retro zerø

Hacker's Handbook

For Ringzer0 Training, by Steve Lord

Contents

Chapter 1

Preface

Hi!

This is the Retro Zer0, a very special present from Ring Zer0 Training. It's a fully functional extended CP/M compatible computer you can understand, use and modify.

It runs professional software from programming languages to word processors, spreadsheets and more. An SD card slot lets you move files between this and more powerful computers. This manual was partly written in WordStar 4 on a Retro Zer0.

The Retro Zer0 has few enough parts that it's inner workings are easy to understand. If taken care of it will still function decades from now. We hope you'll have many years of fun with your Retro Zer0.

Saumil, Steve, and Marizel.

1.1 Safety

Although the Retro Zer0 is designed to run off a 5v USB connection, care should be taken when handling exposed PCBs.

Edges are sharp and parts may be spikey. Always use the supplied M4 stand-offs when placing the Retro Zer0 on any surface.

Keep liquids away from the Retro Zer0. If you spill any liquids, disconnect the USB connector from the source, remove the Retro Zer0 and let it dry out. Sticky residues can be removed with rubbing alcohol and a toothbrush.

Do not short PCB connections. If reprogramming the Retro Zer0, remember that VGA synchronization issues can cause damage to CRT monitors and in some cases could even lead to a fire. Retro Zer0 does not blank the screen after inactivity. Turn the screen off when not needed to avoid burn-in risk.

1.2 Understanding This Document

Several conventions are used throughout this manual.

References to `commands` are in a monospace font.

Things to type `ARE IN THE SAME FORMAT BUT UPPER CASE`

File locations are italicised, like *A:GAMES/GORILLA.COM*.

Things that need to be highlighted to the reader are **in bold**.

Chapter 2

Before You Start

2.1 Check You Have Everything

Your Retro Zer0 package should include the following items:

- Retro Zer0 board (looks like a 5.25” Floppy disk)
- 16Gb SD Card
- PS/2 Keyboard
- VGA Cable
- 3m USB Cable
- 4x M4 Standoffs in a small bag.

2.2 Back Up The SD Card

Bundled with your Retro Zer0 is an SD Card. This contains software, documents and an archive of classic text files from various BBS and CP/M archives.

Before you use your Retro Zer0, back up the SD Card. You can just copy the folders onto another computer. Alternatively a tool like etcher or DD can be used to back up the disk image.

While the chances of Retro Zer0 corrupting the SD Card are small, power loss during an SD Card write could cause the SD card to become corrupted.

CP/M Uses it's own disk filesystem format. Thankfully the CP/M compatible OS used translates the calls into a FAT-compatible form. The upside to this is that you can copy files directly from and to the card and Retro Zer0 should see them. The downside is that CP/M file recovery tools such as UNDELETE are unlikely to work.

2.3 Connecting Up Your Retro Zer0

Connecting Retro Zer0 is easy, but care should be taken to avoid damaging contacts and connectors.

First identify a USB power source. We've provided a USB cable, but not a local power adapter. We recommend using a USB charger plugged directly into the wall. Should anything happen this means only the charger is affected. Computers can't always provide appropriate power levels over USB. If something bad happened while connected to a computer,

Retro Zer0 could damage your computer's USB bus. Use a USB connection to a computer only when you want to access the USB Serial interface, for example when hacking on firmware.

Plug the USB cable into your power adapter (not supplied), and plug the adapter into an appropriate wall socket. Don't connect the USB cable to Retro Zer0 just yet.

We've provided M4 stand-offs to lay Retro Zer0 flat. The PCB components are through-hole and might scratch some surfaces. We'd recommend using the M4 stand-offs where possible. For each screw, put the washer on the between the screw and board, slide the screw through the bottom then screw the nut on the top. If you'd prefer to wall mount your Retro Zer0, you can use the M4 holes to mount to a cork board which you can then wall mount as you please.

Insert the SD Card into Retro Zer0's SD Card socket. The socket is spring-loaded for your pleasure.

Plug the supplied keyboard into Retro Zer0's PS/2 socket. The supplied keyboard has an imprint of a keyboard image on the connector. When properly connected this should face upwards. Do not force the PS/2 Connector in and support the connector with your finger to relieve stress on the board joint.

Connect the VGA cable to Retro Zer0 and your monitor. Support the connector with your finger while connecting and disconnecting to relieve stress on the board joint.

2.4 Powering Up

Now connect the USB cable. If you have a cable with a short connector you can feed it through the hole in the middle of the board. Your Retro Zer0 will look 5% cooler in this configuration.

Power will come on automatically. A red light should appear on the ESP32 board. The LED at D1 should also light up. You can desolder this if you prefer not to have power lights. Suggested LED mods are made in the tinkering section of this handbook.

Turn on your monitor and switch it to VGA. You may need to use your monitor's auto-adjust or manually adjust the monitor for your screen.

2.5 Resetting Your Retro Zer0

To reset your Retro Zer0, press the **EN** button on the ESP32. This is the small button to the left of the USB power connector.

2.6 Powering Down

To power down your Retro Zer0, unplug the USB socket from the wall first. Then, taking care to support the USB connector, unplug the cable. As Retro Zer0 does not have an on/off button, repeated plugging and unplugging at the ESP32 end could damage the USB connector over time.

Chapter 3

First Steps

3.1 Testing The Keyboard

When you first turn on the Retro Zer0, you should see a screen with text in different colours. The key points on the screen are:

- The Drive Search Path (which order CP/M looks for programs)
- That we're running Multissession CP/M 3 (Plus)
- Available drive mounts
- Available and total free memory
- Our Terminal type
- A flashing cursor next to the symbols "A>"

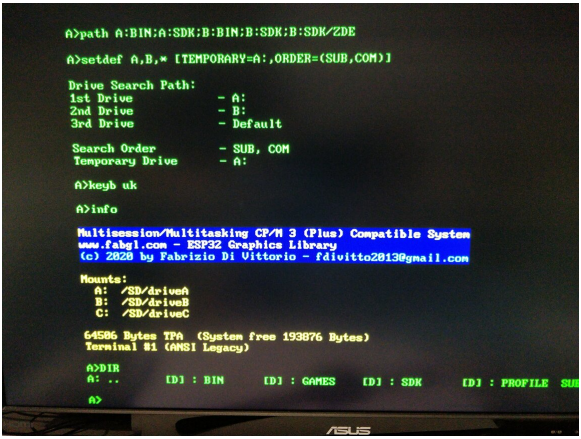

Figure 3.1: First boot and running the dir command

When you type on the keyboard, text should appear next to the A> prompt. Type `dir` press the return key. A directory listing should appear.

