

MEGA65 EDITION

VIXEN-32 User's Guide

A friendly introduction to your 32-bit home computer

CONTENTS

1. Setting up your VIXEN-32
2. Getting started
3. Beginning BASIC programming
4. Advanced BASIC
5. Sprite graphics
6. Creating sound
7. Disks and files
8. The network
9. More machines inside

Appendices: A POKE & PEEK reference · B The sixteen colors · C Boot screens · D Keyboard · E Going further

Welcome

The VIXEN-32 is a home computer in the classic mould: switch it on and a few seconds later a blue screen says **Ready**, waiting for you to type. Underneath that familiar face is a modern heart: a 32-bit RISC-V processor at 100 MHz, 512 KB of fast RAM plus 16 MB of chip RAM, sixteen 64×64 hardware sprites, a copper, a blitter, a SID sound chip *and* a four-voice sample player, real floppy-style disk drives, and an Ethernet port. Bring your own ROMs and it is also a Commodore 64 and a VIC-20, on chips of their own.

The language you meet at the prompt is **BASIC** (EhBASIC 2.22), running on a real 6502 inside the core at 25 MHz, with a full-screen editor that works exactly the way the eight-bit machines taught the world: move the cursor onto any line, change it, press `RETURN`, and it is so.

This guide is written in the spirit of the computer manuals of 1982, the ones that assumed nothing and got you flying a hot-air balloon across the screen by the sprite chapter. That balloon is here too, in Chapter 5, wearing a **V** for VIXEN.

CHAPTER 1

Setting up your VIXEN-32

The hardware

VIXEN-32 runs on the **MEGA65** computer. Its FPGA becomes the VIXEN chipset, and its keyboard, SD card slot, joystick ports and Ethernet socket become the VIXEN's own.



The VIXEN-32, with MEGA65 hardware underneath.

What you need

Three things: a MEGA65 with the VIXEN-32 core installed, a display, and an SD card carrying the machine's software.

The display. The VIXEN outputs both HDMI (720×576 at 50 Hz, 4:3, with sound) and 15 kHz analog RGB at the same time, so a modern television and a proper CRT are equally at home. The picture is true PAL: 384×224 pixels, 50 frames a second. On HDMI it can fill the height of a widescreen set: in the file browser (the `HELP` key) `F1` switches this zoom on and off and `F3` the scanlines, and the machine remembers both in `/VIXEN-32/CONFIG.TXT`.

The SD card. Everything the machine *is* lives in one file: `/VIXEN-32/KERNEL.BIN` (cores up to v0.93: `BOOT.BIN`). At power-on the core reads it from the card and becomes the machine.

Updating your VIXEN is a file copy: no flashing, no tools (Chapter 8 shows how to do it over the network without even removing the card).

The card can sit in either slot: the micro SD at the back, or the full-size slot inside the machine (core v0.84+). With a card in both, the back one is used.

Switching on

Turn the power on. The border flashes through a sequence of colors as the machine boots (on a healthy start they pass in under a second), and then:



The boot screen: the V32 logo at the left, the banner beside it, white on blue inside a yellow border. The white block is the cursor: BASIC, listening.

If instead you get a **red border** and an error code, the machine could not load `KERNEL.BIN`. Nothing is broken, and nothing on the card has been changed. Appendix C lists every code and what to do about it.

CHAPTER 2

Getting started

The keyboard

Every key types what is printed on it. Letters arrive in uppercase, which is what BASIC keywords require. A few keys are special:

KEY	WHAT IT DOES
`RETURN`	Feeds the line under the cursor to BASIC.
Cursor keys	Move anywhere on the screen; they wrap at the edges.
`INS/DEL`	Delete pulls the line left; insert pushes it right.
`HOME` / `CLR`	Cursor to the top corner; shifted, clears the screen.
`RUN/STOP`	Stops a running program: `BREAK IN LINE n`. `CONT` resumes.
`RUN/STOP` + `RESTORE`	The panic button. Clears the screen, restores the boot colors, returns to **Ready**, even from machine code.
`CTRL`	Held during `LIST`, slows the listing.
`NO SCROLL`	Pauses a listing; again to continue.
`SHIFT` + `£`	Switches the network on and off (Chapter 8).
`ESC` then `X`	40 or 80 columns, as on a MEGA65. Going back to 40 clears the screen.
`MEGA` +key	Types the character printed on the front of the key: `MEGA` + `/` is \, `,` is ~, `.` is |, `=` is _, `:` is {, `;` is }, `←` is the backquote.
`HELP`	The file browser (Chapter 7).

The screen is the editor

The screen is 40 columns by 25 rows, and all of it is editable. Type anywhere. Move the cursor up onto an old line, change a character, press `RETURN`, and the changed line is entered as if you had typed it fresh. A logical line may be up to 80 characters (two screen rows), and `RETURN` enters the whole thing. This one idea, *edit the screen, not a buffer*, is how you will fix and re-enter program lines all day.

Your first words

```
PRINT "HELLO, I AM THE VIXEN-32"
HELLO, I AM THE VIXEN-32

PRINT 2+2
 4

PRINT 512*1024
 524288
```

`PRINT` shows things; with quotes it shows text, without them it does the arithmetic first. The operators are `+` `-` `*` `/` and `^` for powers, with parentheses working as they did in school. The machine is also a fine calculator.

Variables remember values: `A=10`, then `PRINT A*3` gives 30. A name ending in `$` holds text: `N$="FOX"`. To clear the screen, print the clear-screen character: `PRINT CHR$(147)`.

