Pocket CP/M

The Companion Guide

CP/M 2.2 on iPhone, iPad and Android

```
A0>dir
A: DDT      COM  |  ED       COM  |  MBASIC   COM
A: PIP      COM  |  STAT     COM  |  SUBMIT   COM
A0>mbasic
BASIC-85 Rev. 5.29
39224 Bytes free
Ok
```




```
A0>
```

FREE EDITION

```
10 FOR I = 1 TO 5
20 PRINT I, I * I
30 NEXT I
RUN
 1            1
 2            4
 3            9
```

Commands · Drives & user areas · Transferring files · MBASIC



Pocket CPM

The Companion Guide

CP/M 2.2 in your pocket, explained

For Pocket CPM 1.0.0 on iPhone and iPad
and Pocket CPM 1.0.1 on Android

Pocket CPM: The Companion Guide
Copyright © 2026 Pymble Software Pty Ltd. All rights reserved.

Free edition. This guide is free. You are welcome to copy and share it, unchanged and in full, with anyone it might help.

First edition, typeset November 21, 2026.

Pocket CPM is published by Pymble Software. Its emulator core is based on RunCPM by Marcelo Dantas. iPhone, iPad, Finder, iCloud Drive and App Store are trademarks of Apple Inc. Android and Google Play are trademarks of Google LLC. CP/M is a trademark of its owners (originally Digital Research, Inc.). Microsoft BASIC, MBASIC, Zork and other product names mentioned are the property of their respective owners and are used here only to describe the app; no endorsement is implied.

Much CP/M software is decades old but may still be under copyright. You are responsible for making sure you have the right to use any software you load into the emulator.

Contents

1 Welcome to Pocket CPM **1**
- 1.1 What the app does . . . 1
- 1.2 What the app does not do . . . 1
- 1.3 Who this guide is for . . . 2
- 1.4 Versions covered . . . 2
- 1.5 How to read this guide . . . 2

2 Getting Started **4**
- 2.1 Installing the app . . . 4
- 2.2 First launch . . . 4
- 2.3 The prompt . . . 5
- 2.4 A five-minute tour . . . 6
- 2.5 Leaving and coming back . . . 7
- 2.6 The iPad . . . 7
- 2.7 Where next . . . 9

3 CP/M in Ten Minutes **10**
- 3.1 A little history . . . 10
- 3.2 What is inside the app . . . 10
- 3.3 The prompt . . . 11
- 3.4 Drives and user areas . . . 11
- 3.5 File names . . . 12
- 3.6 Programs and built-in commands . . . 12
- 3.7 Control keys . . . 13
- 3.8 How CP/M talks to the screen . . . 13

4 The Terminal and the Keyboard **14**
- 4.1 The terminal screen . . . 14
- 4.2 Zooming and panning . . . 15
- 4.3 Typing . . . 15
- 4.4 The key bar . . . 16
- 4.5 What the terminal understands . . . 17

5 The Menu and Settings **19**
- 5.1 Display effects (both platforms) . . . 19
- 5.2 The rest of the iPhone and iPad menu . . . 21
- 5.3 The rest of the Android menu . . . 21

5.4 The Android Settings screen . . . 23
5.5 The ESC log (Android) . . . 24
5.6 Terminal identity (Android) . . . 25

6 Drives, User Areas and Files 26
6.1 Drives and user areas, inside CP/M . . . 26
6.2 How drives map to folders . . . 26
6.3 Where the storage is on each platform . . . 27
6.4 The special files . . . 28
6.5 Bundled files come back . . . 29
6.6 Keeping your files safe . . . 29

7 The Command Processor 30
7.1 Getting help: ? . . . 30
7.2 The standard CP/M commands . . . 31
7.3 Extra commands in Pocket CPM . . . 31
7.4 EXIT: don't . . . 32
7.5 Running programs . . . 32
7.6 Batch files: SUBMIT and AUTOEXEC.TXT . . . 33
7.7 Error messages . . . 33

8 Transferring Files 34
8.1 Before you start: preparing files . . . 34
8.2 iPhone and iPad: using the Files app . . . 35
8.3 iPhone and iPad: from a Mac or PC . . . 36
8.4 Android: Import File . . . 36
8.5 Android: from a computer . . . 38
8.6 Checking a transfer . . . 38

9 The Software: What Comes With It and What to Add 39
9.1 What comes with the app . . . 39
9.2 PIP: copying files . . . 39
9.3 ED, STAT, DDT and SUBMIT . . . 40
9.4 Adding software . . . 40
9.5 MBASIC . . . 41

10 Recipes 44
10.1 Install MBASIC . . . 44
10.2 Write, save and list a BASIC program . . . 45
10.3 Type in a BASIC program on your phone and run it . . . 46
10.4 Get a file out of Pocket CPM . . . 47
10.5 Organise files in user areas . . . 48
10.6 Clear the screen . . . 48
10.7 Start every session with your own command . . . 48
10.8 Make another drive (iPhone and iPad) . . . 49
10.9 Check that a copied program arrived intact . . . 49

11 Troubleshooting and FAQ **51**
11.1 The screen and keyboard 51
11.2 Drives and files 52
11.3 Programs 52
11.4 Frequently asked questions 53

A iPhone and Android Compared **54**

B Quick Reference **56**
B.1 Built-in commands 56
B.2 Line-editing keys (at the prompt, in ED and PIP) 56
B.3 The key bar 57
B.4 Terminal escape sequences 57
B.5 Useful BDOS functions for programmers 58
B.6 Where things are 59

C Glossary **60**

Index **63**

CHAPTER 1

Welcome to Pocket CPM

Pocket CPM puts a complete 1970s microcomputer in your phone. Behind its green-on-black terminal is an emulated Z80 processor with 64 KB of memory, running CP/M 2.2, the operating system that ran most business computers before the IBM PC arrived. When the app starts you get the same `A0>` prompt that greeted people at their Kaypros, Osbornes and Altairs, and you can run the same programs: Microsoft BASIC, WordStar-era editors, the original Digital Research utilities, assemblers, compilers and text adventures.

You do not need to know anything about CP/M to use the app. This guide explains the handful of ideas you need, every command the built-in command processor understands, and, most importantly for real use, how to get programs and files into and out of the emulator on your iPhone, iPad or Android device.

1.1 What the app does

Pocket CPM has one main screen: a terminal. Everything you type goes to CP/M and everything CP/M prints comes back to the screen. Around that terminal the app adds:

- a **key bar** with **ESC**, a latching **CTRL** key and four arrow keys, for the keys a phone keyboard lacks (Chapter 4);
- a **menu** (the ⋮ button at the top right) with display effects such as phosphor colour, CRT glow and scan lines, and links to help and feedback (Chapter 5);
- a **folder of files** on your device that the emulator treats as its disk drives (Chapter 6);
- a way to bring files in from the rest of your device: the Files app on iPhone and iPad, and the **Import File** menu item on Android (Chapter 8).

1.2 What the app does not do

- **It is not a disk-image emulator.** Pocket CPM doesn't open `.DSK` or `.IMG` disk images. CP/M files live as ordinary files in ordinary folders, one folder per drive and user area (§6.2). If you download software as a disk image you need to extract the files first on a computer.
- **It doesn't download software for you.** The **CP/M Software** item on the iPhone menu opens a well-known archive in your browser; fetching and copying the files is up to you (Chapter 8).
- **It doesn't emulate graphics or sound.** CP/M was a text operating system, and so is the app.

- **It isn't a terminal for other computers.** There is no serial port, modem or network inside the emulator.

1.3 Who this guide is for

Two kinds of reader pick up Pocket CPM. Some used CP/M decades ago and want it back; they can skim Chapter 3 and go straight to the details of the app in Chapters 4 to 8. Others are curious about computing history, or want to learn BASIC or assembly language the way people did in 1980; Chapter 3 is a ten-minute primer for them, and Appendix C is a glossary.