Now press Caps Lock. The keyboard's blue caps lock light should light up. Enter `DIR` again. CP/M Is generally not case sensitive, although applications may be.

3.2 Using CP/M

CP/M looks very similar to MS-DOS. Some might say too similar, although CP/M came first. Much of what CP/M does will feel familiar to DOS users, then suddenly things will change from what you might expect.

There are 3 mounted drives (2 if you have no SD card inserted), A:, B: and C:. You can change between them by entering `A:`, `B:`, or `C:` and pressing return.

To change directories, you can use the `cd` command. For example, to move to the *BIN* directory, you could enter `CD BIN` and press return. To move up a directory, use `CD ..`

Instead of copying files, CP/M uses the `pip` command. However, this implementation features internal versions of `cp` and `copy`.

To delete a file, use the `era` (for erase) command. Internal `rm` and `del` commands also work.

To create a directory, use `mkdir`. To delete it use `rmdir`.

Files generally have extensions. Executable programs generally have the extension *.COM* or *.EXE* - e.g. pip.com.

There is a built-in editor, `ed.com`. Using it without reading a CP/M manual first may be a traumatic experience.

On boot, CP/M executes the script *A:PROFILE.SUB*. If you've ever edited AUTOEXEC.BAT on MS-DOS, this is a simpler startup script.

For more information about how to use CP/M, the CPM 3 Users' Guide is on the SD card under the docs folder. The CP/M 3 guide also covers ed should you need it. You'll also find useful text files under *B:ARCWALNUTCD/CPMHELP* and *B:ARC/WALNUTCD/CPMINFO* - these may have extensions like *.HLP* or *.ART* instead of *.TXT*.

Figure 3.2: Rogue on Retro Zer0. RTX Not needed.

3.3 Would You Like To Play A Game?

Lets start with a game. Enter the following:

```
A:
CD /GAMES
INFO
GORILLA
```

The Gorilla game should start. It'll ask you to select a terminal type. If you look above there should be some text in yellow that says, "(ANSI Legacy)". This is the default terminal type, and can be changed with the built-in `emu` command, which we'll look at later. For now select option 3 - ANSI (Color) and enjoy a two-player game of gorilla. Pick and angle and speed and try to take on the other player. If you'd like to quit, press

Ctrl-C to return to the menu, then '4', then 'y' to return to the command prompt.

Everything is terminal-based in CP/M. There are basic graphics options, but none are currently supported by Retro Zer0. That doesn't mean that CP/M is bad. CP/M offers you a **distraction free environment focused on single tasks**. Because Fabrizio Di Vittorio's excellent Multi-tasking CP/M implementation supports multiple concurrent sessions, you can maintain focus across multiple sessions.

The reason CP/M is terminal based is because back in the 70s screens were expensive and terrible, so many people used teletypewriters. Because CP/M became as popular as DOS it ran on a lot of different hardware with different terminal capabilities.

Lets try a different game to get an idea of what this means to us. Enter the following at the command prompt:

```
B:
CD /GAMES/ROGUE1A
ROGUE1A
```

Gibberish should appear on your screen. This is because Rogue expects a different terminal to the ANSI terminal currently in use. If you see gibberish when running a tool, use the `emu` command to try a different terminal type. To quit Rogue, press 'Q' then 'y'. If you see 'Nothing appropriate' when pressing 'q', make sure you press 'Q' instead.

Enter `EMU` to see a list of possible terminal options. Use `EMU 1` to set the terminal type to ADM 3A and enter `ROGUE1A` to start. You should see a box and some symbols on the screen. If

you've never played a roguelike before, this is how the game is supposed to look. Terminal settings are per-session, so you can have an ANSI terminal on one session and an ADM31 session in another.

Press the 'F2' key to start a new session. Change directories to *B:GAMES/ROGUE1A* as before. Enter `DIR` to list all files and directories at the current location. Directories have a '[D]' next to them. You should see a *ROGUE.DOC* file in the listing. Enter `PEEP ROGUE.DOC` to launch the `peep` text file viewer. Use 'f' to go forward a page and 'j' to go back. Now press the 'F1' key to return to the game, and 'F2' to return to the documentation. You can quit `peep` by pressing 'X'.

There's a Rogue tips guide at *B:TEXTFILE/RPG/ROGUE.TXT*. This can also be opened on a new session to use as reference while playing Rogue.

When you're ready, quit Rogue by pressing 'Q'.

CP/M is also the home of Interactive Fiction. Enter the following:

```
B:
CD /GAMES/ZORK123
ZORK1
```

This is the classic text adventure Zork. It's in 3 parts, and will take quite a while to complete. You explore the world of Zork by entering commands in plain English. Try these to start:

```
OPEN MAILBOX
READ LEAFLET
```

To exit, enter quit followed by 'y'. Solutions for the Zork games can be found under *B:TEXTFILE/ADVENTURE/INFOCOM/*. Several adventure games, including the original Colossal Cave adventure are available in the archives under *B:ARC*.

3.4 MultiCPM Extras

CP/M is a single-tasking operating system. One user can use it at a time and one session can run at a time. While there is a concept of numbered user areas, most systems are single user in practice. This implementation of CP/M 3 Plus can create separate sessions using the keyboard function keys. These sessions share the same filesystem, but have independent memory. In practice this means you may experience file errors in one session if another has files open. Use the function keys to move between them.

If you enter `HELP` a display of built-in commands appears. Although `dir` is a CP/M command, the version used is a built-in one. Additional comfort commands such as `copy`, `cp`, `ls`, `md`, `mkdir`, `rmdir`, `rm`, `del` and `delete` are also present here, but not normally in CP/M.

CP/M uses an 8 character filename and 3 character extension filename limit (known as 8.3 filenames). MultiCPM is aware of longer filenames, but applications may not be. If you have trouble with filenames copy them on another computer and rename the copy in 8.3 format. You can remove the SD card as long as you've saved your work and closed your files, although you will need to reboot afterwards.

File extensions are important in CP/M. When using wildcards

such as the asterisk symbol, make sure you specify an extension. `DIR FOO*` will come up empty, while `DIR FOO*.*` will list any files starting with 'FOO'.

The `info` and `dinfo` commands provide some internal system info. You may recognise `info`'s output from the startup sequence.

