D65C, 91
$D65D, 91
$D65E, 91
$D65F, 91
$D660, 91
$D661, 91
$D662, 91
$D663, 91
$D664, 91
$D665, 91
$D666, 91
$D667, 91
$D668, 91
$D669, 91
$D66A, 91
$D66B, 91
$D66C, 91
$D66D, 91
$D66E, 91
$D66F, 91
$D670, 91
$D671, 91
$D672, 91
$D673, 91
$D674, 91
$D675, 91
$D676, 91
$D677, 91
$D678, 91
$D679, 91
$D67A, 91
$D67B, 91
$D67C, 91
$D67D, 91
$D67E, 91
$D67F, 91
revers, 47
Root directory, 101

SD card, 17
set16bitcharmode, 46
setcharsetaddr, 44
setcolramoffset, 44
setextendedattrib, 46
sethotregs, 46
setmapedpal, 50
setpalbank, 49
setpalbanka, 49
setpalentry, 50
setscreenaddr, 43
setscreensize, 45

textcolor, 47
togglecase, 46, 47

underline, 48
Utility Menu, 85

vline, 52

wherex, 54
wherey, 55