1.4 Versions covered

Platform	Version	Runs on
iPhone and iPad	Pocket CPM 1.0.0 (build 1001)	iOS / iPadOS 26.2 or later
Android	Pocket CPM 1.0.1 (build 1010)	Android 9 (API level 28) or later

Both apps share the same emulator core, written in C and based on RunCPM by Marcelo Dantas, so CP/M itself behaves identically on both: the same prompt, the same commands, the same file layout. The app *around* the emulator is written separately for each platform (Swift for Apple devices, Kotlin for Android) and those parts differ, sometimes a lot. The biggest difference is how you transfer files (Chapter 8). Wherever the platforms differ the guide says so, and Appendix A collects every difference in one table.

The app calls itself **Pocket CPM**, without a slash, on its home-screen icon and title bar, so this guide uses that name for the app and *CP/M* for the operating system it runs.

1.5 How to read this guide

- **Chapter 2** gets the app running and shows you around.
- **Chapter 3** is a ten-minute primer on CP/M.
- **Chapter 4** covers the terminal, the keyboard and the key bar.
- **Chapter 5** covers the menu and the settings.
- **Chapter 6** explains drives, user areas and files, and where they live on your device.
- **Chapter 7** is a reference to every built-in command.
- **Chapter 8** explains how to transfer files in and out, step by step for each platform.
- **Chapter 9** covers the software that comes with the app and how to add more.
- **Chapter 10** is a set of recipes, and **Chapter 11** is troubleshooting.
- **Appendices** compare the platforms, give a quick reference and define terms.

Names of buttons and menu items are printed in **bold** exactly as they appear in the app. Commands you type, and what CP/M prints, are shown in `code type`. CP/M doesn't care about upper or

lower case in commands, so `dir` and `DIR` are the same; this guide uses capitals for CP/M commands, as the manuals of the time did.

> **Tip.** Every screenshot in this guide was taken from the real apps running on an iPhone 17 simulator, an iPad Air simulator and an Android phone emulator. Where two screenshots sit side by side, the iPhone is on the left and Android on the right.

CHAPTER 2

Getting Started

This chapter gets Pocket CPM installed and running and takes you on a five-minute tour: your first commands, your first program, and where to go next.

2.1 Installing the app

Pocket CPM is free on the App Store (for iPhone and iPad) and on Google Play (for Android phones and tablets). Search for **Pocket CPM** and look for the green CP/M screen icon from Pymble Software.

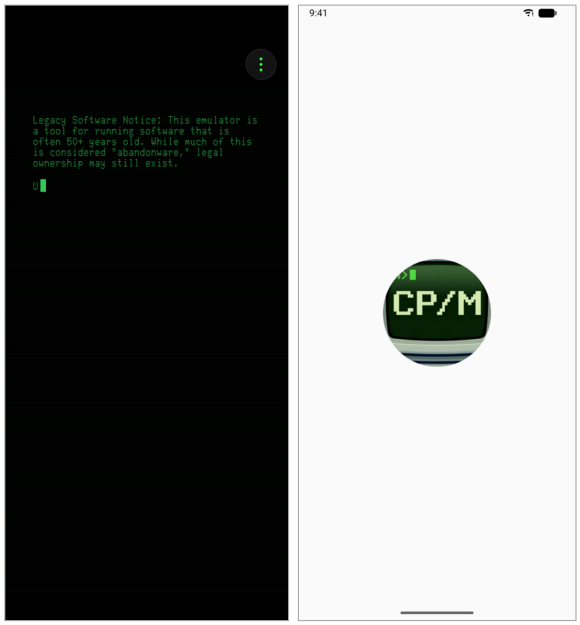
	iPhone and iPad	Android
Needs	iOS or iPadOS 26.2 or later	Android 9 or later
Permissions asked for	None	None
Internet needed	No (only for the menu's web links)	No (only for the menu's web links)

The app doesn't ask for any permissions and doesn't need a network connection: the emulator runs entirely on your device.

2.2 First launch

On iPhone and iPad, Pocket CPM opens with a *Legacy Software Notice*, typed out a character at a time on a black screen with a blinking cursor, in the style of an old terminal. It reminds you that much CP/M software, though often called "abandonware", may still be owned by someone, and that you are responsible for having the right to use what you load. The notice moves on to the terminal by itself after about twenty seconds; **tap the screen** to skip it.

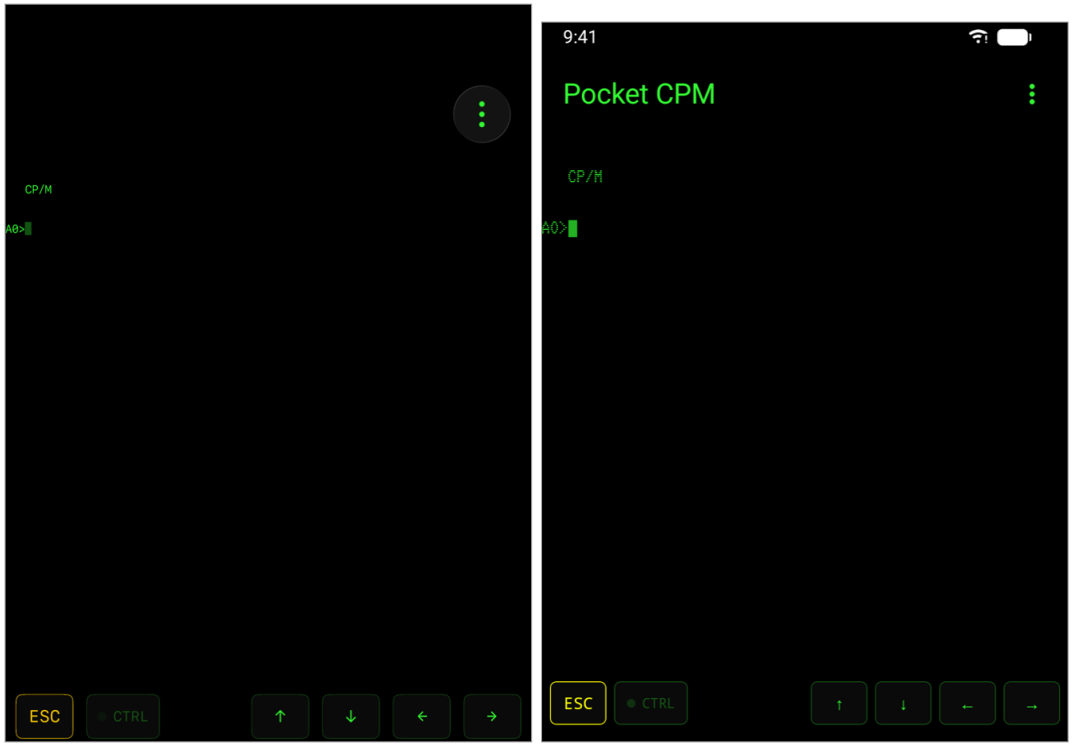
On Android the app shows its icon briefly while it starts and goes straight to the terminal. The same notice is included in the `READ.ME` file on drive A: (type `TYPE READ.ME`).



Starting up: the Legacy Software Notice on iPhone (left), and Android's start-up screen (right).

2.3 The prompt

A moment later CP/M has started. The screen shows `CP/M` and then the prompt, and the keyboard slides up:



Ready: the `A0>` prompt on iPhone (left) and Android (right).

`A0>` means CP/M is using drive A:, user area 0, and is waiting for a command (§3.3). If the keyboard isn't showing, tap anywhere on the black terminal.

2.4 A five-minute tour

Type each command and press **Return**. Upper or lower case doesn't matter.

1. See what's on the disk.

```
A0>dir
```

`DIR` lists the files on drive A:. On both platforms the app comes with the same six files: the classic Digital Research tools `DDT`, `ED`, `PIP`, `STAT` and `SUBMIT`, and a `READ.ME` (Chapter 9).