The `term` command lets you create additional sessions and switch between them from the command line, but it's usually easier just to use the function keys. The `emu` command sets the terminal emulation for a terminal session. Different sessions can use different terminal emulation. Your supplied keyboard will use a UK layout.

The `wifiscan`, `wifi`, `ping` and `telnet` utilities allow you to use the Retro Zer0 as a wifi telnet client. These are covered later in the handbook.

The `format` command will format whatever storage is in use. If no SD card is inserted, it will wipe the SPIFFS flash storage. If an SD card is inserted, it'll wipe that instead.

The `ls` command is faster than `dir`, but doesn't pause at the end of a screen. The `ls` command also sets colours for different types of file.

At the bottom of the help menu, text is displayed to tell you that F1-F6 allows you to create or switch sessions.

3.5 Useful Tools

The `peep` utility is a text viewer. To view the smiley database on the *B:* drive, enter the following:

```
B:
cd /textfile/fun
peep smiley.txt
```

Peep can be controlled via the home keys 'A' through ';'. The space bar moves a page forward, and you can exit the program with either 'X' or less politely with Ctrl-C.

The main keys for `peep` are:

- A: Go to end
- S: Scroll forward until a key is pressed
- D: Move forward one line
- F: Page down
- G: Forward search for a string
- H: Find next
- J: Page Up
- K: Move backward one line
- L: Scroll backward until a key is pressed
- ;: Display first screen
- X: Quit

The more observant reader may notice that moving forwards is with the left hand on the home row, while the right hand moves backwards.

`sq` and `usq` are "Squeeze" and "Unsqueeze" compression tools. Squeezed files use a 'Q' in the middle of their extension. For example, FOO.TXT, when squeezed will be named FOO.TQT.

To unsqueeze a file, use `USQ <FILENAME> C:`, where <FILENAME> is the name of the file you want to unsqueeze. In this example the `usq` tool will unsqueeze to the current working directory on the C: drive. It's better to mess things up on C: than on B: or A:.

The `delbr` utility is a tool for decompressing LBR archives. LBR means "Library", but is the CP/M equivalent of a tar archive. Files contained inside LBR archives are often squeezed.

Some files are compressed using LZW compression. These files use a "Z" in the middle of the extension, for example FOO.TXT would become FOO.TZT. These files can be handled with the `crunch` and `uncr` utilities.

LZH/LHA files can also be handled with `uncr`. Files where the extension's middle letter is "Y" are often LHA compressed, although .LZH and .LHA extensions are also used.

The `unarc` utility can be used to decompress .ARK files.

3.6 What's On The Card?

The SD Card contains gigabytes of utilities, libraries, documents, forum messages, usenet archives, zines and

textfiles to keep you busy.

As mentioned earlier, there are 3 drives bundled with the CP/M installation, *A:*, *B:* and *C:*. These correspond to the directories */driveA*, */driveB* and */driveC* on the SD card.

If no SD card is inserted at boot a minimal *A:* and *B:* drive is created using flash memory.

The *A:* drive contains the core Operating System, tools and programs. Under here you'll find:

- **BIN**: Core commands
- **GAMES**: A few games such as Gorilla and Catchum
- **SDK**: Various low-level programming languages

The *B:* drive contains end-user applications, text collections and more general things to explore. Under here you'll find:

- **APPS**: Applications such as Wordstar
- **ARC**: An archive of classic CP/M files and documents
- **BASIC**: Several collections of MS-BASIC programs
- **BIN**: Extended utilities for things like handling compression
- **DB**: Professional Database Software like DBase
- **GAMES**: A plethora of computer games including Infocom adventures
- **SDK**: More high-level programming languages, such as Forth, Prolog etc.

- **TEXTFILE**: Selected archives from textfiles.com

The *C:* drive is for you to store your files and use as a scratch area. This ensures that whatever you do on the *C:* drive is kept away from the core and application areas.

As well as the installed software, there are a large amount of files under *B:ARC* and *B:TEXTFILE*. These are taken from the Classic CMP CP/M archives and the Textfiles.com BBS Archive. Good starting points to explore include:

- **B:TEXTFILE/MAGAZINES**: A collection of electronic magazines, mostly hacking. We recommend The *LOD* Technical journal and *PHRACK* Magazine.
- **B:TEXTFILE/ETEXT**: Fiction, non-fiction, reference books, the lot. Check out Hugo Cornwall's 1985 Hackers' Handbook at *MODERN/HCKR_HND.TXT* or Edwin Abbot Abbot's classic mathematical romance, Flatland *FICTION/FLAT10A.TXT*.
- **B:ARC/OAKCD**: A Collection of CP/M Software, documentation, source code, games etc. Most directories have a *00-index.txt* file summarizing the contents.
- **B:ARC/WALNUTCD**: More Software, documentation, source code. Lots of useful stuff under *BEEHIVE* but not everything works.
- **B:ARC/RLEE**: A massive software archive sorted by publisher name.
- **B:ARC/RA**: A big organised collection of applications and tools.

Under */docs/* you'll find a collection of PDF manuals and magazines to read through on a more traditional system. Unfortunately there is no PDF reader for CP/M.

Under /fw/ you'll find the source code for the emulation framework used to run CP/M. You are actively encouraged to fold, spindle and mutilate this into any way you wish.

3.7 Putting It All Together

Lets go through an archive, find a program, unarchive it, test it and store it somewhere. A Tetris-like game can be found on the Walnut Creek CD Archive. Lets prepare a directory under C:, find the game and unarchive it.

```
C:
MKDIR QUATRIS
CD QUATRIS
B:
CD /ARC/WALNUTCD/BEEHIVE/GAMES
PEEP 00-INDEX.TXT
F
F
X
UNARC QUATRIS2.ARC C:
```

Switch to the C: drive and view the documentation extracted from the archive.

```
C:
```

```
PEEP QUATRIS.DOC
F
F
F
X
```

Note that the documentation says that the program is configured for a “Kaypro/Televideo/ADM” type terminal. Change the terminal to match using the `emu` function.

```
EMU 5
QUATRIS
9
```

Enjoy the game, then return to the command prompt when ready. To keep the C: drive clear for scratchpad use, you should move the game to the B: drive.

```
B:
CD /GAMES
MKDIR QUATRIS
CD QUATRIS
C:
COPY *.* B:
ERA *.*
Y
CD ..
RMDIR QUATRIS
Y
B:
DIR
```

From now on you can run Quatris from *B:GAMES/QUATRIS* and your C: drive is clear for future use.

