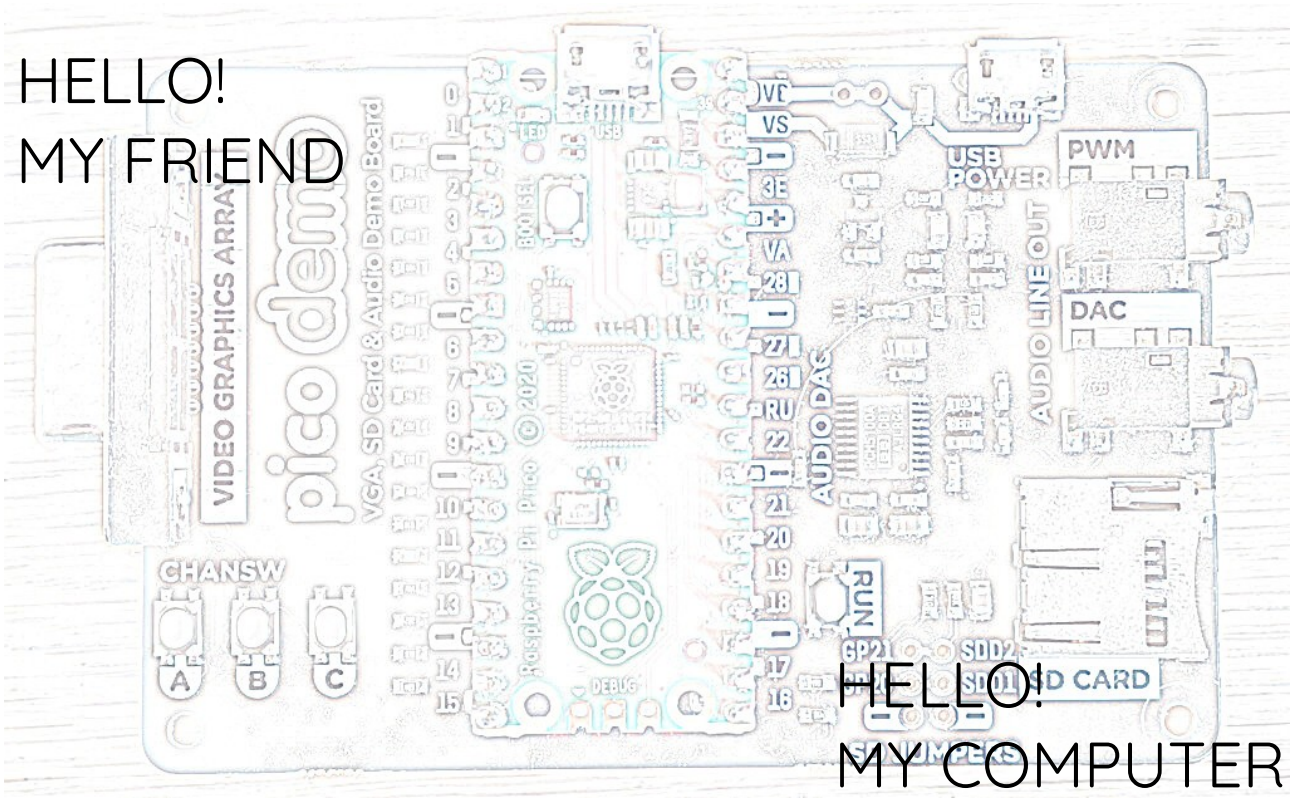
Pico MZ-80K

User and Systems Manuals



Pico MZ-80K Manual, Version 1.0.5, November 2024.

Table of Contents

Introduction

The Sharp BASIC manual from 1979 introduces the user to the MZ-80K in this way.

> **Here’s a new friend for you**
>
> The MZ-80K is ready to enjoy conversation with you. Through conversation, it will help you solve difficult calculation problems or become a partner to play a game with. More than that, it has unknown potentialities to be opened up with you. This is just like a journey into unknown space. Together with your new friend, let’s make the journey now.

The Pico MZ-80K aims to faithfully re-create this iconic computer so that your conversation can carry on more than four decades after the journey began.

But first, you will need to understand a little more about how to set up the Pico MZ-80K, so that it can be your new friend.