2. Read the READ.ME.

```
A0>type read.me
```

`TYPE` shows a text file. This one has the legal notice and short instructions for copying in software. When the screen fills, `TYPE` pauses; press any key for the next page or Ctrl-C to stop (to type Ctrl-C, tap **CTRL** on the key bar and then **c**).

3. Ask for help.

```
A0>?
```

The `?` command lists all the built-in commands (Chapter 7).

4. Check the disk.

```
A0>stat
```

`STAT` is a program (`STAT.COM`) rather than a built-in command. It reports how much space is free on the drive.

5. Add BASIC. Microsoft BASIC isn't included, but once you have copied `MBASIC.COM` into the app (Chapter 8 and Recipe 10.1 show how, for each platform), a first session looks like this:

```
A0>mbasic
BASIC-85 Rev. 5.29
[CP/M Version]
Copyright 1985-1986  $  by Microsoft
Created: 28-Jul-85
39224 Bytes free
Ok
10 PRINT "HELLO, WORLD"
RUN
HELLO, WORLD
Ok
SYSTEM
A0>
```

`SYSTEM` leaves BASIC and returns to CP/M. The same steps bring in any other CP/M program: Chapter 9 suggests some to try.

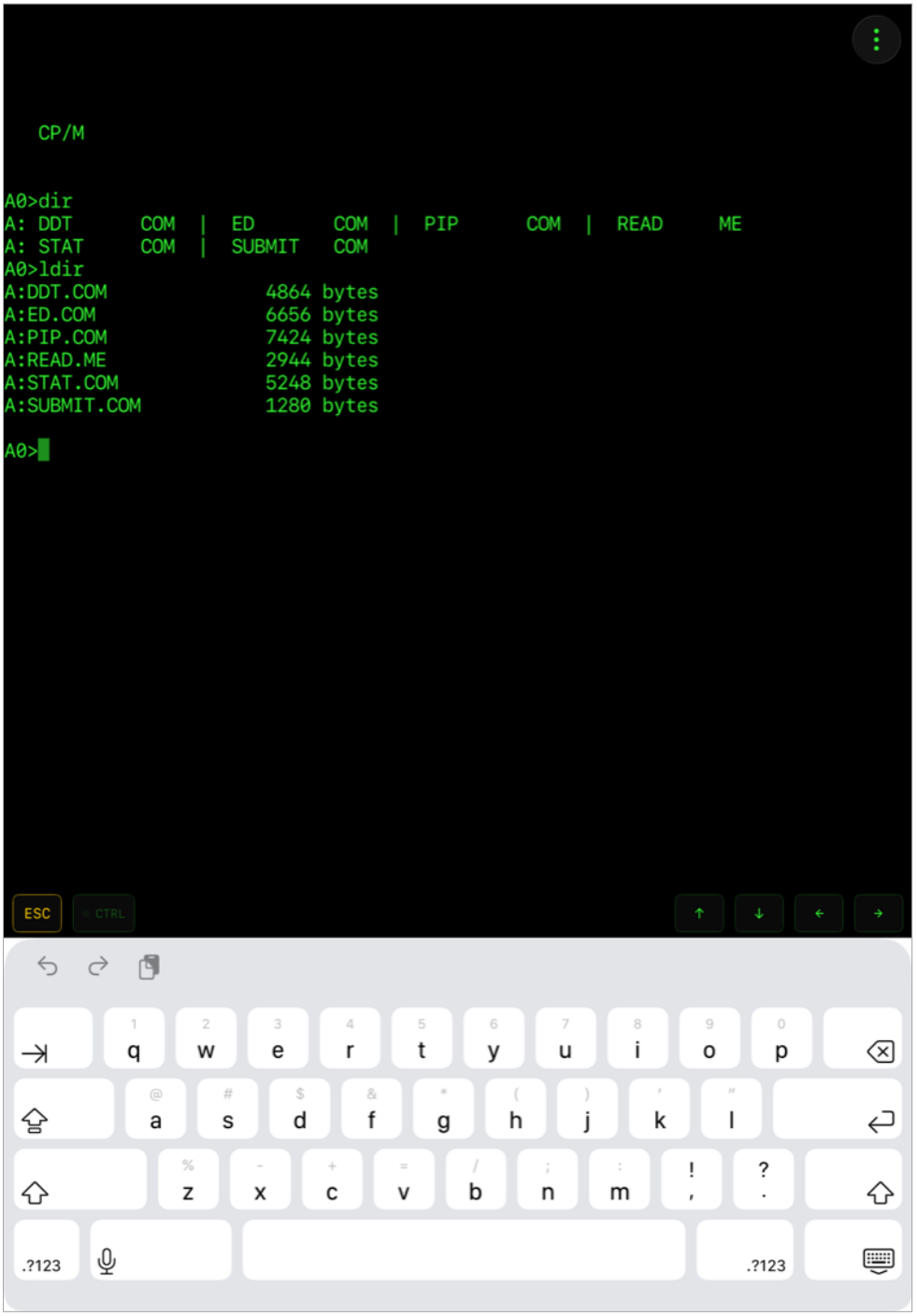
2.5 Leaving and coming back

There is nothing to save and no need to quit. When you switch away from Pocket CPM, CP/M carries on where it was; when you come back, the keyboard reappears. If your device closes the app in the background, the next launch starts CP/M afresh at the `A0>` prompt.

Files are different from the session: everything you save to disk is stored straight away in the app's storage on your device (Chapter 6) and is still there next time. Only what was on the screen, and any unsaved program in BASIC's memory, is lost when the app closes.

2.6 The iPad

On iPad the app is the same as on iPhone, with more room: the 80-column terminal is readable at full width without zooming, and with a hardware keyboard attached Pocket CPM feels like a real CP/M machine. Letters, digits, Return and Delete on a hardware keyboard reach CP/M just as the on-screen ones do; for control characters, keep using the **CTRL** key on the key bar.



Pocket CPM on an iPad Air.

2.7 Where next

- To understand what you are looking at, read **Chapter 3**, *CP/M in Ten Minutes*.
- To install software, go to **Chapter 8**, *Transferring Files*.
- To try things straight away, the recipes in **Chapter 10** need nothing but the app.

CHAPTER 3

CP/M in Ten Minutes

This chapter is for anyone who has never used CP/M. It covers the history in a page and then the half-dozen ideas you need to feel at home at the `A0>` prompt. If you used CP/M in the 1980s, skip to Chapter 4.

3.1 A little history

In 1974 Gary Kildall, then teaching at the Naval Postgraduate School in California, wrote a small operating system so that a computer built around Intel's 8080 microprocessor could use a floppy disk drive. He called it CP/M, the *Control Program for Microcomputers*. He and his wife Dorothy McEwen founded Digital Research to sell it.

CP/M's trick was portability. All the code that depended on a particular machine was kept in one small part, the BIOS, which each manufacturer wrote for its own hardware. Everything above the BIOS was the same everywhere. So a program written for CP/M on one computer would run on hundreds of others, and by the early 1980s CP/M ran on most small business computers, from the Altair and IMSAI to the Osborne 1, the Kaypro II, the Amstrad PCW and add-in cards for the Apple II. A huge library of software grew up around it: WordStar, dBASE II, SuperCalc, Microsoft BASIC, Turbo Pascal and thousands of public-domain utilities and games.

When IBM chose Microsoft's MS-DOS for its PC in 1981, CP/M's days were numbered, but MS-DOS borrowed heavily from it. If you know the old DOS prompt, drive letters, `DIR`, `TYPE`, `REN` and eight-character file names with three-character extensions, you already know a lot of CP/M.

CP/M 2.2, the version Pocket CPM emulates, appeared in 1979 and became the most widely used release.