3.8 Where To Read More

The CP/M Plus Command Summary (*/docs/cpm3-cmd.pdf*) is a great place to start. To learn about CP/M Plus internals, the CP/M Plus Programmer's Guide (*/docs/cpm3-pgr.pdf*) is the standard reference. Both of these can be found under /docs on your SD Card. Other possibly useful books on the SD card include a CP/M primer, Mastering CP/M and the Osborne CP-M User Guide.

- TechTinkering is a blog featuring videos and tutorials for various CP/M programs - `https://techtinkering.com/`
- The ClassicCMP Humongous CP/M Archive contains everything you might ever need - `http://www.classiccmp.org/cpmarchives/`
- The SD Card has a complete archive of the National CP/M Users Group at *B:ARC/CPMUG*. Various other user group archives from around the world are in various folders under *B:ARC*.

Chapter 4

Being Productive With Retro Zer0

You might think a computer running a 30-40 year old Operating System would have very limited abilities. Almost anything you can do in a terminal you can do on Retro Zer0.

While a 16Gb SD card might not sound like much, CP/M is tiny. In this chapter you will learn how to do word processing, spreadsheets and programming in different languages.

4.1 WordStar

WordStar is a professional Word Processing tool from the 1980s. George R.R. Martin uses WordStar on DOS for writing.

Figure 4.1: This manual being written with WordStar on a Retro Zer0

The WordStar environment is a nice, distraction-free way to write documents. Two versions of WordStar are available on Retro Zer0, WordStar 3 and WordStar 4. We'll use WordStar 4. From the command prompt, change directory to *B:APPS/WS4* and enter `WS4` to start.

Press 'N' to open a nondocument. Give it the name "letter.ws4", and press 'y' to create a new document.

Now type in some text over several lines. It doesn't really matter what it is, we're just playing around.

At the top of the screen you'll see the edit menu. This gives you some handy commands to move around. WordStar's movement functions are really, **really** cool. Lets take a look at how they work:

Figure 4.2: WordStar's diamond

Figure 4.3: WordStar secondary function keys

WordStar uses a diamond format for moving around. Use Ctrl-E to go up, Ctrl-S to go left, Ctrl-D to go right and Ctrl-X to go down. You can also use the cursor keys, but it gets better.

WordStar's secondary functions are all built around the diamond. Ctrl-W and Z scroll the screen up and down. Ctrl-R and C page up and down. Ctrl-A and F move the cursor left or right by a word at a time. Try moving around with the Ctrl-keys and see what you think.

WordStar has most of the features you'd expect from a

modern word processor, such as text manipulation, justification, centering and more. Some of WordStar's more interesting features include bookmarks (there can be 10 bookmarks in a document, great for moving around long documents), really good block handling for moving text around and mail merge features. The full manual including tutorials can be found on the SD card.

Printing doesn't work yet. You can print to an ASCII printer and it will produce an ASCII text file. Future versions of the Retro Zer0 will support transfer over USB Serial, and possibly printing via USART.

WordStar uses a special document format. Older versions of Microsoft Office and StarOffice can handle WordStar format natively. The format is mostly text, but may choke when handled without consideration. Conversion tools are available online for those not wishing to use MS Office 97 or StarOffice.

4.2 Supercalc

Supercalc is a professional Spreadsheet tool. In fact it was at one point the most popular spreadsheet around. Supercalc 2 is quick and has some nice features. Like WordStar it uses the same diamond for movement (or cursor keys if you prefer) and has excellent help features. SuperCalc2 is installed in "B:APPS/SC2". Change to the path and start SuperCalc2 with `SC2`.

You can enter "?" for an overview or press return to get straight to the spreadsheet view. Once you're on the spreadsheet view, press '/' to enter the slash command

Figure 4.4: Balancing a budget with SuperCalc2

prompt. Press '?" to get a list of what the possible commands are and what they mean, and then press a key to return to the spreadsheet view. Type'/lbudget,a'. This loads ('/l') the *BUDGET.CAL* sheet ('budget') in full ('a', or all).

The ">" prompt is for entering values and formulae into the highlighted cell. Press '?' from the ">" prompt to see a guide to "Initial Character Meanings". For example, "!" forces the sheet to recalculate in full.

Go to A14 and enter "SuperCalc2 is cool (with the double-quote). Now go to B14 and type *SUM(B6:B10)*. You should see the sum of all of the amounts.

Unfortunately the excellent 10 minutes to SuperCalc2 guide is no longer available. 10 Minutes to SuperCalc3 is on the SD card under /docs. Some features are missing (most notably graphs). This just means that instead of 10 minutes to SuperCalc3, you get 7 minutes to SuperCalc2 instead.

4.3 DBase

Ashton Tate's DBaseII was the dominant database tool in the 1980s. It's an incredibly powerful tool that has more in common with tools such as SAP than MySQL. It wouldn't be possible to do it justice here, but DBaseII v2.43 can be found in *B:DB/DBASEII*. A DBaseII guide is included on the SD Card under /docs, but also worth reading are:

- Programming with DBaseII
- DBase Demystified

There are many DBaseII databases ready to try under the Walnut and Oak CD archives under *B:ARC*.

4.4 Microsoft BASIC

To start Microsoft BASIC-80 5.21, use the `mbasic` command. This brings up the Microsoft BASIC interpreter. As well as Microsoft BASIC, the S-BASIC Compiler is also available in A:/SDK/SBASIC. Lets start with a simple BASIC program. Type in the following:

```
10 PRINT "In what year were you born?"
20 INPUT B
30 IF B < 1974 THEN PRINT "You're older than CP/M!"
40 IF B > 1974 THEN PRINT "You're younger than CP/M!"
```

Press return. Now type RUN. Your Retro Zer0 will ask your age and comment based on your response. If this is your first BASIC program, well done! BASIC is a fantastic programming language to start with, and you have one of the best versions of the day.

To return to CP/M type SYSTEM at the BASIC prompt. A selection of BASIC computer games are provided on the B: drive. Enter the following:

```
B:
CD /BASIC/CCGAMES/DISK1
MBASIC HAMURABI.BAS
```

Hamurabi is a classic BASIC computer game. Press CTRL-C to break out of the program, type `LIST` and press return. That scrolled quick! Type `SYSTEM` to return to the OS and try `PEEP HAMURABI.BAS`. Looking through the code it appears that:

- The `Y` variable defined on line 220 sets the price of land.
- The `A` variable is decreased on line 370 with the `A=A-Q` statement.
- Line 430 checks to see if we're trying to feed people more grain than we have.
- The `E` variable handles the rat population at line 690.