Getting started

Hardware requirements

The Pico MZ-80K uses the Pimoroni VGA demo base with a Raspberry Pico microcontroller to re-create the hardware of a Sharp MZ-80K. To run the emulator, you will need:

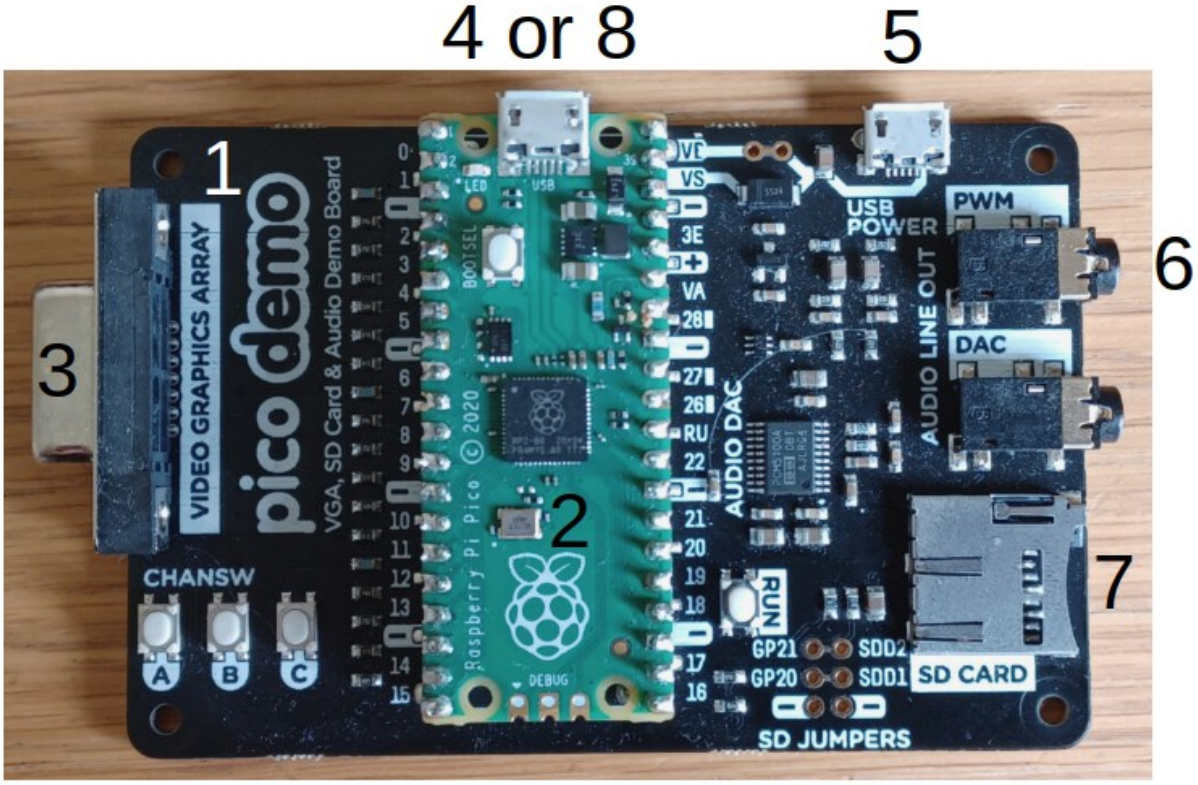
1. A Pimoroni VGA demo base

2. A Raspberry Pico H (or solder headers to a standard Pico).

3. A VGA cable to enable your VGA demo base to be plugged into a suitable monitor.

4[1]. An OTG adaptor or cable to allow a USB keyboard to be plugged into the micro USB port on your Pico.

5[2]. A power supply. This must be plugged into the micro USB port on your VGA demo base marked ‘USB POWER’. An official Raspberry Pi 5V, 12.5W Micro USB Power Supply is suitable.



6. A speaker or speakers, connected to a 3.5mm stereo jack, and plugged into the PWM socket on the VGA demo base. The DAC socket is not currently supported.

1 An alternative to using a separate keyboard and power supply is to connect the Pico’s USB port to a computer’s USB port. A terminal emulator, such as minicom, can then be used to provide input to the emulator using the computer’s keyboard. If the debug version of the emulator is to be used, this type of setup is essential. See the section on using a terminal emulator with the Pico MZ-80K later on in this manual for more details.