Color, first taste

The border and background each have a color register you can `POKE` (Chapter 4 explains POKE properly; for now, trust it):

```
POKE 45088,2   : REM RED BORDER
POKE 45089,0   : REM BLACK BACKGROUND
```

Sixteen colors, numbered 0-15. The full chart is Appendix B. Change them, break things, then remember `RUN/STOP` + `RESTORE` puts everything back.

CHAPTER 3

Beginning BASIC programming

A program is lines with numbers. Numbered lines are stored, not executed; `RUN` executes them in order, `LIST` shows them, `NEW` throws them away. Number in tens, so there is room to insert lines later.

Loops and decisions

```
10 FOR I=1 TO 5
20 PRINT "VIXEN ";I
30 NEXT I
RUN
VIXEN  1
VIXEN  2
VIXEN  3
VIXEN  4
VIXEN  5
```

`FOR…NEXT` repeats; `STEP` changes the stride (`FOR I=10 TO 0 STEP -1` counts down). `GOTO` jumps to a line. `IF…THEN` decides. `GOSUB…RETURN` calls a subroutine and comes back. `INPUT` asks the person at the keyboard. Those six ideas are most of BASIC; everything else is vocabulary.

A whole game, twenty lines shy

```
10 REM GUESS THE NUMBER
20 N=INT(RND(1)*100)+1
30 T=0
40 INPUT "YOUR GUESS";G
50 T=T+1
60 IF G<N THEN PRINT "TOO LOW" : GOTO 40
70 IF G>N THEN PRINT "TOO HIGH" : GOTO 40
80 PRINT "GOT IT IN";T;"TRIES"
```

`RND(1)` gives a random fraction between 0 and 1; line 20 turns it into a whole number from 1 to 100. Type it in, `RUN` it, then change it: make the range bigger, count down the guesses, insult the player. Changing a working program is how programming is learned.

When a program is worth keeping, `SAVE "GUESS"` writes it to disk and `LOAD "GUESS"` brings it back. Chapter 7 has the whole story.

CHAPTER 4

Advanced BASIC

PEEK and POKE: the machine's control panel

`POKE address,value` writes a byte straight into the machine; `PEEK(address)` reads one. Most of the VIXEN's hardware (colors, sprites, sound, joysticks, the copper) is a row of such addresses starting at 45056 (`$B000`), laid out *exactly like a Commodore 64's* `$D000` block: the border is at `$B020` as it was at `$D020`, so a C64 routine ports by changing one digit. Appendix A is the full map.

READ and DATA

```
10 FOR I=1 TO 4
20 READ A : PRINT A*A
30 NEXT I
40 DATA 1,2,3,4
```

`DATA` lines hold numbers (or strings) inside the program; `READ` consumes them in order and `RESTORE` rewinds. This is how sprite shapes travel inside a listing, as Chapter 5 is about to show. `DIM A(10)` makes arrays when single variables run out.

Reading the joystick

`PEEK(45152)` is joystick port 1, `PEEK(45153)` port 2. Each direction is a bit, and the bits are *not* in a tidy order, so test them with `AND`:

BIT VALUE	1	2	4	8	32
Means	down	up	fire	right	left

```
10 J=PEEK(45152)
20 IF J AND 4 THEN PRINT "FIRE!"
30 IF J AND 32 THEN PRINT "LEFT"
40 IF J AND 8 THEN PRINT "RIGHT"
50 GOTO 10
```

A mouse in the port reads through `PEEK(45154)` / `PEEK(45155)` (X and Y) and `PEEK(45156)` (buttons). The counters run freely and wrap at 256, so keep the previous reading and subtract to get the movement since last time.

Waiting for a key

`GET A$` takes one keypress without stopping the program (empty string if none is waiting). That is the difference between a form and a game. While a program runs, the keys are its own, as on a C64: nothing is typed on the screen, and `RETURN` and the cursor keys come to `GET` as their C64 codes.