3.2 What is inside the app

Pocket CPM emulates the parts of a CP/M computer that matter to software:

Part	On a real machine	In Pocket CPM
Processor	Intel 8080 or Zilog Z80 chip	An emulated Z80 (which also runs 8080 programs)
Memory	Up to 64 KB of RAM	64 KB, of which about 60 KB is free for programs

Part	On a real machine	In Pocket CPM
Disks	Floppy or hard disks, A: to P:	Folders on your device, one per drive and user area
Console	A video terminal or teletype	The app's on-screen terminal
Printer	A printer on the LST: device	A text file, `LST.TXT`, on drive A: user 0

The emulator core is RunCPM, an open-source CP/M emulator by Marcelo Dantas. Pocket CPM uses RunCPM's built-in command processor rather than the original Digital Research one; it understands all the standard commands plus some handy extras (Chapter 7).

3.3 The prompt

When CP/M is ready for a command it shows a *prompt*:

```
A0>
```

The letter is the current **drive** (A: to P:), the number is the current **user area** (0 to 15), and `>` means it is waiting for you. Traditional CP/M showed only `A>`; RunCPM adds the user number so that you always know where you are. While a `SUBMIT` batch job is running, the `>` changes to `$`.

Type a command and press **Return**. If CP/M doesn't recognise it, it repeats the word with a question mark:

```
A0>hello
HELLO?
```

That is CP/M's only error message for an unknown command. It isn't rude, just economical: the 1979 version had to fit in a few kilobytes.

3.4 Drives and user areas

A CP/M computer could have up to sixteen disk drives, named A: to P:. To switch to another drive, type its letter and a colon:

```
A0>b:
B0>
```

If the drive doesn't exist, CP/M says `Bdos Err on B: Select` and, after you press a key, puts you back on the drive you were on (§11.2 explains how to create drive B: and the others).

Each drive is divided into sixteen **user areas**, numbered 0 to 15. They were CP/M's way of keeping different people's (or different projects') files apart on one disk, long before folders were invented. Change user area with the `USER` command, or with a number and a colon; you can combine both, as in `B3:`:

```
A0>user 1
A1>0:
A0>b3:
B3>
```

Files in one user area are invisible from the others, with one helpful exception in Pocket CPM: if you type the name of a program that isn't in the current drive and user area, the command processor also looks for it on drive A: user 0 (§7.3). So programs kept in `A0:` can be run from anywhere.

3.5 File names

A CP/M file name has up to eight characters, a dot, and an extension of up to three characters: `MBASIC.COM`, `README.TXT`, `SQUARES.BAS`. CP/M ignores case and shows names in capitals. The extension tells you what a file is:

Extension	Meaning
`.COM`	A program you can run, by typing its name without the extension
`.BAS`	A BASIC program, run with `MBASIC NAME`
`.SUB`	A batch file for `SUBMIT`
`.ASM`, `.MAC`	Assembly-language source
`.HEX`	A program in Intel hex format, turned into a `.COM` by `LOAD`
`.TXT`, `.DOC`, `.ME`	Text
`.$$$`	A temporary file

Two *wildcards* let one command work on several files: `?` matches any single character and `*` matches the rest of the name or extension. `DIR *.COM` lists every program; `ERA TEST?.BAS` erases `TEST1.BAS`, `TEST2.BAS` and so on.

3.6 Programs and built-in commands

There are two kinds of command:

- **Built-in commands**, such as `DIR`, `TYPE`, `ERA`, `REN`, `SAVE` and `USER`, are part of the command processor and are always available. Pocket CPM adds more (`COPY`, `LDIR`, `DUMP`, `?` and others); Chapter 7 covers them all.
- **Transient programs** are `.COM` files on disk. Type the name without `.COM` and CP/M loads the file into memory at address `0100h` and runs it. When the program ends you are back at the prompt.

Anything you type after the command is passed to the program, so `MBASIC SQUARES` starts BASIC and tells it to load `SQUARES.BAS`.

3.7 Control keys

CP/M uses a few control characters (hold **Ctrl** and press a letter; on the phone, tap **CTRL** on the key bar and then the letter):

Key	What it does
Ctrl-C	On an empty input line: abandon it and return to CP/M (a "warm boot")
Ctrl-H	Backspace (the phone's delete key sends this)
Ctrl-W	On an empty line: bring back the last command you typed
Ctrl-A / Ctrl-F	Move the cursor one character left / right within the line
Ctrl-B	Jump to the start of the line, or back to the end
Ctrl-G	Delete the character under the cursor
Ctrl-K	Delete from the cursor to the end of the line
Ctrl-X	Delete everything to the left of the cursor
Ctrl-U	Delete the whole line
Ctrl-R	Retype the line
Ctrl-E	Move to a new screen line without ending the command
Ctrl-Z	End of text: ends input in `ED` and `PIP`, and marks the end of a text file

These editing keys belong to the line editor built into Pocket CPM's emulator, which is richer than the one in the original CP/M 2.2 (the cursor movement and Ctrl-W recall came later, with CP/M 3). They work wherever a program reads a whole line through CP/M, which includes the command prompt, `ED` and `PIP`. Programs that read keys one at a time, such as MBASIC and full-screen editors, have their own rules; Ctrl-C, for example, stops a running BASIC program.

3.8 How CP/M talks to the screen

CP/M itself knows nothing about screens: it sends characters to a terminal and the terminal decides what they mean. Ordinary characters are printed; a few control characters move the cursor (carriage return, line feed, backspace); and *escape sequences*, an ESC character followed by a few more, ask the terminal to clear the screen, move the cursor or turn on reverse video. Different terminals understood different sequences, which is why so much CP/M software came with an installation program to choose your terminal.

Pocket CPM's screen understands the DEC VT100 (ANSI) sequences, the most widely supported set. Chapter 4 describes what it supports and Appendix B lists the sequences.

CHAPTER 4

The Terminal and the Keyboard

The terminal is the whole of Pocket CPM's main screen. It plays the part of the video terminal that sat on the desk beside a CP/M computer: a screen of green characters on black, and a keyboard. This chapter covers how to read it, how to type into it, and the key bar that supplies the keys a phone keyboard leaves out.

4.1 The terminal screen



The terminal after `DIR`*, on iPhone (left) and Android (right). The key bar sits between the screen and the keyboard.*

From top to bottom the screen has:

- **A title bar.** On Android it shows **Pocket CPM** and the ⋮ menu button. On iPhone the title is blank, the system status bar is hidden to make room, and the menu button is a round ⋮ button at the top right.
- **The terminal itself**: 80 columns of text, as on a real terminal. The iPhone screen has 24 lines, like a VT100; the Android screen has 25.
- **The key bar**: **ESC**, **CTRL** and four arrow keys (§4.4).
- **The keyboard**, which appears when the terminal starts and whenever you tap the screen.

There is no scroll-back. As on a real terminal, when text reaches the bottom the screen scrolls up and the top line is gone for good. If you need to read a long file, use `TYPE` (which pauses every page, §7.2) rather than scrolling.

The two platforms draw characters differently. The iPhone uses the system's monospaced font. Android uses *Glass TTY VT220*, a font modelled on the characters of a DEC VT220 terminal, which gives it a more authentic look (the same font draws the startup notice on iPhone, §2.2).

4.2 Zooming and panning

Eighty columns is a lot for a phone held upright, so the app fits the full width of the terminal to the screen when it starts, which makes the text small. To read it more easily:

- **Pinch outwards** to zoom in, up to four times; **pinch inwards** to zoom out, down to half size.
- **Drag** with one finger to move around the zoomed screen.

Turning the phone to landscape gives the terminal more width. Both apps allow landscape on phones and tablets.

> **Tip.** On iPad (Chapter 2) the full 80 columns are comfortably readable without zooming, and a hardware keyboard makes Pocket CPM feel like a real CP/M machine.

4.3 Typing

Tap anywhere on the terminal to bring up the keyboard. Everything you type goes straight to CP/M: there is no input box and no **Send** button. **Return** sends a carriage return (which ends a command), and the **delete** key sends a backspace (Ctrl-H), which CP/M treats as "rub out the last character".