2 As footnote 1. A separate power supply is not used if the Pico’s USB port is connected to a computer.

7. A FAT32 formatted microSD card[3], containing the Sharp MZ-80K software you wish to run. These should be ‘.mzf’ format files. Other formats, such as .wav, .mzt and .m12 are not currently supported by the emulator.

Good sources of .mzf files include:

https://sharpmz.no/original/

https://mz-archive.co.uk/

https://github.com/psychotimmy/sharpmz-80k

As a minimum, a language interpreter (such as Sharp BASIC SP-5025) or Z80 development environment (such as Avalon ZEN) should be written to the microSD card. The Pico MZ-80K is of little use without an interpreter or development environment.

A copy of the Sharp MZ-80K SP-5025 BASIC manual can be found at:

https://archive.org/details/sharp-basic-manual-mz-80-k

8. A USB cable with a micro USB plug for the Pico, to enable the Pico MZ-80K software to be installed and re-installed from a computer.

3 See the section on microSD card support for known working / not working microSD cards.

Installing / re-installing the Pico MZ-80K software

The most recent release of the Pico MZ-80K software can be found at:

https://github.com/psychotimmy/picomz-80k

To install the software onto the Pico, download picomz-80k.uf2 or picomz-80k-diag.uf2[4] and:

1. Push and hold the BOOTSEL button while connecting your Pico with a USB cable to a computer. Release the BOOTSEL button once your Pico appears as a Mass Storage Device called RPI-RP2.
2. Copy the .uf2 file onto the RPI-RP2 volume. Your Pico will reboot.
3. Disconnect the USB cable and plug in the OTG adaptor and USB keyboard to the Pico. Apply power to the USB POWER port. Alternatively, use a terminal emulator from the computer the .uf2 file was copied from to provide keyboard input to the Pico MZ-80K.
4. You are now running the Pico MZ-80K emulator.

First boot

If everything is working as expected, your VGA monitor should display:

```
** MONITOR SP-1002 **
*
```

Use the F1 key to cycle to a m/c code ‘tape’ to load (for example, the SP-5025 BASIC interpreter).

Type LOAD <return>

The ‘tape’ will take some time to load (although not quite as long as a real tape on a Sharp MZ-80K).

If you choose to load the SP-5025 BASIC interpreter, you should see the message:

```
* SHARP BASIC SP-5025
34680 BYTES
READY
▒
```

4 picomz-80k.uf2 requires a physical UK USB keyboard to be plugged into the Pico. If you are going to be using a terminal emulator (such as minicom) from another computer to provide keyboard input, you will need to use picomz-80k-diag.uf2 instead.

Keyboard layout

The MZ-80K keyboard layout

The Sharp MZ-80K keyboard has 78 keys, arranged in five rows. There are 14 keys on the bottom row, and 16 keys on each of the other four.

The rightmost 5 keys on each row are blue and allow access to 75 of the Sharp's character graphics.

The remainder of the bottom row are yellow, and implement the space bar, carriage return, shift keys, break key, cursor movement and editing functions.

The first 11 keys on each of the other four rows (with the exception of the 11th key on the second row from the bottom – SML/CAP) implement alphanumeric and punctuation characters. By default, the alpha characters are upper case.

The shift key enables the character on the top of the key to be used – for example, shift Q returns >, and shift S the heart graphic.

The SML/CAP key allows lower case characters to be used on the alpha keys or if a symbol is printed on the lower right hand side of a key, that symbol. If the SML/CAP key is selected, the led to the right of the MZ-80K keyboard (usually green when power is on) is turned red. When deselected, the led returns to green.

Mapping the MZ-80K keyboard to a UK USB keyboard

Black MZ-80K keys

With Caps Lock off, the lower case alpha keys on a USB keyboard are mapped to the upper case alpha keys on the MZ-80K keyboard. Numeric keys are mapped to the numeric keys as expected.

With Caps Lock on (or shift <character> used), most of the alphanumeric keys on a USB keyboard are mapped to the shifted MZ-80K keys (for example, shift <1> maps to !, shift <Q> maps to > and shift <S> maps to the heart symbol).