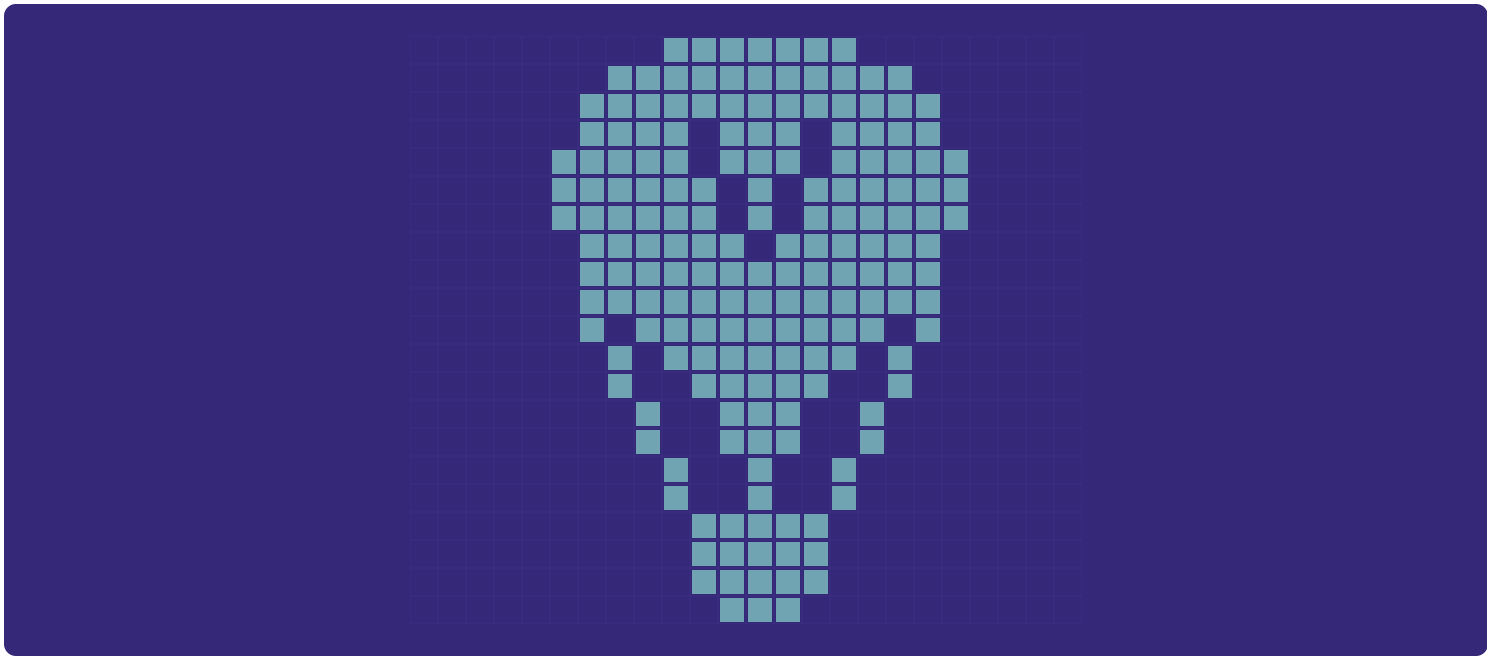
CHAPTER 5

Sprite graphics

A sprite is a picture the hardware moves over the screen for free: no redrawing, no flicker, no arithmetic. The VIXEN-32 has **sixteen** of them, each up to **64×64 pixels**, each in a single color (1 bit per pixel) or full 256-color (8 bits per pixel). All sixteen can share a scanline; there is no per-line limit to outwit.

Designing a sprite

A one-color sprite is a grid of on/off pixels, eight pixels to a byte, the leftmost pixel in the highest bit, the same order the Commodore 64 used, so classic sprite `DATA` pokes straight in. Here is ours: the hot-air balloon, 24×21, with a **V** punched out of the canopy where the old machines wore a C:



The balloon, 24×21 pixels, three bytes per row. Each row of the grid becomes three numbers in a `DATA` line.

Getting the picture into the machine

A C64 poked sprite shapes into its own RAM. The VIXEN keeps sprite images in *chip RAM*, in numbered 4096-byte **slots**, reached through a little three-register window:

POKE	HEX	MEANING
`45184,n`	`$B080`	Select slot *n* (resets the write position)

`45185 / 45186`	`$B081/2`	Write position within the slot, low / high
`45187,v`	`$B083`	Write one byte; the position steps forward by itself

A 1-bit-per-pixel image is 8 bytes per row, 64 rows: 512 bytes. Our balloon is 24 pixels wide, so each row is its three data bytes and five blank ones, and after 21 rows the rest of the slot is padded with zeros.

Aiming a sprite

Select a sprite channel, then set what it shows and how:

POKE	HEX	MEANING
`45104,n`	`$B030`	Select sprite 0-15
`45105 / 45106`	`$B031/2`	X position, low byte / high bit
`45107,y`	`$B033`	Y position
`45108,s`	`$B034`	Image slot number
`45109,c`	`$B035`	Color (1-bpp sprites)
`45110,f`	`$B036`	Flags: 2=8bpp, 4=opaque, 8=flip-H, 16=flip-V, 32=on top
`45077,m`	`$B015`	Enable sprites 0-7 (one bit each, like `$D015`)
`45112,m`	`$B038`	Enable sprites 8-15

Coordinates count from the top-left of the *border*, C64-style: the visible 320×200 window runs from X=32 to 351 and Y=12 to 211. X can exceed 255, which is why it has a high bit. The listing below shows the classic idiom for splitting it.

The familiar C64 register layout is also mirrored for the first eight sprites (`POKE 45056+2*n` for X, `45057+2*n` for Y, `45095+n` for color), so old fingers find everything where it was.

Up, up and away

Everything above, in one program. Type it in exactly, then `RUN`:

```
10 REM UP, UP AND AWAY -- THE VIXEN BALLOON
20 PRINT CHR$(147)
30 POKE 45088,0 : POKE 45089,6
40 PRINT "UP, UP AND AWAY!"
100 REM LOAD THE IMAGE INTO SLOT 1
110 POKE 45184,1
120 FOR R=0 TO 20
130 FOR B=0 TO 2 : READ Q : POKE 45187,Q : NEXT B
140 FOR B=3 TO 7 : POKE 45187,0 : NEXT B
150 NEXT R
160 FOR N=1 TO 344 : POKE 45187,0 : NEXT N
200 REM SPRITE 0: SLOT 1, CYAN, ON TOP
210 POKE 45104,0 : POKE 45108,1 : POKE 45109,3 : POKE 45110,32
220 POKE 45077,1
300 REM FLY, RISING TO THE RIGHT
310 FOR X=32 TO 288
320 Y=150-X/2
330 POKE 45105,X AND 255 : POKE 45106,-(X>255) : POKE 45107,Y
340 NEXT X
350 GOTO 310
1000 DATA 0,127,0,1,255,192,3,255,224,3,221,224
1010 DATA 7,221,240,7,235,240,7,235,240,3,247,224
1020 DATA 3,255,224,3,255,224,2,255,160,1,127,64
1030 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
1040 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

What each part does:

Lines 110-160 pour the picture into slot 1: three `DATA` bytes per row, five blanks to fill the 8-byte row, then zeros to the end of the 64-row image. **Line 210** points sprite 0 at slot 1, colors it cyan, and sets the on-top flag. **Line 220** switches it on. **Lines 310-340** fly it: `X AND 255` is the low byte and `-(X>255)` is a neat BASIC trick: the comparison is −1 when true, and the minus sign turns it into the 1 the high-bit register wants.

Collisions, and further

`PEEK(45086)` tells you which sprites touched each other this frame, `PEEK(45087)` which touched the background. It is pixel-perfect, one bit per sprite, just like `$D01E/$D01F`. When you outgrow one balloon: the machine's own demonstration, `test_hw.bas`, flies this same balloon *twice* (the second copy scaled up 3× in BASIC into a second slot) and is worth reading line by line.

CHAPTER 6

Creating sound

The VIXEN's classic voice is a **SID**, the famous three-voice synthesizer chip, cycle-modelled, with its filters and its character intact. Its 29 registers live at `POKE 46080+r` (`$B400`, the C64's `$D400` block relocated whole). A value stays set until you change it.

The beeper: one POKE makes sound

Before the SID, meet the machine's simplest instrument: a single square wave, the same thing a PC speaker is. Two registers take the pitch *in hertz*, low byte and high byte; either write sets the tone, and 0 Hz is silence:

```
POKE 45230,1 : POKE 45229,184   : REM 440 HZ, CONCERT A
POKE 45229,0 : POKE 45230,0     : REM SILENCE
```

440 is 1×256 + 184, hence the two POKEs; a pitch under 256 Hz needs only the low byte once the high byte is cleared. There is no envelope (the tone starts and stops instantly), which makes the beeper the right tool for key clicks, alarms, and this:

```
10 REM SIREN -- RUN/STOP ENDS IT
20 FOR F=300 TO 900 STEP 5
30 POKE 45229,F AND 255 : POKE 45230,INT(F/256)
40 NEXT F
50 FOR F=900 TO 300 STEP -5
60 POKE 45229,F AND 255 : POKE 45230,INT(F/256)
70 NEXT F
80 GOTO 20
```

(After breaking out, `POKE 45229,0 : POKE 45230,0` by hand, or press `RUN/STOP`+`RESTORE`, which silences everything.) A `PEEK` of either register reads back the pitch you asked for.

First SID sound

```
10 POKE 46104,15        : REM VOLUME UP
20 POKE 46080,137 : POKE 46081,65   : REM VOICE 1 PITCH
30 POKE 46085,0   : POKE 46086,240  : REM ENVELOPE
40 POKE 46084,17        : REM TRIANGLE WAVE, GATE ON
50 FOR T=1 TO 500 : NEXT
60 POKE 46084,16        : REM GATE OFF
```

The shape of every SID sound is here: set the *volume* (register 24), a *pitch* (registers 0 and 1, low and high), an *envelope* (how the note attacks, decays, sustains and releases: registers 5 and 6), then open the *gate* with a waveform (register 4: 17 triangle, 33 sawtooth, 65 pulse, 129 noise; subtract 1 to close the gate and let the note release).

Voices 2 and 3 are the same seven registers at offsets +7 and +14. Try noise (129) with a fast envelope for drums, or two voices a fifth apart. And when the SID feels tame, the machine also carries a four-voice digital sample player that games and demos use for real recorded sound.

CHAPTER 7

Disks and files

Drives A: and B:

The VIXEN mounts disk images as **drive A:** and **drive B:** (pick them in the file browser, below). BASIC speaks fluent disk:

```
SAVE "TEST"        : REM TO DRIVE A:
LOAD "TEST"
LOAD "B:GAME"      : REM FROM DRIVE B:

DIR                : REM LIST THE DIRECTORY
CD DATA            : REM ENTER A DIRECTORY, CD \ GOES BACK
DEL OLDFILE
COPY GAME B:       : REM COPY BETWEEN DRIVES
MOVE OLD NEW       : REM RENAME
MKDIR SAVES
```

Command arguments are separated by spaces, DOS-style. Long filenames, mixed case and subdirectories all work. Programs are stored in the classic `.PRG` form (two bytes of load address, then the program), so files move freely between BASIC and the machine's file menu.

The file browser

The file browser (the `HELP` key, at **Ready**) shows the card's folders and loads three kinds of file: a `.BAS` types itself into the editor (you `RUN` it), a `.PRG` loads as if typed, and a `.RVK` is a native machine-code application (a game, a demo, a terminal) that waits for `RUN`, takes over the whole machine and hands it back when it exits. Move with the cursor keys or the mouse. A `.BAS` or `.PRG` makes `C:` and its own folder current, so it finds the files that sit beside it.



The file browser in the `img` folder, with `Basic-A.img` in drive A:. Its keys are on the top and bottom lines.

KEY	WHAT IT DOES
`RETURN`	Loads the file, opens the folder, or puts an `.IMG` in drive A:.
`INS/DEL`	Up a folder.
`CTRL` + `A` / `B`	Puts the `.IMG` under the bar in drive A: / B:.
`CTRL` + `SHIFT` + `A` / `B`	Ejects drive A: / B:.
`F1` / `F3`	HDMI zoom / HDMI scanlines, on or off.
`F5`	The commander.
`HELP` or `RUN/STOP`	Back to BASIC, untouched.

The commander

`F5` in the browser opens a two-panel file manager in 80 columns, a drive in each panel. C: is the SD card, where it works only inside `/VIXEN-32/`.



The commander: `img` on the left, `/VIXEN-32/` on the right. Its keys are along the bottom.

KEY	WHAT IT DOES
`F1`	The next drive: A:, B:, C:.
`F2`	Renames the entry at the cursor.
`SPACE`	Marks an entry; `CTRL`+`A` marks them all.
`F3`	Views a file as text or hex.
`F5` / `F6`	Copies / moves to the other panel.
`F7` / `F8`	Makes a folder / deletes.
`TAB`	The other panel.
`F10` or `HELP`	Back to the browser.
`RUN/STOP`	Back to BASIC.

Files as data: READF and WRITEF