Playing with these lines and values will tweak different aspects of the game. Explore what they do, and what their impact is on the game. To test changes, enter `RUN`. To stop the running program press 'Ctrl-C'.

There are a lot of games and BASIC programs to try in the SD Card archives. A Basic-80 reference manual is available in the /docs folder on the SD card.

As well as the BASIC programs here, you might want to try your hand at some of the Usborne computer book series

Different computers used their own BASIC interpreters. As a guide, the TRS-80 listings will often be the closest version to MBASIC but may still need some tweaking to work. A version of BBC Basic for CP/M is available from R. T. Russell's site

4.5 Turbo Pascal

The Retro Zer0 comes with the best version of the best Pascal IDE for CP/M, Turbo Pascal 3. Pascal was invented in 1970 by Nikolaus Wirth and is still used professionally today. Borland Turbo Pascal is the predecessor of the modern Delphi RAD tools suite. Pascal is influenced by an earlier language called Algol, also available for CP/M.

```
A:
CD SDK/TURBO3
TURBO
```

Press 'y' to include error messages and you'll see the main menu. We need a work file, so press 'w' and provide the filename 'test.pas'. To enter the editor, press 'e'. Type in the following program:

```
program myName;
```

```
const
  TotalTimes = 20;

var
  Name : String[25];
  NumberOfTimes : Integer;

begin
  Write('What is your name? ');
  Readln(Name);
  ClrScr;
  for NumberOfTimes:=1 to TotalTimes do
    begin
      Writeln(Name, ' is cool!')
    end;
end.
```

After typing that in press CTRL-K to switch to command mode, and type ‘d’, then press return to get a command prompt. Type ‘s’ to save, then ‘r’ to compile and run the program. If you get an error, compare your typing to the listing here. When you’re done press ‘q’ to return to the command prompt.

If you’d like to learn Pascal, the full Turbo Pascal Manual and Turbo Pascal Tutorial docs are on the SD Card under /docs. There are many several large libraries of Pascal code, a couple of Turbo Pascal tutor programs and documents under B:ARC. If you’d like to do Pascal the right way, read Stan Sieler’s How to code: Pascal

TechTinkering has a fantastic Turbo Pascal for CP/M tutorial

There’s also an excellent collection of Turbo Pascal code to look at under *B:ARC/WALNUTCD/cpm/turbopas*.

4.6 Forth 83

Most programming languages follow common conventions. Forth is not like most programming languages and is more a way of thinking that's very different. Although there is an ANSI Forth specification, most Forth implementations are different to each other. Don't be disheartened if things don't work first time. Persevere and you'll get there in the end.

```
B:
CD /SDK/CPMFORTH
F83
```

Henry Laxen and Michael Perry's F83 Forth is a Public Domain Forth Implementation. Forth uses a stack-based model and Reverse Polish Notation (RPN). This might feel little confusing at first, but lets start with some simple ideas. To add two numbers, RPN places the two numbers then the operator, like so:

```
2 3 +
.
```

The '.' symbol reveals the content of the stack and 'pops' it off. If you try '.' again you should get a Stack Undeflow error. RPN has no operator precedence. Instead numbers are added onto a stack and processed from left to right. For example:

```
2 3 4 + *
```

In this example, the following happens:

1. 2 is added onto the stack. Our stack contains the number 2.
2. 3 is added onto the stack. From top to bottom, our stack now reads 3, 2.
3. 4 is added to the stack. From top to bottom, our stack reads 4, 3, 2.
4. A + operator is encountered. This takes two arguments. The first two stack entries are 'popped' off and added together, 4 + 3 = 7. 7 Is added to the stack. Our stack now reads 7, 2.
5. A * operator is encountered. This takes two arguments. The first two stack entries are popped off and multiplied, 7 * 2 = 14. 14 Is added to the stack. Our stack just contains 14.

It isn't as complicated as it first seems. It's just very different to what we're normally used to.

Steven P. Curtis wrote a great post on visualising RPN

Lets try something more interesting. Enter `STAR`. You should get an error as F83 doesn't have that word defined. The ASCII character code for '*' is 42 in hexadecimal. We can define our own words in FORTH, like so:

```
: STAR ( -- )
42 EMIT ;
```

Type `STAR` again, and you'll see '* ok' appear. Type `STAR STAR` and two asterisks will appear. So far, so underwhelming. Lets do something else:

```
: STARS ( n -- )
0 DO STAR LOOP ;
```

Now type `5 STARS`, and five stars will appear followed by 'ok'.

```
: TRIANGLE ( n -- )
1 + 1 DO
I STARS CR
LOOP ;
```

Now type `5 triangle` and a triangle will appear. Forth suits itself well to a particular way of thinking. Leo Brodie's Starting Forth and Thinking Forth are probably your best starting points. These are included on the SD card under /docs, along with a copy of Dr. C. H. Ting's Inside F83, 4th Edition.

If that's not enough, the Forth Interest Group have a selection of tutorials.

If on the other hand that's quite enough, enter `bye` to quit F83.

4.7 ZDE

ZDE Is an Integrated Development Environment (IDE). The `zde16` editor lives under *B:/SDK/ZDE* but the path has been added to the profile so it can be called anywhere via `zde16`. If things look garbled, then the terminal settings for ZDE can be changed with `ZDENST16 ZDE16`. ZDE is useful as a general editor as well as a code editor. If you supply a filename as an argument zde16 will open the file, otherwise you'll get a blank editor to work with.

The following keys may be useful:

- ESC h - Bring up the help screen
- ESC q - Quit without saving
- ESC x - Save and Exit
- ESC l - Load file
- ESC s - Save file
- CTRL-r - Page Up
- CTRL-c - Page Down

The WordStar Diamond controls also apply here, although standard cursor keys work too:

- CTRL-e - Up
- CTRL-x - Down
- CTRL-s - Left
- CTRL-d - Right

A '<' symbol will be added at each newline. This isn't in the document, it's just to show you where a line ends.