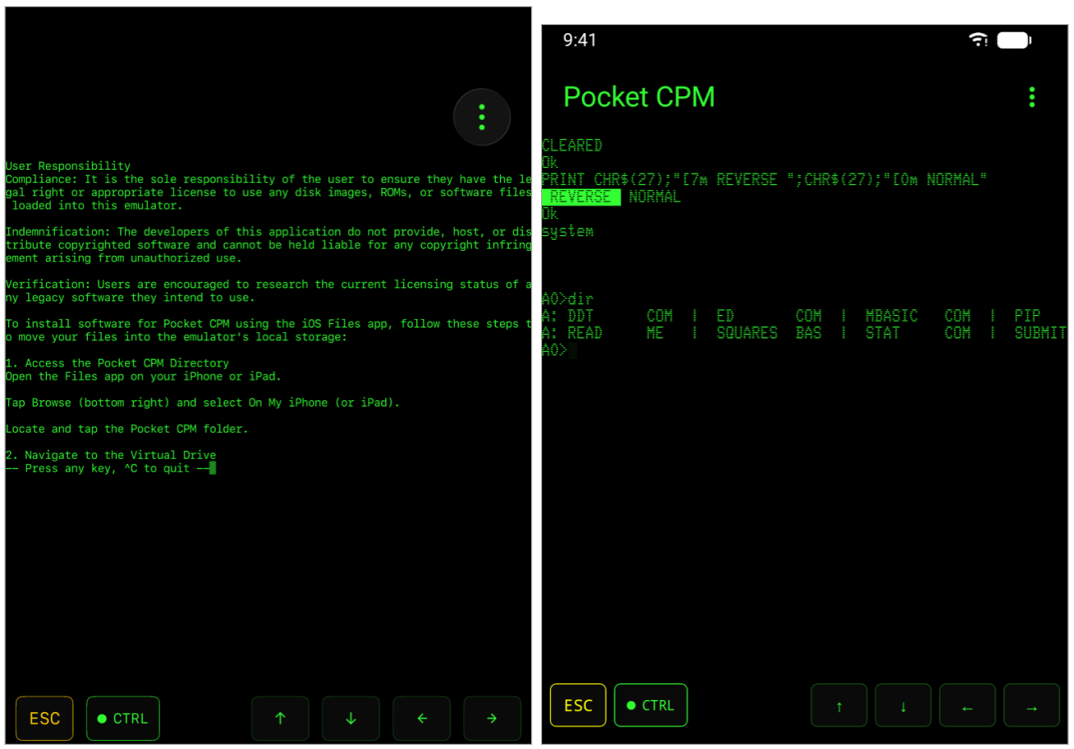
The apps switch off the keyboard features that get in the way of a command line. On iPhone, auto-correction, automatic capitals, smart quotes and smart dashes are all off, and the keyboard is the plain ASCII one. On Android, the app asks the keyboard for plain text without suggestions; if your keyboard still capitalises or "corrects" what you type, turn those features off in its settings.

Two more differences:

- **On Android the system Back button doesn't hide the keyboard** while the terminal is showing; the app keeps the keyboard up so you don't lose it by accident. The keyboard also reappears every time you return to the app.
- **On iPhone, a hardware keyboard works too** (an iPad keyboard, or a Bluetooth keyboard), sending the same characters.

CP/M itself doesn't care about upper or lower case in commands and file names; it converts them to capitals. Programs are a different matter: MBASIC, for instance, accepts keywords in either case but keeps string contents as typed.

4.4 The key bar



*The key bar with **CTRL** armed, on iPhone (left) and Android (right). The dot on the CTRL key lights up.*

Key	Sends	Use it for
ESC (yellow)	The escape character, `1Bh`	Leaving modes in editors and games; the first key of many WordStar-style commands

Key	Sends	Use it for
CTRL	Nothing by itself	Turns the next key into a control key (below)
↑ ↓ ← →	The VT100 cursor-key sequences `ESC [ A`, `ESC [ B`, `ESC [ D`, `ESC [ C`	Full-screen programs that understand VT100 cursor keys

CTRL is a latch. Tap it and its dot lights up; tap a letter on the keyboard and the app sends the matching control character (Ctrl-C, Ctrl-Z and so on), then releases CTRL by itself. Tap CTRL again before typing a letter to cancel it. There is a small difference between the platforms here:

- On **iPhone**, CTRL works with any key: the app keeps the lowest five bits of the character, as a real terminal did, so `CTRL` then `[` sends ESC, and `CTRL` then `@` sends a null.
- On **Android**, CTRL works with the letters A to Z. With any other key, the key is sent unchanged and CTRL is released.

The arrow keys are for full-screen programs. They send the VT100 sequences that programs configured for a VT100 or ANSI terminal expect. CP/M's own command line doesn't understand them: at the `A0>` prompt, the ESC is ignored and the rest of the sequence (`[A`, `[B` …) appears as ordinary characters. To move along a command line, use the CP/M editing keys instead: Ctrl-A and Ctrl-F to move, Ctrl-W to recall the last command (§3.7).

4.5 What the terminal understands

Programs control the screen by sending control characters and escape sequences (§3.8). Pocket CPM's terminal behaves like a DEC VT100, the most common choice in CP/M software's installation menus. It understands:

- carriage return, line feed, backspace and form feed (Ctrl-L, which clears the screen); Android also expands tabs to the next multiple of eight columns, while the iPhone ignores the tab character;
- cursor movement: `ESC [ row ; col H`, and up, down, left and right by a number of places;
- erasing: `ESC [ J` and `ESC [ K` with their variations, to clear the screen or a line;
- character attributes with `ESC [ … m`: bold, dim, underline, blink, reverse video and invisible;
- saving and restoring the cursor position (`ESC 7` and `ESC 8`), scrolling (`ESC D`, `ESC M`), and a full reset (`ESC c`);
- reverse-screen mode (`ESC [ ? 5 h`); the 132-column switch (`ESC [ ? 3 h`) gives a real 132-column screen on Android, while on iPhone it only clears the screen and stays at 80 columns;
- the "identify yourself" request, `ESC [ c`.

The Android terminal also answers cursor-position requests (`ESC [ 6 n`), which some full-screen programs use to measure the screen, and supports a few more cursor-movement sequences. Appendix B lists every sequence and which platform supports it.

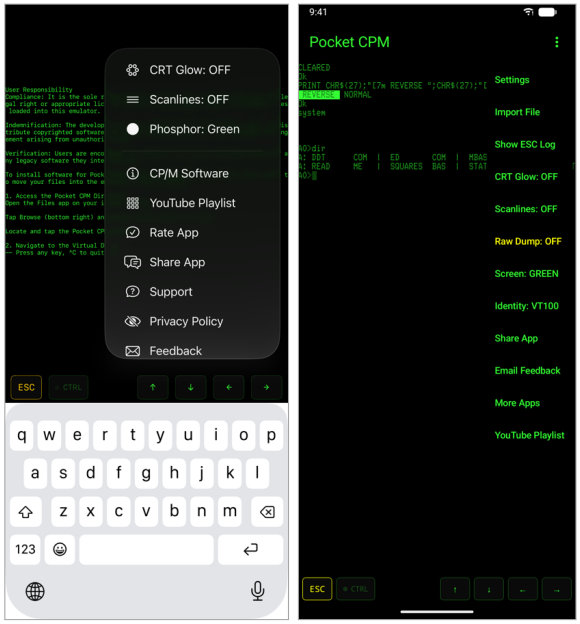
Colours are not supported: CP/M terminals were monochrome, and colour codes are ignored. Neither is the ADM-3A terminal's own command set, nor the VT52's, so when a program asks which terminal you have, choose **VT100** or **ANSI**.

> **Tip.** A program that prints strange `[2J` or `=` sequences on the screen was installed for a different terminal. Re-run its installation program (often called `INSTALL`, `SETUP` or `xxINST`) and choose VT100 or ANSI.