```
READF  0,"SCORES",49152     : REM FILE -> MEMORY
WRITEF 0,"SCORES",49152,100 : REM MEMORY -> FILE
```

Unit 0 is BASIC's own memory, unit 1 is the 16 MB chip RAM (sprite sheets, level maps, samples), and unit −1 asks a file's size without reading it. The result lands where a program can see it: `PEEK(45247)` is the status (0 means success) and 45244-45246 hold the byte count. **Always check the status:** a missing file does not stop the program, it just says so there.

Bootable disks

A disk whose first sector carries the `V32B` signature *boots* when mounted in A: at the **Ready** prompt. There is no key to hold; the disk decides. It stays an ordinary disk otherwise: it lists, copies, and holds files as usual. Only drive A: boots, so any disk can be examined safely from B:.

CHAPTER 8

The network

The VIXEN-32 has real Ethernet, and boots with it **off**. A retro machine should not join a network just because someone switched it on. When you want it:

```
<SHIFT + £>
NET                : REM LINK, ADDRESS, COUNTERS
PING 192.168.5.1   : REM ROUND TRIP IN MICROSECONDS
PING GOOGLE.COM    : REM BY NAME, VIA DNS
```

Out of the box the machine uses DHCP. A plain text file on the SD card, `/VIXEN-32/NETWORK.TXT`, sets a fixed address, gateway, DNS or MAC when you want one. Edit it on any PC. While the machine sits at **Ready** it answers pings, even while a BASIC program runs.

TFTP: files over the wire

```
TFTP ON
```

...and the machine becomes a file server. (`MEGA`+`£` does both at once: network on, server on.) From a PC on the same network:

```
C:\> tftp -i 192.168.5.69 put game.rvk
C:\> tftp -i 192.168.5.69 get README.TXT
```

Files named plainly land on the SD card under `/VIXEN-32/`; prefix `A:` or `B:` to write to a mounted disk. About 64 KB/s in either direction, twice the speed of an Amiga floppy. This is also how the machine itself is updated: upload a new `KERNEL.BIN`, read it back to compare, power cycle, or press the reset button (core v0.80+).

Calling the board

The terminal program (`TERMINAL_BNN.RVK`, in `/VIXEN-32/network/`) dials the VIXEN-32 bulletin board, complete with the handshake sound and `CONNECT 1200`, the logon art painting in at 1200 baud the way it really felt. On the board: multi-user chat, who-is-on, message bases, a

file area you can download from straight to the SD card, and updates for the machine itself. Bring the network up first (`SHIFT` + `£`); `RUN/STOP` hangs up.

There is an IRC client beside it for the real chat networks, and **Network Pong**: press 1 to play the machine, 2 to be matched against whoever else is out there, your mouse as the paddle. First to 9.

CHAPTER 9

More machines inside

BASIC is the front door, but three more machines live on the same card. Each one starts from a word typed at **Ready**, on a line of its own, and each asks `ARE YOU SURE (Y/N)?` first, because it ends BASIC and the program in it. Unmount drives A: and B: before you start one. The reset button brings you back.

VMOS