4.8 Z80 Assembler

Z80 Assembly sounds scary but is fairly simple and easy. As a test of ZDE we'll create a simple assembly program that prints out a message.

```
C:
zde16 hello.z80
```

The ZDE interface will appear and we can start typing away. Enter the program below:

```
; org sets the starting address to 0100 in hexadecimal
org  0100h

; Set up the print call
       ld   de,msg   ; Load the address of "msg" into de
       ld   c,9  ; Load c with BDOS call 9 (print string)
       call 5    ; Execute BDOS call

; back to the OS
       ld   c,0  ; Load c with 0 (return to dos)
       call 5    ; Execute BDOS call
       halt

msg:      db   "Hello, World$"
       end
```

CP/M uses dollar-symbol terminated strings by default, although this can be changed with BDOS calls. A much fancier version of this code using traditional NULL-terminated strings can be found at Symbolic Debugger's blog. BDOS stands for Basic Disk Operating System, and a BDOS call is a way of asking BDOS to do something like print a string. This makes our program incredibly small.

Once you've entered the code, use 'ESC x' to save and quit. Now it's time to assemble some z80.

```
z80asm hello
```

Now enter `hello.com` to run the program. If you'd like to learn Z80 Assembly, several guides can be found on the SD card in */docs* along with the Z80ASM User Guide. The CPM Plus Programmer's Guide also provides details of all of the BDOS calls. I'd also recommend Rodnay Zaks' book (/docs/zaks_book.pdf) on the SD Card if you really want to get into Z80 Assembly. Now for something easier it's time to do some reverse engineering!

For this we can use the Multisession capability. Open up hello.asm in ZDE16 in one session for review, and switch to another for the reversing. This way we can compare source to object code.

Although `z80dis` can disassemble our code, we'll use the `zsid` debugger instead.

```
C:
ZSID HELLO.COM
```

Enter `l` to list the disassembly. You'll note that references to labels such as vprint have been replaced with addresses. You can switch back and forth between sessions using the function keys. At address 'CALL 010C' appears at address 0103, but it isn't shown in the listing. To see the contents of memory at 011B, enter `d011B`. You should see a hex dump with ascii text on the right. The first line will say 'Hello, World!'. If you type `l` you'll see the location has changed. To see the listing from the start of the program enter `l0100`. When you're ready, enter `g` to execute the program, which will return to the Operating System when finished.

If you just want to get a hex dump of a binary file, you can use the dump command. For example, to get a hex dump of our program, enter `DUMP HELLO.COM`.

The SD Card has documentation for all of the tools used above. You can also use tools such as Z88DK to cross-compile for CP/M from more powerful systems.

Lets do one more thing in assembly. Although Retro Zer0 uses a screen, CP/M thinks we're using a terminal and to do terminal-based housekeeping it uses terminalcodes. These do things like moving the cursor around the screen. There are different standards but some are common across most. We can modify our *HELLO.Z80* program to create a useful command to clear the screen and move the cursor to the top left.

The control codes we will send are:

```
27   ; Escape code (tells the terminal what follows is a command)
[2J  ; Clear the screen
27   ; Escape code
[H   ; Move the cursor to the top left of the screen
```

As you might imagine these codes can get rather complicated, but you don't have to worry about it. copy *HELLO.Z80* to *CLS.Z80*. Open up *C:CLS.Z80* in `zde16` and change the line starting with msg from:

```
msg:      db   "Hello, World$"
```

to:

```
msg:      db   27,"[2J",27,"[H$"
```

Save, exit and assemble by entering `z80asm cls`. Give it a try. If you find the command useful copy it to *B:BIN*. You can get a

hex dump of the binary by running `DUMP CLS.COM`. If you look at the binary on another computer you'll see it's 128 bytes in size, but only a few bytes are actually used.

Your `CLS` command should work on every terminal setting except VT52, which uses a different terminal command set. Why not see if you can modify the program to work on all supported terminal types?

4.9 Hi-Tech C

Hi-Tech C is an ANSI C compiler for CP/M. While it predates the C89 standard, it's ANSI compliant. Most simple software can be made to compile with few, if any tweaks. Hi-Tech C compiles code to Z80 COM files, and the output should work on most Z80 CP/M 3 systems. To test Hi-Tech C, write a C program to explore the Fibonacci Sequence.

The Fibonacci sequence is a series where each number is the sum of the two previous numbers. The Fibonacci sequence is strongly related to the golden ratio, commonly depicted as a spiral that shows up everywhere. The first 9 numbers in the sequence are:

0, 1, 1, 2, 3, 5, 8, 13, 21

The Fibonacci sequence is a good way to show off C as it tends towards implementation based on recursion. Recursion is where something is defined in terms of itself. In Computer Science, recursion is normally used to describe a function applied within it's own definition. For more information on recursion, re-read this paragraph.

We'll write a recursive fibonnaci program that will call itself to calculate the sequence. It will be much slower than if run on a regular computer. Why not try your code on a Raspberry Pi and see if you can spot the difference?

```
#include <stdio.h>
#include <stdlib.h>

int fib(int n) {
     if(n == 0){
          return 0;
     } else if (n == 1) {
          return 1;
     } else {
          return (fib(n - 1) + fib(n - 2));
     }
}

int main(int argc, char *argv[]){
     int n,i;
     if (argc !=2){
          printf("Usage: fib <n>\n");
          printf("Where <n> is the number of items\n");
          return 1;
     }

     n = atoi(argv[1]);

     printf("Fibbonaci of %d: ",n);

     for(i = 0; i < n; i++){
          printf("%d ", fib(i));
     }
}
```

Our C program has two functions, main() and fib(). The main() function is called at the start of a C program. This calls our fib() function. The fib() function checks to see if it's been called with 0 or 1 and calls itself to add the last two numbers. If you're new to C it might be a bit of a head scratcher, but by tweaking the code you should be able to figure out how it all works. For example, you could add printf statements to show what's being called and when or you could use `zsid` to step through the compiled program and inspect the memory as it changes.

Assuming you saved the file as *C:FIB.C*, enter the following from the command prompt to compile your program:

```
B:
CD /SDK/HITECHC
C -V C:FIB.C
FIB 9
```

Check the output against the first 9 Fibonacci numbers earlier. If it matches, your program is doing it right. If you get compiler errors, check the code in ZDE matches the listing. It's not uncommon to find dangling braces (squirly brackets) or missing semi-colons sending C programs astray. If you run `fib` on its own, you'll get a nice usage message. Try `FIB 25`. You'll notice the number turns negative. Why is that? Can you fix it?

If you'd like to learn more, the Hi-Tech C compiler manual is on the SD card, along with a copy of Kernighan and Ritchie's The C Programming Language, 2nd Edition which is a period-specific C programming book worth reading. C Has changed a lot since the 1980s. Modern tutorials might need some work to reimplement. Still, there's lots of fun to be had learning C on a Retro Zer0.