CHAPTER 5

The Menu and Settings

Everything that isn't CP/M lives on a single menu, opened with the ⋮ button at the top right of the terminal. The menus on the two platforms share their display options but otherwise differ quite a lot, so this chapter describes each in turn.



The menu on iPhone (left) and Android (right).

5.1 Display effects (both platforms)

Three items change the look of the terminal. Each takes effect immediately, and the menu closes after each tap.

Phosphor colour. Real terminals came with different phosphors, and the app imitates the three classics. Each tap moves to the next colour: green, then amber, then white, then back to green. The item shows the current colour, and the whole app follows it: text, cursor, key-bar labels and menu button.

Colour	Normal text	Bold text	Original
Green	`#33FF33`	`#66FF66`	P1 phosphor, as on early terminals and the Kaypro
Amber	`#FFB000`	`#FFD060`	P3 phosphor, popular in the early 1980s as easier on the eyes
White	`#E0E0E0`	`#FFFFFF`	P4 phosphor, as on most later terminals

The item is called **Phosphor: Green** (and so on) on iPhone and **Screen: GREEN** on Android.

CRT Glow. Draws a soft halo around each character, like the bloom of phosphor on glass. The halo is subtler on iPhone than on Android.

Scanlines. Darkens every other line of pixels, like the visible raster lines of a CRT. Combined with the glow and amber phosphor it makes a convincing old monitor.



Amber phosphor with CRT Glow and Scanlines turned on.

Remembering your choice. On iPhone and iPad the app remembers the phosphor colour, glow

and scanlines, and uses them the next time you open it. On Android these settings last only until you close the app; it always starts in green, with glow and scanlines off.

5.2 The rest of the iPhone and iPad menu

Item	What it does
CP/M Software	Opens the CP/M archive at `cpmarchives.classiccmp.org` in Safari: a large collection of CP/M programs, manuals and source code. Chapter 8 explains how to copy what you download into the app.
YouTube Playlist	Opens a playlist of Pocket CPM videos on YouTube.
Rate App	Opens Pocket CPM's page in the App Store, ready for a review.
Share App	Opens the share sheet with a link to the app, to send to a friend.
Support	Opens the Pymble Software web site.
Privacy Policy	Opens Pymble Software's privacy policy.
Feedback	Opens a new email to Pymble Software's support address, with the app name and your device model already in the subject line.
irc channel	Opens the `#pocket-cpm` chat channel on the Libera.Chat IRC network in iRelayChat, Pymble Software's IRC app. If iRelayChat isn't installed, it opens its App Store page instead.
More Apps	Shows Pymble Software's other apps in the App Store.

The iPhone menu has no settings screen and no import command: you adjust the display from the menu itself, and move files with the Files app (Chapter 8).

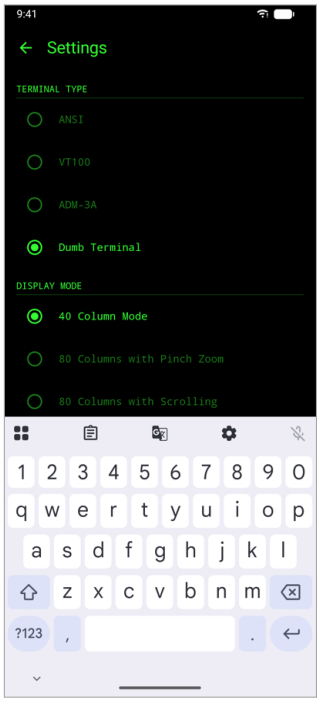
5.3 The rest of the Android menu

Item	What it does
Settings	Opens the Settings screen (§5.4).

Item	What it does
Import File	Opens the system file picker and copies the chosen file to drive B:, user 0 (§8.4).
Show ESC Log / Hide ESC Log	Shows or hides a panel over the bottom of the terminal listing the escape sequences programs have sent, with a description of each (§5.5).
Raw Dump: ON / OFF	(In yellow.) Adds every byte of output, in hex, to the ESC log, and opens the log. A tool for diagnosing misbehaving programs.
Identity: VT100	Chooses how the terminal answers when a program asks what kind of terminal it is (§5.6). Each tap moves on: VT100, VT52, ADM3A, DUMB.
Share App	Opens Android's share sheet with a link to the app on Google Play.
Email Feedback	Opens your email app with a message to Pymble Software's support address, with the app name, phone model and Android version in the subject line.
More Apps	Shows Pymble Software's other apps on Google Play.
YouTube Playlist	Opens a playlist of Pocket CPM videos, in the YouTube app if you have it.

There are no **Rate**, **Support**, **Privacy Policy**, **CP/M Software** or IRC items on Android.

5.4 The Android Settings screen



The Settings screen on Android. Only the bright options can be chosen.

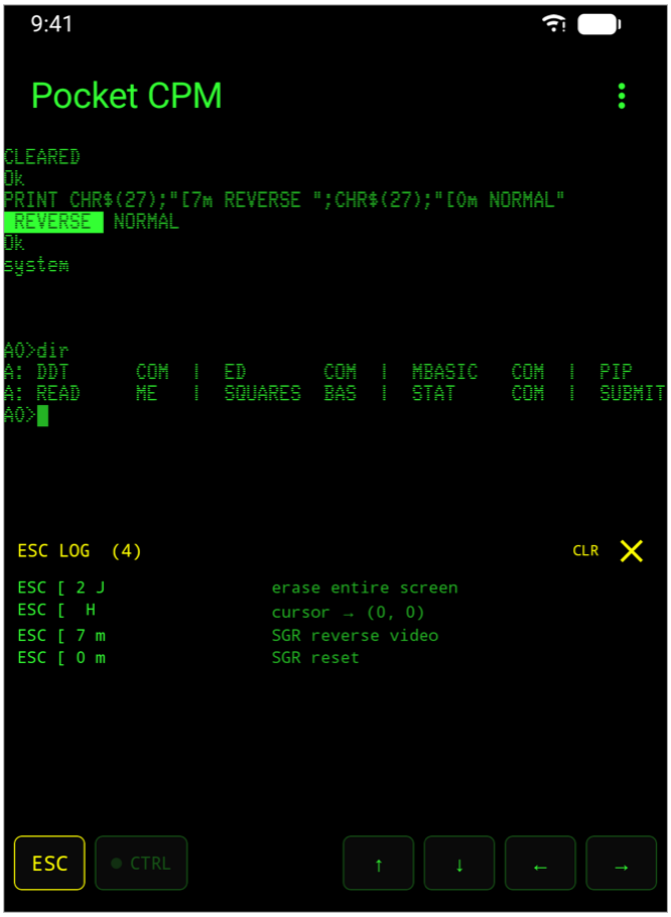
Settings shows two groups of options, with most of them greyed out:

- **TERMINAL TYPE**: **ANSI**, **VT100**, **ADM-3A** and **Dumb Terminal**. Only **Dumb Terminal** can be selected.
- **DISPLAY MODE**: **40 Column Mode**, **80 Columns with Pinch Zoom** and **80 Columns with Scrolling**. Only **40 Column Mode** can be selected.

In this version the screen is a preview of options planned for later releases: the choices have no effect on the terminal, which always works as described in Chapter 4 (an 80-column VT100-style screen you can pinch to zoom), and they reset when you leave the screen. Tap the ← arrow at the top left to return to the terminal.

> **Warning.** Returning from Settings clears the terminal, resets the display effects to green, and, because of a fault in this version, starts a second copy of CP/M that competes with the first for your keystrokes, so typing becomes unreliable (§11.1). Close and reopen the app afterwards. Until this is fixed, there is no reason to open Settings.

5.5 The ESC log (Android)



The ESC log panel on Android, over the bottom of the terminal.