```
VMOS
```

VMOS is a language in the manner of AMOS: BASIC with procedures and labels and no line numbers, compiled straight to RISC-V machine code. Its programs are plain `.VMS` text files in `/VIXEN-32/VMOS/`. The VMOS Beginner's Guide takes it from there.

Bring your own ROMs

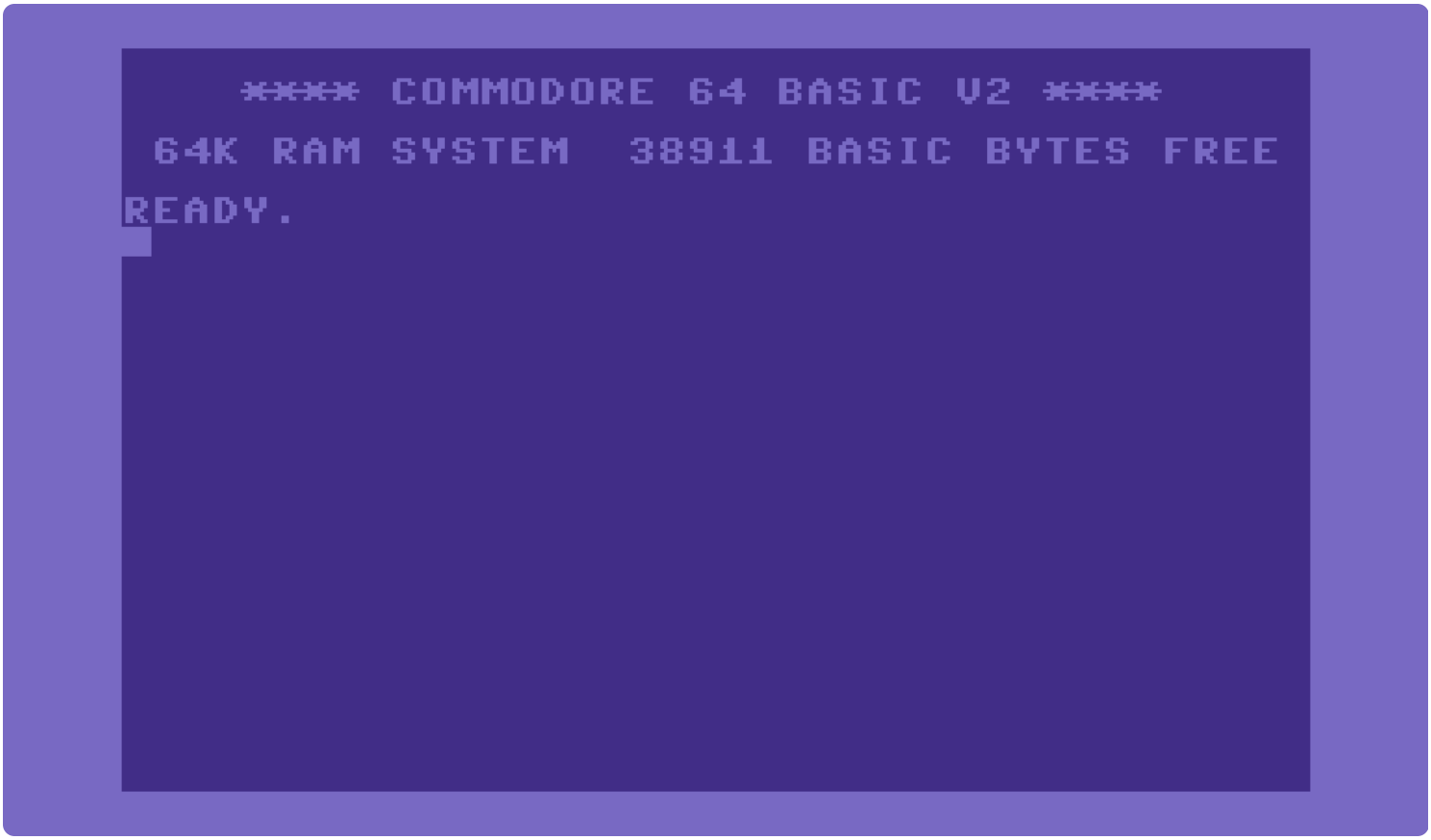
The VIXEN-32 can be a Commodore 64 and a VIC-20. These are not emulators: the 6502, the VIC-II, the VIC, the CIAs and the VIAs are chips inside the core, running at the real machines' speed. What the core does not hold is a single Commodore ROM. You put your own on the card, three files for each machine:

FILE	SIZE	IN `/VIXEN-32/C64/`	IN `/VIXEN-32/VIC20/`
`KERNAL.ROM`	8 KB	The KERNAL	The KERNAL
`BASIC.ROM`	8 KB	C64 BASIC	VIC-20 BASIC
`CHAR.ROM`	4 KB	The character ROM	The character ROM

If one is missing or the wrong size the machine says so and stays at **Ready**.

The Commodore 64