The exceptions are:

<shift> 3 – maps to £, rather than #

<shift> 6 – maps to π, rather than ^

<shift> 7 – maps to &, rather than ‘

<shift> 8 – maps to *, rather than (

<shift> 9 – maps to (, rather than)

<shift> 0 – maps to), rather than π.

Where a punctuation symbol appears on the USB keyboard and there is a match on the Sharp MZ-80K keyboard, that key corresponds to the Sharp MZ-80K key. For example, ‘:’ maps to ‘:’ (<shift> O on the MZ-80K keyboard) and ‘@’ maps to ‘@’ (<shift> U>).

The SML/CAP key is mapped to the ‘~’ (tilde) key. When selected, the Pico’s green led is lit. This is equivalent to the Sharp MZ-80K’s power led turning red (from green). When the SML/CAP key is deselected, the Pico’s led is turned off.

Yellow MZ-80K keys

The yellow keys are mapped as follows:

Left hand shift key – mapped to either USB shift key. Because shifted characters are taken care of automatically, use <Ctrl> L if a program is expecting **only** a left hand shift key as input.

Right hand shift key – mapped to either USB shift key. Because shifted characters are taken care of automatically, use <Ctrl> R if a program is expecting **only** a right hand shift key as input.

CLR – mapped to the End key

HOME – mapped to the Home key

INST – mapped to the Insert key

DEL – mapped to the Delete and Backspace keys. <Ctrl> H will also work.

SPACE – mapped to the spacebar

Cursor up, down, left, right – mapped to the cursor up, down, left, right keys respectively

BREAK – mapped to the PgDn key

SHIFT BREAK – mapped to the PgUp key

CR – mapped to the carriage return key and the numeric keypad’s Enter key. <Ctrl> M will also work.

Blue MZ-80K keys

The blue graphics keys are mapped to Alt keys as shown in the table below.

Alt Q Top left blue key	Alt W	Alt E	Alt R	Alt T Top right blue key
Alt Y	Alt U	Alt I	Alt O	Alt P
Alt A	Alt S	Alt D	Alt F	Alt G
Alt H	Alt J	Alt K	Alt L	Alt M
Alt Z Bottom left blue key	Alt X	Alt C	Alt V	Alt B Bottom right blue key

In common with the black keys, <shift><Alt><key> selects the symbol on the top of the MZ-80K key. If SML/CAP is active (steady green led lit on Pico), the symbol on the bottom right of the MZ-80K key is selected instead.

Function keys

The function keys allow the following tasks to be accomplished.

F1 – Step ‘forwards’ through files on the microSD card ‘tape’.

F2 – Step ‘backwards’ through files on the microSD card ‘tape’.

When F1 or F2 are pressed, the next tape file that a LOAD command will use is displayed in the emulator status area.

F3 – Reset the tape counter to 000.

F4 – Clear the emulator status area.

F3 and F4 do not affect the next tape file that a LOAD command will use.

F9 – Change the keyboard scan mode. The default is 1, which is fine for most programs. However, some programs (usually games written in machine code) require the scan mode to be set to 2 or 3 to ensure that keypresses are reported correctly. When F9 is pressed, the keyboard scan mode (1,2 or 3) is displayed in the bottom right of the emulator status area.

F11 – Read a memory dump file (MZDUMP.MZF) from the microSD card. This restores the state of user RAM, video RAM and the z80 to the point at which the memory dump was created.

F12 – Store the contents of user RAM, video RAM and the z80 state to a memory dump file (MZDUMP.MZF). Note that the previous contents of this file are always overwritten.

F5 to F8 and F10 - Currently unused.

Cassette tape emulation

Loading files from a microSD card

The microSD card acts in much the same way as a cassette tape works on a real Sharp MZ-80K.

Use the F1 key to position the tape read head at the start of a new file. This is the equivalent of using the fast forward key and tape counter on a real machine. Repeatedly pressing F1 will cycle forwards through all of the files on the microSD card before stopping at the last file.

Use the F2 key to position the tape read head at the tape file before the current one. Repeatedly pressing F2 will cycle backwards through the files on the microSD card until the first file is reached.

F3 will position the tape read head at the first file on the microSD card.