The ESC log is a diagnostic tool, useful when a full-screen program draws its screen wrongly. While it is showing, every escape sequence and control character the terminal receives is added to the list, with an explanation such as `ESC [ 2 J - erase entire screen` or `UNKNOWN final byte … discarded`. It keeps the latest 200 entries. **CLR** clears it; the close button or **Hide ESC Log** hides it.

If the log shows many "unknown" or "ignored" sequences, the program is probably set up for a different terminal (§4.5).

5.6 Terminal identity (Android)

Some programs ask the terminal to identify itself by sending `ESC [ c`, and adjust their behaviour to the answer. The **Identity** item sets the answer:

Identity	The terminal replies
VT100 (the default)	`ESC [ ? 1 ; 2 c`, "a VT100 with advanced video"
VT52	`ESC / Z`
ADM3A	Nothing
DUMB	Nothing

The identity changes only this reply; the terminal still understands the same VT100 sequences whichever you choose. Leave it on VT100 unless a program needs otherwise. The iPhone always answers as a VT100.

CHAPTER 6

Drives, User Areas and Files

A real CP/M computer kept its files on floppy disks. Pocket CPM keeps them in ordinary folders on your device, one folder per drive and, inside it, one folder per user area. Knowing that layout is the key to getting software in and out of the emulator (Chapter 8). This chapter explains it.

6.1 Drives and user areas, inside CP/M

From inside CP/M nothing looks unusual. There are up to sixteen drives, `A:` to `P:`, each with sixteen user areas, `0` to `15`. The prompt shows where you are (`A0>`, `B3>`), you change drive with `B:`, and user area with `USER 3` or `3:` (§3.4). Each drive behaves like a large disk: `STAT` reports the free space on your device's storage, several megabytes at least (`R/W, Space: 7208k` on the iPhone used for this guide), far more than any floppy.

6.2 How drives map to folders

Pocket CPM's emulator (RunCPM) translates every CP/M file into a file on your device, using this rule:

```
<Pocket CPM storage>/<drive letter>/<user area>/<FILENAME.EXT>
```

So `A0:MBASIC.COM` is the file `A/0/MBASIC.COM`, and `B3:NOTES.TXT` is `B/3/NOTES.TXT`. User areas 10 to 15 use a single hexadecimal digit for the folder: user 10 is folder `A`, user 15 is folder `F`. File names are stored in capitals.

CP/M sees	Folder on your device
Drive A:, user 0	`A/0/`
Drive A:, user 1	`A/1/`
Drive B:, user 0	`B/0/`
Drive A:, user 10	`A/A/`
Drive P:, user 15	`P/F/`

Three consequences matter in practice:

- **A drive exists only if its folder exists.** Typing `C:` when there is no `C` folder gives `Bdos Err on C: Select`.

- **A user-area folder is created when you switch to it.** `USER 4` on drive A: creates `A/4/` if it isn't there yet.
- **Anything you put in those folders appears in CP/M**, the next time a program asks for a directory listing. There is no import step and no disk image: copy a file into `A/0/` and `DIR` shows it.

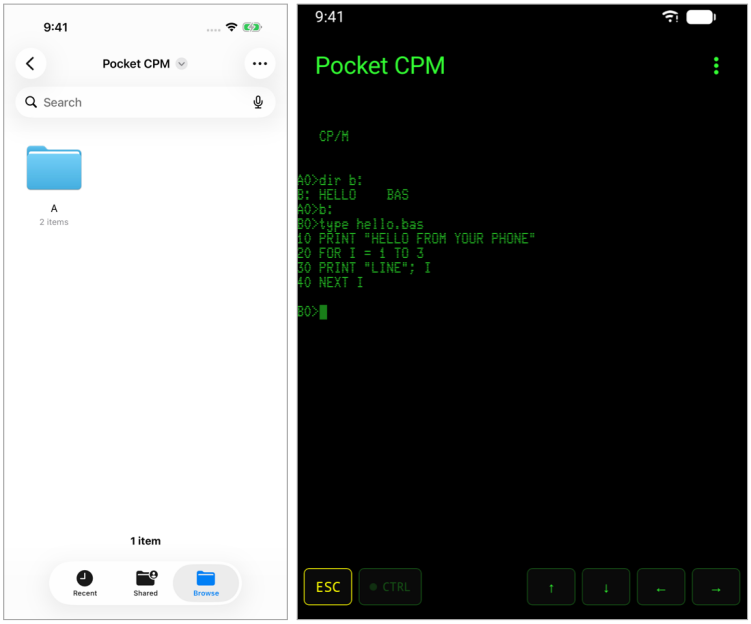
> **Tip.** Give files CP/M-style names before copying them in: no more than eight characters, a dot, and up to three characters of extension, in capitals, with no spaces. A file called `my program.bas` or `README.markdown` is not a valid CP/M name; it may not show up in `DIR`, or show up in a form you can't type.

6.3 Where the storage is on each platform

This is the first big difference between the platforms.

iPhone and iPad. The storage is the app's **Documents** folder. Because the app turns on iOS file sharing, the Files app shows it as **On My iPhone › Pocket CPM** (or **On My iPad › Pocket CPM**), and a Mac or PC can reach it through Finder or the Apple Devices app (§8.3). Each time the terminal opens, the app makes sure `A/0` exists and copies in any of its bundled files that are missing (§9.1). No other drive or user-area folder is created until you, or CP/M, create it.

Android. The storage is the app's private folder on shared storage, `Android/data/com.pymblesoftware.pocketcpm/files/`. Each time it starts, the app creates **all sixteen drives, A to P, each with folders `0` to `15`**, and copies any missing bundled files into `A/0`. So every drive letter works from the first launch. (The folders called `10` to `15` are never used: as §6.2 explains, CP/M names those user areas `A` to `F` and creates those folders itself when you switch to them.)



Left: on iPhone, the Files app shows Pocket CPM's storage with its `A` drive folder. Right: on Android, drive B: exists from the start, here with a file imported into it (Chapter 8).

	iPhone and iPad	Android
Storage	App's Documents folder	`Android/data/com.pymblesoftware.pocketcpm/files/`
Visible in	Files app: **On My iPhone › Pocket CPM**; Finder / Apple Devices on a computer	`adb`, Android Studio's Device Explorer, or a computer over USB on devices that allow it (§8.5)
Drives at first launch	A: only	A: to P:
User-area folders at first launch	`A/0`	`0` to `15` on every drive
Bundled files	Copied into `A/0` if missing	Copied into `A/0` if missing

6.4 The special files

A few files have a special meaning to the emulator:

File	Where	Purpose
`AUTOEXEC.TXT`	Top of the storage, next to the `A` folder	First line runs as a command when CP/M starts (§7.6).
`LST.TXT`	`A/0/`	Everything a program sends to the printer (CP/M's `LST:` device) is written here.
`PUN.TXT`	`A/0/`	Output to the paper-tape punch (`PUN:`), likewise.
`INFO.TXT`	`X/n/`	A label for user area *n* of drive *X*, shown by `VOL` (§7.3).
`$$$.SUB`	`A/` (any user)	The working file `SUBMIT` creates while a batch job runs; deleted when it finishes.

Programs that print through CP/M's standard printer call reach `LST.TXT`; `PIP LST:=REPORT.TXT` "prints" a file there, for example. The file is started afresh the first time something is printed after the emulator starts. Some programs, MBASIC's `LLIST` and `LPRINT` among them, send their output to the printer by a lower-level route (the BIOS) that Pocket CPM ignores, so nothing from them reaches `LST.TXT`.

6.5 Bundled files come back

Every time the terminal starts, the app copies each of its bundled files into `A/0/` **if a file with that name isn't already there**. Two things follow:

- If you erase a bundled program (`ERA PIP.COM`), it reappears the next time you open the app.
- If you replace a bundled file with your own copy of the same name, the app keeps yours; it never overwrites.

Your own files are never touched.