```
GO 64
```



After `GO 64` and `Y`: the C64's own start-up screen, 38911 BASIC bytes free.

This needs core v0.96 or later. The MEGA65's keyboard *is* a C64 keyboard, key for key, and both joystick ports work as they did on the real machine. Put your games in `/VIXEN-32/C64/GAMES`, in folders if you like, and press `HELP` to open the game menu. It lists `.PRG`, `.CRT`, `.D64` and `.TAP` files, and the C64 stands still while the menu is open.

IN THE MENU	WHAT IT DOES
`RETURN`	Opens a folder, or starts the game. A `.D64` disk goes into drive 8 and its first program is loaded and run. A `.TAP` tape goes in, PLAY is pressed, and LOAD and RUN are typed for you.
`SHIFT` + `RETURN`	Puts a `.D64` into drive 8 without starting it (a game's second disk), or a `.TAP` in with PLAY pressed (side 2 of a tape); on the disk or tape that is in, takes it out.
`INS/DEL`	Up a folder.
`F1`	Switches the C64 off and on: back to its BASIC.
`F3`	What the Amiga mouse in control port 1 is: a joystick in port 1, two paddles in port 1 or 2, or a 1351 mouse in port 1 or 2.
`SHIFT` + `F3`	Swaps the joystick ports: the MEGA65's port 1 is the C64's port 2, and port 2 its port 1. The menu says PORTS SWAPPED, and they stay swapped from game to game until you press it again. Paddles and the 1351 stay in their own port.

`F5` / `F7`	The mouse slower / faster as paddles or a 1351, from 12% to 150%. It starts at 50%.
`HELP`	Back to the C64 as it was.

As paddles, left and right move paddle 1, up and down paddle 2, and the two buttons are their fire buttons. A paddle or 1351 game still needs its own setting, as it did on a real C64. `ALT`+cursor keys move the picture, which brings sprites in the open top and bottom borders into sight, and `ALT`+`CLR/HOME` puts it back.

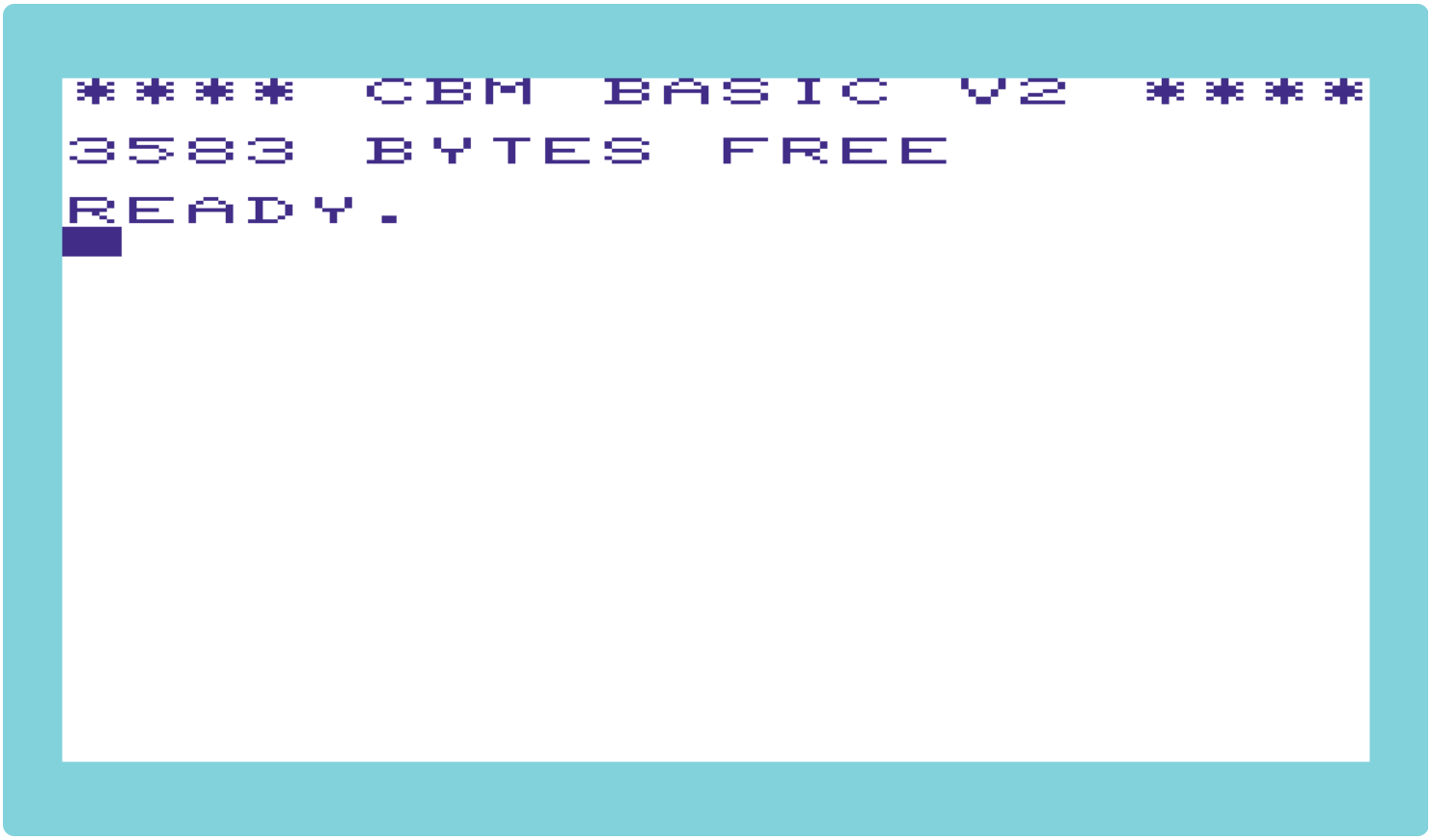
> **If the keyboard stops working.** Set to a joystick in port 1, the Amiga mouse *is* joystick 1, and on a C64 that joystick shares its lines with the keyboard. A mouse that has been moved can stop with a line held, and then keys stop working or type the wrong thing, as on a real C64. Move the mouse a little or unplug it. Or press `F3` in the menu and choose paddles or a 1351: then the mouse's movement no longer reaches those lines.

Cartridges. The menu plays these `.CRT` types: normal 8K, 16K and Ultimax, EasyFlash, Final Cartridge I and III, Ocean, Zaxxon, Simons' Basic, Fun Play, Super Games, Westermann, Rex Utility, C64 Game System, Dinamic, Magic Desk, Comal 80 and GMod2. A GMod2 game keeps its saved games on the card, in a file next to the cartridge with `.EEP` at the end of its name. Any other type, such as Action Replay, says it is not supported and is not started. There is no freeze button.

Tapes. A `.TAP` plays into the C64 as the Datasette did, pulse for pulse and at the same speed, so games with their own fast loader load too, and take as long as they did then. The tape stops while the menu is open, follows the C64's motor control, and is rewound when the C64 is switched off and on. Nothing is ever recorded onto a tape. Tapes need core v0.98 or later.

The VIC-20