The bottom five lines of the Pico MZ-80K's display are used to display 'tape' status. The name of the next file to be loaded is displayed (note that this is not necessarily the same name that the file has on the microSD card), along with the file type (one of m/c code, BASIC etc. data or unknown).

Files of type m/c code must be read directly from the SP-1002 monitor.

Files of type BASIC etc. must be read from the appropriate interpreter or development environment.

Files of type data are for use by the originating program. For example, the game "The Valley" allows your character to be saved and loaded from tape. These are stored in files of type data.

Use the LOAD command when in the SP-1002 monitor or BASIC SP-5025 (or the equivalent if you are using another interpreter or development environment) to transfer this file into the Pico MZ-80K's memory.

Unlike on a real Sharp MZ-80K, using the LOAD command simulates you pressing the PLAY button on the cassette deck automatically, and stops once the end of the file is reached.

Loading files from the microSD card takes a little time as it is emulating a real tape. However, tapes are read slightly more quickly than on a real machine.

The F4 key will clear the 'tape' status display until the next time F1, F2 or F3 is pressed.

Saving files to a microSD card

Use the SAVE command when in BASIC SP-5025 (or the equivalent if you are using another interpreter or development environment) to write the contents of the Pico MZ-80K's memory to the microSD card.

The name of the file saved to the microSD card is not necessarily the same as the name given to the SAVE command. This is because the permitted characters in a FAT32 file name are not equivalent to the ones permitted in Sharp MZ-80K file names (and vice-versa).

Note that if a file on your microSD card already exists it will be overwritten.

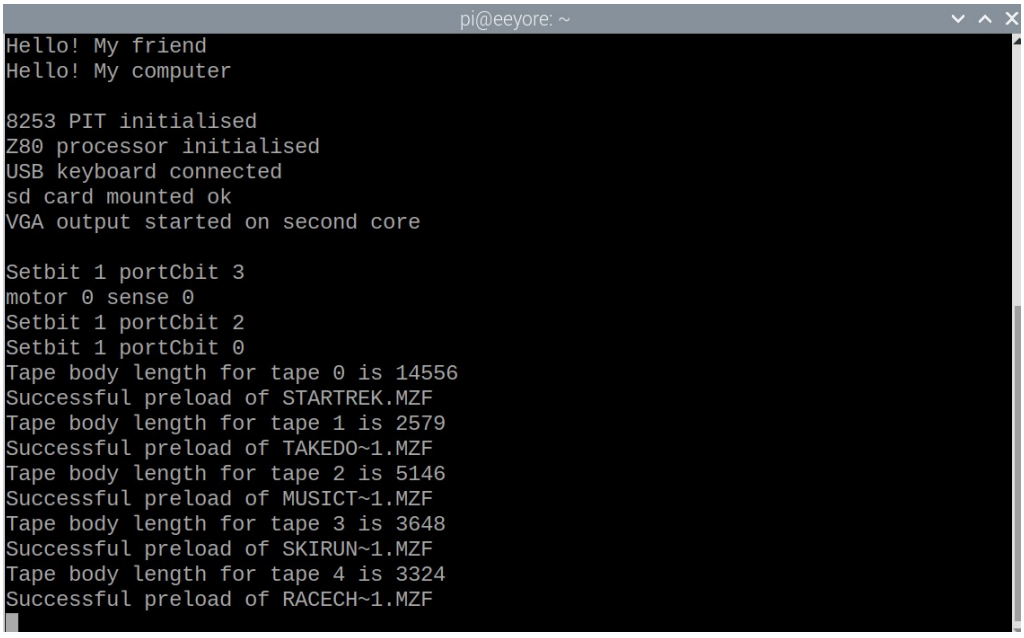
Using a terminal emulator with the Pico MZ-80K

A terminal emulator, such as minicom, may be used to provide power and keyboard input to the Pico MZ-80K instead of a USB keyboard and 5V power supply. If you choose to run the emulator in this way, the (diagnostics) picomz-80k-diag.uf2 image **must** be used.

For example, using minicom 2.8 from a Raspberry Pi computer requires the following settings:

Ensure that the minicom window has keyboard focus before changing from HDMI input on the Raspberry Pi to VGA input from the Pico, otherwise the emulator will not see the keystrokes sent from minicom.