4.10 XLisp

Lisp is the second oldest high-level programming language in the world after Fortran, but is still used today. The Retro Zer0 includes David Betz' XLISP 1.1 in "B:SDK/XLISP". XLisp can be run either interactively or against files. To try it out, use ZDE to create C:HELLO.LSP, and add the below:

```
(princ "Hello, World!\n")
(exit)
```

Save the file and exit ZDE. To run it, enter the following:

```
B:
CD /SDK/XLISP
XLISP11 C:HELLO.LSP
```

As a result of changes over time, the version of XLisp differs slightly from modern common lisp implementations. An XLisp manual can be found under /docs on the SD Card.

4.11 Installing New Software

Software is still being released for CP/M today. There are developers around the world writing programs and applications for a whole range of purposes many would've never conceived of when CP/M was first written in 1974. Places you can find software include:

- The Humongous CP/M Software Archives
- Gene Buckle's CP/M Retro Archive
- PC Sauro's CP/M World
- Z80.eu's CP/M archives
- The Tim Olstead CP/M Web Site

Usenet's Comp.os.cpm is another good place to find CP/M software, and discuss CP/M if you have a suitable Usenet client.

We will download and install zcsoli, a CP/M Solitaire game. Download ZCSoli on your main computer, and unarchive it onto your SD card under the */driveC* directory. Then eject your SD card, put it in your Retro Zer0 and power up.

Switch to *C:* and enter `zcsoli` to play. The game works with VT100 and ADM3A-based terminals so there's no need to change emulation. If you'd like to keep it on your Retro Zer0, copy *ZCSOLI.COM* to *B:GAMES*. You can now delete the contents of the *C:* drive to clean up.

ZCSoli was written in Z80 assembly language and is incredibly well commented. It's designed to be cross-compiled. As such, the carriage returns and line feeds aren't quite right for CP/M. You can view the source in an editor such as Notepad++ on Windows or nano on Linux.

4.12 Going Online

The core project the Retro Zer0 is an offline-first system. The on-board ESP32 can provide Wifi connectivity, and MultiCPM supports it. You can use your Retro Zer0 to access systems over telnet. Combined with the ANSI support this makes for an authentic feeling BBS experience.

Check that your network can be found with `WIFISCAN`. The ESP32 only supports 2.4Ghz wifi and cannot connect to 5Ghz-only networks. MultiCPM assumes a WPA-PSK wifi network, and cannot connect to WPA-EAP networks. MultiCPM also assumes a DHCP server is accessible over WiFi and is IPv4 only.

To connect to a wifi network, use the `wifi` command. For example, to connect to a 2.4Ghz WPA-PSK wifi network with an SSID of 'Pretty_Fly_For_A_Wifi' and password of 'password', enter:

```
wifi Pretty_Fly_For_A_Wifi password
```

If the connection is successful you'll see the IP address given by DHCP.

The `ping` command can be used to check connectivity over ICMP, while `telnet` can be used to connect to a host. A port can be specified as a second argument to the telnet command. To quit a telnet session, use `CTRL-C`.

4.13 What Next?

Hopefully this chapter has given you some inspiration to start creating with your Retro Zer0. No matter what you need to do, there's usually a tool to do it. A plethora of software and code can be found under *B:ARC*. Whether your want to manage a Church membership system in DBaseII, calculate finances in Supercalc or MBASIC, calculate RF propagation or just play games, the software will be on your SD card. The rest is up to you!

Chapter 5

Tinkering With Your Retro Zer0

5.1 Understanding Your Retro Zer0

The Retro Zer0 is based on the ESP32 Microcontroller and runs a modified version of Fabrizio Di Vittorio's Multisession CP/M 3 (Plus) Compatible System. The modifications are mostly cosmetic, but also cover the extra C: drive. This is not the only software the Retro Zer0 is capable of running and you are encouraged to hack this thing in any way you desire.

The Retro Zer0 is designed to be simple and is based on a broader project looking at long-term computing with replacement parts. As such every part on your Retro Zer0 can be replaced, and there is no reason that your Retro Zer0 won't last 50-100 years with appropriate care and servicing.

To repair part of your Retro Zer0, desolder the part and resolder a replacement. A full schematic and circuit layout plot are provided under /docs on the SD Card.

The ESP32 is the brains of the operation. It's a feature-packed microcontroller, most of which aren't being used. The ESP32 emulates a Z80 CPU and enough parts to run CP/M, and interfaces with on-board i/o.

The VGA Connector is a Norcomp 181-015-213R561 connector and can easily be replaced with almost any right-angled VGA connector desoldered from a motherboard elsewhere. VGA is a common standard, so any source that implements it should work, although it may need some cutting or sawing to fit in place.

Next to the VGA connector are 6 resistors, they provide 2x Red, Green and Blue outputs for a 6-bit 64 colour palette. R5, 7 and 9 are 806 ohms while R6,8 and 10 are 402 ohms. The values aren't really that important from a CP/M perspective although they will affect colours. Each pair of resistors is configured as a voltage divider to provide half or full brightness levels.

On the other side an MD-60S PS/2 connector is powered by the 5v ESP connector through a voltage divider. This goes through a mildly complex setup involving a 120k and 2k resistor for each of the PS/2 Clock and Data lines. These resistors must be at the correct value or PS/2 performance may be affected.

An Amphenol 10067847-001RLF SD Card slot completes the peripheral setup. No resistors are required for this. However, the component is surface mount. To replace this, desolder the part and optionally bodge wire a jig to a standard SPI layout.

From there it should be possible to solder on a replacement SD Card adapter breakout, such as one from Adafruit. Alternatively it is possible to solder SD and Micro SD cards directly to SPI headers and they will work. This does sacrifice portability.

A power LED and 330 ohm resistor are provided to signal that the system is on. In the event of the system losing power, the LED will not light up.

5.2 The USART Header

There is a line of 4 holes at one end of the board for a header to be soldered on. This is wired up to the ESP32's built-in USART and operates at 3 volts only. Do not connect 5v components without a level shifter. The power goes directly to the 3v3 and common ground so be careful using this when powered over USB.

Nothing has been set up for the USART header, although this may change in future software revisions. The holes are there for when this starts being used.

5.3 Reprogramming Your Retro Zer0

The ESP32 breakout is the DevKit-C board. You can reprogram this using the arduino interface, platformio or even the esp-idf tools. Obviously reprogramming this overwrites the firmware, but there's a copy on the SD Card under /fw.