```
GO V20
```



After `GO V20` and `Y`: the VIC-20 with no memory added, 3583 bytes free.

`GO 20` does the same. Put your games in `/VIXEN-32/VIC20/GAMES` and press `HELP` for the game menu. A VIC-20 game needs the right memory, so the first time you load one a box asks which blocks to fit, and the machine remembers the answer.

IN THE MENU	WHAT IT DOES
`RETURN`	Opens a folder, or loads the game.
`INS/DEL`	Up a folder.
`M`	The memory box for the game under the bar.
`F1`	Restarts the VIC-20 to BASIC; `SHIFT`+`F1` chooses its memory.
`F3`	Fit to screen, on or off.
`F5`	The Amiga mouse as two paddles, on or off (core v0.96).
`F7` / `SHIFT`+`F7`	The paddles faster / slower.
`HELP`	Back to the VIC-20 as it was.

APPENDIX A

POKE & PEEK quick reference

The I/O window starts at 45056 (`$B000`) and mirrors the C64's `$D000` layout wherever a C64 equivalent exists.

Screen and layers

ADDRESS	HEX	FUNCTION
45088	$B020	Border color (as C64 $D020)
45089	$B021	Background color (as C64 $D021)
45120 / 45121	$B040/1	Tile layer on/off; tiles in front
45122-45125	$B042-5	Tile scroll X / Y
45136	$B050	Bitmap layer on/off
45137 / 45138	$B051/2	Bitmap pointer
45139-45141	$B053-5	Bitmap scroll
45074	$B012	Raster position (read; BASIC is too slow to chase it)

Sprites

ADDRESS	HEX	FUNCTION
45056+2n / 45057+2n	$B000+	Sprite *n* X / Y (C64 layout, sprites 0-7)
45072	$B010	Sprite X high bits (as $D010)
45077 / 45112	$B015/$B038	Enable sprites 0-7 / 8-15
45095+n	$B027+	Sprite *n* color (as $D027)
45086 / 45087	$B01E/F	Collisions: sprite-sprite / sprite-background (read)
45104-45110	$B030-6	Select window: sprite no., X lo, X hi, Y, slot, color, flags
45184-45187	$B080-3	Image uploader: slot, offset lo/hi, data byte

Input

ADDRESS	HEX	FUNCTION

45152 / 45153	$B060/1	Joystick 1 / 2 (1 down, 2 up, 4 fire, 8 right, 32 left)
45154-45156	$B062-4	Mouse X, Y (wrapping counters), buttons
45280 / 45281	$B0E0/1	Last key code / still held

Sound, copper, machine

ADDRESS	HEX	FUNCTION
46080+r	$B400+	SID register r (0-28), C64 $D400 layout
45229 / 45230	$B0AD/E	Beeper pitch in Hz, low / high byte (0 = silence)
45169-45177	$B071-9	Copper: scanline, register id, value, commit entry, run n entries
45244-45247	$B0FC-F	READF/WRITEF result: byte count (3 bytes) + status
45214	$B09E	POKE 1: print the machine's licence notices

APPENDIX B

The sixteen colors

NO.	COLOR	NO.	COLOR
0	black	8	orange
1	white	9	brown
2	red	10	light red
3	cyan	11	dark grey
4	purple	12	grey
5	green	13	light green
6	blue	14	light blue
7	yellow	15	light grey

These are the boot palette: the classic set, in the classic order. The full palette holds 256 entries chosen from 65,536; native applications repaint it freely.

APPENDIX C

Boot screens and error codes

A red border with a code means `/VIXEN-32/KERNEL.BIN` could not be loaded. Nothing has been damaged; fix the card and power cycle.

CODE	MEANING	WHAT TO DO
E01	No SD card	Insert or reseat the card.
E02	No filesystem	Format the card FAT16/FAT32.
E03	KERNEL.BIN not found	Copy it to `/VIXEN-32/KERNEL.BIN`.
E04	KERNEL.BIN size wrong	The copy is truncated. Copy it again.
E05	Card read failed	Try another card.
E06	Not a boot image	The file is not a VIXEN kernel.
E07	Boot image too new	Update the core, or use a matching KERNEL.BIN.
E08	Boot image damaged	Copy it again.
E09	Boot image unusable	The image has no startup code. Use a released KERNEL.BIN.

During a normal boot the border flashes through a fixed color sequence, one color per completed step. If the machine ever *stops* on a color, that color names the last step that succeeded. That is useful to mention when asking for help.

APPENDIX D

Keyboard summary

KEY	EFFECT
`RETURN`	Enter the line under the cursor
Cursor keys	Move anywhere; wrap at the edges
`INS/DEL`	Insert / delete within the logical line
`HOME` / `SHIFT`+`CLR`	Cursor home / clear screen
`RUN/STOP`	Break a running program (`CONT` resumes)
`RUN/STOP`+`RESTORE`	Full reset to **Ready**, boot colors restored
Reset button (side of the machine)	Restart from `KERNEL.BIN`, exactly like power-on, even after a game has taken all the memory (core v0.80+)
`CTRL` / `NO SCROLL`	Slow / pause a listing
`SHIFT`+`£`	Network on / off
`MEGA`+`£`	Network and TFTP server on
`ESC` then `X`	40 / 80 columns
`MEGA`+key	The character printed on the front of the key: `\ ~ | _ { }` and the backquote
`HELP`	The file browser: load files, mount disks. In the C64 and the VIC-20: the game menu

All keys auto-repeat. Letters are sent uppercase, as BASIC keywords require.

APPENDIX E

Going further

When BASIC starts feeling small, the machine goes much deeper:

The Chipset Reference (`docs/chipset/Vixen-32_Chipset_Reference.md`) is the register-level manual (the copper, the blitter, the sample engine, interrupts, Ethernet), written for programming the machine in C.

More manuals come with the machine: the *VMOS Beginner's Guide* and the *VMOS Command Reference*, the *VIC-20 User's Manual* and the *C64 User's Manual*.

Thirteen tutorials in `risc/tuts/` take you from "hello world" to a five-layer parallax scroller, each one a single commented C file that builds a loadable `.RVK`.

Worked BASIC examples live in `risc/`: `test_hw.bas`, the full balloon demonstration, among them.

`POKE 45214,1` prints the machine's acknowledgements. VIXEN-32's BASIC is derived from EhBASIC by Lee Davison, free for non-commercial use, a condition the machine carries with pride.

VIXEN-32 User's Guide, MEGA65 edition. Written in the tradition of the great home-computer manuals of 1982: assume nothing, promise a balloon, deliver the balloon.