Information will be echoed back to the minicom window when a SHOW (printf) statement is executed by the emulator. For example:



MicroSD card support

The following microSD cards and formats are known to work in the emulator.

microSD card make / type	**microSD card format and partition sizes**
Transcend 16GB microSDHC, Class 10, UHS 1	FAT32, partition sizes up to and including the whole card
Kingston 32GB microSDHC, Class 10, UHS 1	FAT32, 2GB partition size

The following microSD cards and formats are known **not** to work in the emulator.

microSD card make / type	**microSD card format and partition sizes**
SanDisk Ultra 32GB microSDHC Class 10, A1, UHS 1	FAT32, all partition sizes

Troubleshooting

Symptom	Likely cause	Remedy
Fast flashing green led (200ms) on the Pico; no output seen on the VGA display.	A USB keyboard (or terminal emulator) is not active.	Check connections and try again by pressing the RUN button on the VGA base.
Slow flashing green led (1s) on the Pico; no output seen on the VGA display.	There is no microSD card present, or the microSD card cannot be read.	Review the manual section that discusses SD card support.
‘tapes’ fail to load or save correctly.	The cassette tape deck emulation is out of synchronisation.	Press the ‘BREAK’ key (pgDn) or the shifted ‘BREAK’ key (PgUp) and try again. If this fails, restart the emulator by pressing the RUN button on the VGA base.
‘tapes’ fail to save correctly and the cassette tape deck emulation is not out of synchronisation.	The microSD card (or partition being used on the card) is full.	Remove the microSD card from the emulator and delete unwanted files. Reinsert the card and try again.
Keypresses are not recognised by the emulator in some programs.	Some machine code games read data from the keyboard without using the monitor subroutines provided for this purpose.	Use the F9 key to try again using keyboard mode 2 or 3. The mode that the keyboard is in is reported in the bottom right hand side of the emulator status area.

Systems manual

Compiling Pico MZ-80K for the Pimoroni VGA base

Pre-requisites for Raspberry Pi OS (Debian Bookworm)

CMake (version 3.13 or later) and a gcc cross compiler.

```
sudo apt install cmake
sudo apt install gcc-arm-none-eabi libnewlib-arm-none-eabi build-essential
```

The Pico MZ-80K emulator has been tested using the Raspberry Pico SDK version 1.5.1[5]. This and the pico-extras repository must be available on your computer if you wish to compile the emulator.

Assuming that you are already in the subdirectory you wish to install the Pico SDK under, issue the commands:

```
git clone https://github.com/raspberrypi/pico-sdk.git -b 1.5.1
git clone https://github.com/raspberrypi/pico-extras.git -b sdk-1.5.1
```

Then clone the current version of the Pico MZ-80K repository.

```
git clone https://github.com/psychotimmy/picomz-80k.git
```

Ensure that the Pico SDK and Pico Extras subdirectories have been exported. For example, if these libraries have been installed under /home/pi, use:

```
export PICO_SDK_PATH=/home/pi/pico-sdk
export PICO_EXTRAS_PATH=/home/pi/pico-extras
```

Then issue the commands:

```
cd pico-sdk
git submodule update --init --recursive[6]
cd ../picomz-80k
mkdir build
cd build
cmake -DPICO_BOARD=vgaboard ..
make
```

There should now be two new .uf2 files in the build directory for the emulator that can be installed on your Pico, depending on the way you are providing keystrokes. (picomz-80k.uf2 for a physical UK USB keyboard, picomz-80k-diag.uf2 for a terminal emulator running on another computer).

5 Pico MZ-80K will build with SDK 2.0.0, but runs significantly more slowly (around 8% on benchmarks conducted with BASIC SP-5025) and seems less stable than when built with SDK 1.5.1.