Copy the MultiTaskingCPM directory to your Arduino projects folder. Configure your Arduino IDE as shown in the screenshot above. You'll need to install the FabGL Arduino Library and remember to use the local MultiTaskingCPM one, not the one in the FabGL examples.

Bitluni's ESP32 VGA libraries will work but require substantial tinkering to function. FabGL demos including the VIC-20 emulator and Altair 8800 emulator will work perfectly fine, although the Retro Zer0 lacks a mouse and sound port.

5.4 Troubleshooting Hardware

Although the Retro Zer0 is user-friendly, you will at some point have problems with it. These may help.

5.4.1 Text Keeps Dropping Off The Screen/Weird Positioning

Adjust your monitor height and width settings. The default resolution is 640×480 at 60hz. Most monitors have an auto-adjust setting.

5.4.2 No Screen Output

Check your VGA cable connections both to the monitor and the Retro Z3r0. Make sure Monitor inputs are configured for VGA and not HDMI or DisplayPort. If the monitor is definitely

on, set to VGA and definitely connected, the question is whether this is really a screen issue or it's somewhere else.

When the Retro Zer0 is powered up the keyboard lock keys should flash on the PS/2 keyboard. If they don't it's not the screen at fault. Wait a few seconds for boot to complete, press caps lock. If the caps lock light doesn't toggle properly you have other problems.

Connect the Retro Zer0 to a computer and use a terminal tool such as the Arduino Serial Monitor or Minicom set to 8/N/1 at 115200 baud. On reset you should see "Reset" appear over Serial. If you do, this means that it's booting. Keep an eye out for references to SD cards or SPIFFS.

If the Retro Zer0 boots up successfully but nothing is displayed, try a different VGA cable and monitor. If you still can't get anything, inspect the solder joints for gaps and bridges. Test continuity between the pins on the underside of the VGA connector and the appropriate ESP32 pins. Check the resistors close to the controller too. If the HSYNC or VSYNC lines aren't sending a signal all the way through, you won't get a usable screen.

5.4.3 Keyboard Doesn't Work

Check the PS/2 Connector is correctly connected. It should be fully inside the PS/2 Socket and can only be oriented one way to fit. On a Periboard keyboard the correct orientation leaves the imprint of a keyboard facing up at the connector. DO NOT FORCE THE KEYBOARD CONNECTOR INTO THE PS/2 SOCKET. We have killed too many prototypes doing this in the dark. Treat it like an elderly grandparent.

If it's just one key it may be the keyboard is duff. In this case whole rows may also stop working. Try another PS/2 Keyboard. If this doesn't work, consider recompiling the CP/M implementation and remapping keys to working ones.

If the characters on the screen don't match key presses, you probably have the wrong keyboard layout. Try typing the first 6 characters from 'Q'. If you get 'QWERTZ', ihre Tastatur ist auf Deutsch eingestellt. The `keyb` command can be used to change keyboard settings. By default, the keyboard layout is set to UK in *A:PROFILE.SUB*. To switch to a US keyboard layout, enter `KEYB US`.

5.4.4 SD Card Doesn't work

If you use the serial monitor as described in 'No Screen Output' you will see 'mountSDCard Succeeded' if the SD Card is mounted. If there is no SD Card detected, the system will fall back to a minimal CP/M Plus implementation in flash memory. While this is fairly cool it does miss all the extra toys we've added. If you don't have that option, this is easily explained by the C: drive being missing and most of the software on B: and A: gone with it.

If you use a fresh SD Card with no folders, the A: and B: rom images should be copied to the SD card. You can then add programs and data accordingly. Always back up your SD cards.

If the SD Card is sometimes detected on boot but sometimes not, try pushing on the top of the adapter and resetting. Keep a finger or something mildly heavy on the adapter. If it detects the SD Card correctly, you may have a loose or corroded

contact. Check the solder points for damage and resolder where needed.

5.4.5 No Power Lights

If the LED next to the notch doesn't light up, it could be the LED at fault. If the LED on the ESP32 doesn't light up, you have problems. Check the USB connector on the DEVKIT-C board to make sure power is flowing through the board. Check the 5v, 3v3 and GND pins with a multimeter, and test different points on the board to make sure power is coming through. Over time the PCB may fail in some areas. It may be necessary to use bodge-wires to recreate corroded or degraded tracks.

5.5 Modding Your Retro Zer0

5.5.1 LED Mod

The LED next to the hole is a standard red LED connected to the 3v3 line with a 330ohm inline resistor. This can be desoldered and replaced with anything. We recommend avoiding blue LEDs as they need more power than most, and choosing low-power LEDs where possible to reduce current drain. Orange, yellow and low-power green LEDs are fine.

5.5.2 Using Pins

The following pins are used specifically for CP/M:

- 13 (Carrier Select on SD Card)
- 14 (SD Card Clock)
- 33 (Keyboard Clock)
- 32 (Keyboard Data)
- 23 (VGA Horizontal Sync)
- 22 (Red 1)
- 23 (UART TX)
- 24 (UART RX)
- 21 (Red 2)
- 19 (Green 1)
- 18 (Green 2)
- 5 (Blue 1)
- 17 (SD Card MOSI)
- 16 (SD Card MISO)
- 4 (Blue 2)
- 15 (VGA Vertical Sync)

The following pins are reserved for other purposes:

- 27 (2nd PS2 Data)
- 26 (2nd PS2 Clock)
- 25 (DAC 1 out)

The other pins are available for general use, although two pins have yet to be defined for use with an MCP23017 GPIO port extender that may affect future builds.

Chapter 6

Whose Fault Is This?

Concept, Electronics, Hardware Steve Lord

Build and Documentation Steve Lord and Marizel Fourie

Special Thanks To:

Fabrizio Di Vittorio - For his ESP32 VGA work and the beautifully coded CP/M emulator

Bitluni - For his work on ESP32 VGA and informative videos

Saumil Shah - For making the Retro Zer0 possible

Gary Kildall - For CP/M. I hope you would've liked this.

6.1 Software Used

Documentation - Sphinx, Xelatex, Pandoc, Neovim, WordStar 4, RSMS' Inter font

Hardware Design - KiCAD, GIMP

Software Design - Arduino IDE, Platform.io, ESP-IDF

6.2 Contact Details

Web: `https://ringzer0.training/` and `https://rawhex.com/`

Mail: `support@rawhex.com`

Post:

Raw Hex Ltd
Unit 8b
Basepoint Business Centre
Stroudley Road
Basingstoke
United Kingdom
RG24 8UP

Updates to this handbook will be published at `https://ringzer0.training/`