6 Brings in the libraries for TinyUSB, which ironically, are huge.

Pico MZ-80K software architecture

Overview

<table>
<tr><td rowspan="6">sharpmz.h

Common header file for the emulator</td><td>sharpmz.c

Main entry point for the emulator.</td></tr>
<tr><td>sharpcorp.c

The decoded SP-1002 monitor and computer graphics ROM (UK version) for the Sharp MZ-80K.</td></tr>
<tr><td>8255.c

A simplified Intel 8255 emulator, specifically for use in this emulator.</td></tr>
<tr><td>8253.c

A simplified Intel 8253 emulator, specifically for use in this emulator. The mechanism for producing sounds from the Pico’s PWM is also included in this source file.</td></tr>
<tr><td>keyboard.c + tusb_config.h

Emulates a Sharp MZ-80K keyboard on a UK USB keyboard or via a terminal emulator.</td></tr>
<tr><td>cassette.c

Emulates reading and writing Sharp MZ-80K tapes (.mzf format) using the VGA board’s microSD card.</td></tr>
<tr><td colspan="2">vgadisplay.c

Provides a monochrome VGA representation of the Sharp MZ-80K’s display, plus emulator status information in the lower 40 scanlines.</td></tr>
<tr><td colspan="2">Third party libraries

zazu80 – a z80 instruction set emulator.
Forked from https://github.com/superzazu/z80
fatfs – a file system for the Raspberry Pico microSD card.
Version 0.15 w/ patch 1 forked from http://elm-chan.org/fsw/ff/00index_e.html
sdcard – low level routines for the fatfs library.
Forked from https://github.com/elehobica/pico_fatfs with changes made to support the pinout used by the Pimoroni VGA demo base sd card.</td></tr>
<tr><td colspan="2">Raspberry Pi libraries

Pico SDK – Version 1.5.1, including TinyUSB.
Pico Extras – Version 1.5.1</td></tr>
</table>

The MZ-80K memory map

MZ-80K Addresses		Pico MZ-80K Emulator
0xF000 – 0xFFFF (61440 – 65535) **FD ROM uses first 1024 bytes of this space when installed**	FD ROM and unused addresses	Not implemented
0xE000 – 0xEFFF (57344 – 61439) **Only addresses 0xE000 – 0xE008 used**	Devices (8255, 8253, Sound)	Implemented by 8255.c and 8253.c
0xD000 – 0xDFFF (53248 - 57343)	Video RAM	Stored in mzvram[]
0x1200 – 0xCFFF (4608 – 53247)	User RAM	Stored in element 512 onwards of mzuserram[]
0x1000 – 0x11FF (4096 - 4607)	Monitor stack and workarea	Stored in elements 0 - 511 of mzuserram[]
0x0000 – 0x0FFF (0 - 4095)	Monitor ROM	Stored in mzmonitor[]

Memory dump files

The Pico MZ-80K, through the F12 function key, supports storing the state of monitor stack and workarea, user RAM, video RAM and the z80 at the point in time when the key is pressed. This state can be restored later by using the F11 function key.

This file is based on the .mzf format with extensions and omissions. At the time of writing, this format is still evolving, so should not be used as a permanent storage mechanism for your Pico MZ-80K programs.

Pico MZ-80K memory dump file format as of October 2024.

Header

The first 128 bytes of the file. The first byte stores the value 0x20, to identify that this is a Pico MZ-80K format memory dump file. The next 12 bytes are used to store a file name (in the same way that an .mzf file does). These are:

0x4d	0x92	0xb3	0xb7	0x9d	0xbd	0x20	0x9c	0xa5	0xb3	0x9e	0x0d
M	e	m	o	r	y	<sp>	d	u	m	p	<cr>

The remainder of the header block is undefined and unused.

Monitor workarea and user RAM

The next 49,152 bytes. Populated with the contents of the monitor workarea and user RAM at the time the F12 key is pressed.

Video RAM

The next 1024 bytes. Populated with the contents of the video RAM at the time the F12 key is pressed.

Z80 state

The final 56 bytes. Populated with the contents of the mzcpu global structure, used to maintain the state of the Z80 cpu.

Acknowledgements

As well as directly including the third party libraries detailed in the architectural overview of the Pico MZ-80K, some of the code was also inspired by other projects. These include:

The KM-Z80 MZ-80K emulator by Katsumi

A MZ-80 series emulator for Raspberry Pi by Nibbles Lab

VHDL implementations of Sharp MZ series computers by Philip Smart

Picoterm by RC2014

The RC2040 by Extreme Electronics

… and, of course, the people who run and take part in RetroChallenge. Much of the work completing the first version of this emulator was performed during the October 2024 event.

My own notes made during this time can be found at retrocomputing ephemera.