6.6 Keeping your files safe

Your CP/M files are ordinary files in the app's storage, so they are backed up and removed with the app:

- **Deleting the app deletes all its files**, on both platforms. Copy anything you care about out first (Chapter 8).
- On iPhone and iPad the Documents folder is included in iCloud and computer backups of the device.
- On Android, files in an app's `Android/data` folder are removed when the app is uninstalled, and **Clear storage** in the system's app settings also deletes them.

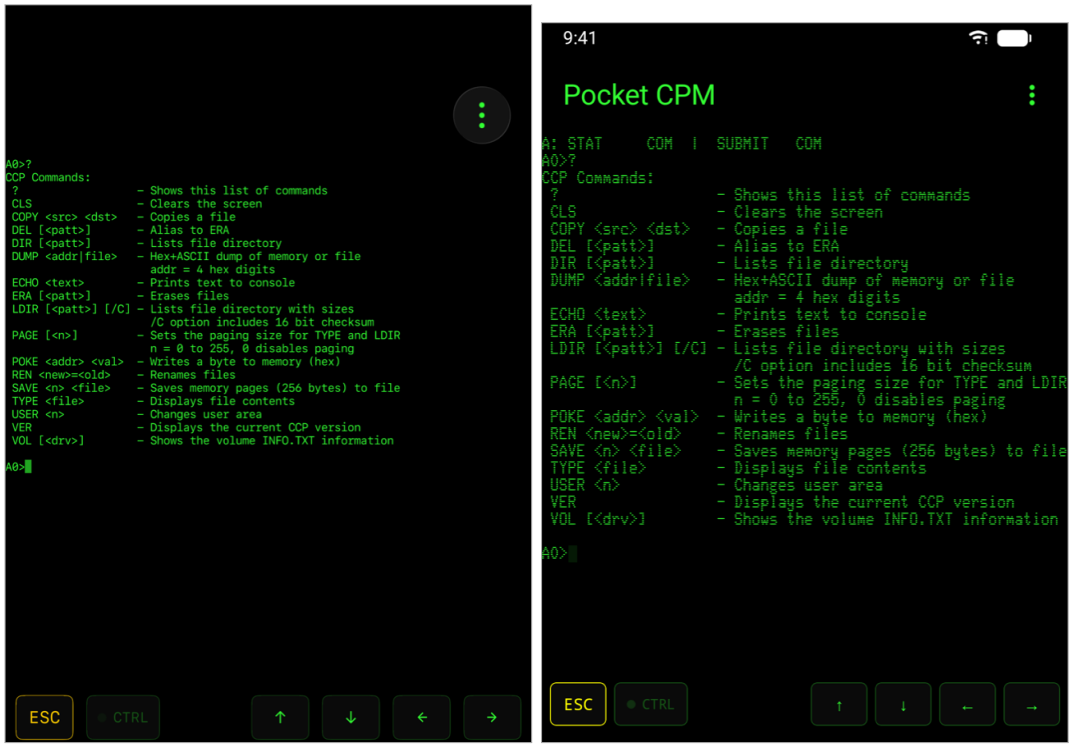
CHAPTER 7

The Command Processor

The *console command processor*, or CCP, is the part of CP/M that prints the prompt, reads your command and either carries it out itself or loads a program from disk. Pocket CPM uses the internal CCP built into its RunCPM emulator core (version 3.3), so it behaves the same on iPhone, iPad and Android. This chapter describes every command it understands.

7.1 Getting help: ?

Type `?` and press **Return** for the built-in list of commands:



The `?` command on iPhone (left) and Android (right).

The list includes `EXIT` among the commands the CCP accepts, but leaves it out of the help text, deliberately: see §7.4.

7.2 The standard CP/M commands

These six were built into every copy of CP/M 2.2.

`DIR [pattern]` lists files in the current user area, four to a line, each line starting with the drive letter. Without a pattern it lists everything; with one it lists the matching files: `DIR *.COM`, `DIR B:*.BAS`, `DIR M*.*`. If nothing matches, it prints `No file`.

```
A0>dir
A: DDT      COM  |  ED       COM  |  PIP      COM  |  READ     ME
A: STAT     COM  |  SUBMIT   COM
```

`TYPE file` displays a text file. Output stops at the end of the file or at a Ctrl-Z character. After every 22 lines it pauses with `-- Press any key, ^C to quit --` (change the number with `PAGE`, §7.3). Try `TYPE READ.ME`.

`ERA pattern` erases files. `ERA OLD.BAS` erases one file; `ERA *.BAK` erases all files with the extension `.BAK`. Unlike some versions of CP/M, Pocket CPM does **not** ask "ALL (Y/N)?" before `ERA *.*`, so take care: an erased file is deleted from your device and can't be recovered from inside the app.

`REN new=old` renames a file. Note the order: the *new* name comes first, as in an assignment.

```
A0>ren squares.bas=sq.bas
```

If the old file doesn't exist you get `No file`.

`SAVE n file` writes `n` pages of memory (256 bytes each), starting at address `0100h`, to a file. It was used to save a program you had just patched in memory with `DDT`. Each page written prints a dot.

`USER n` changes the user area to `n` (0 to 15). A number outside that range is rejected with the number and a question mark (`20?`). The quicker form `n:` does the same (§3.4).

7.3 Extra commands in Pocket CPM

RunCPM's internal CCP adds several commands that CP/M 2.2 never had.

Command	What it does
`LDIR [pattern] [/C]`	A long directory: one file per line with its size in bytes. With `/C`, also a 16-bit checksum of each file, handy for checking a transfer. Pauses every page.
`COPY src dst`	Copies a file: `COPY HELLO.BAS B:HELLO.BAS`. Prints `Copying... Done.` An existing destination is overwritten without warning.
`DEL [pattern]`	Another name for `ERA`.

Command	What it does
`DUMP addr`	Shows 128 bytes of memory in hex and ASCII, starting at a four-digit hex address: `DUMP 0100`. Any key shows the next 128 bytes; Ctrl-C stops.
`DUMP file`	The same for a file: `DUMP PIP.COM`.
`ECHO text`	Prints the text. Useful in `SUBMIT` files.
`PAGE [n]`	Sets how many lines `TYPE` and `LDIR` show before pausing (default 22). `PAGE 0` turns pausing off.
`POKE addr val`	Writes one byte (hex) into memory: `POKE 0100 C9`. Prints `OK`.
`VOL [drive:]`	Lists the user areas that exist on a drive, with the first line of each area's `INFO.TXT` file as a label.
`CLS`	Meant to clear the screen; see the note below.
`VER`	Meant to display the CCP version; see the note below.

Note. In this version of Pocket CPM, `CLS` and `VER` are accepted but do nothing visible: the app's emulator core has the screen-clearing routine and the version banner switched off. To clear the screen, run a program that sends a clear-screen sequence; in MBASIC, `PRINT CHR$(27);"[2J";CHR$(27);"[H"` does it (§10.6).

7.4 EXIT: don't

`EXIT` stops the emulated processor, prints `CPU Halted.` and ends CP/M. There is no way to restart it from inside the app: the screen stays, the keyboard does nothing, and you must close Pocket CPM completely and open it again (on iPhone, swipe up from the bottom and flick the app away; on Android, use the recent-apps screen). Your files are safe; only the session ends. That is why the help list leaves `EXIT` out.

7.5 Running programs

Anything that isn't a built-in command is treated as the name of a program. The CCP looks for `NAME.COM`:

1. on the drive you named (`B:PROG`) or, if you didn't name one, on the current drive, in the current user area;
2. if you didn't name a drive and it wasn't found, on **drive A:, user 0**;
3. if you are in a user area other than 0 and it still wasn't found, in user 0 of the current drive.

This is why the programs that come with the app, which live in `A0:`, can be run from any drive and user area, and why `A0:` is the best place for programs you copy in. If no `.COM` file is found the CCP tries `NAME.SUB` in the same places and, if it finds one, runs it with `SUBMIT` (§7.6). If neither is found it prints the name with a question mark.

Typing a drive letter in front of a name forces the CCP to treat it as a program, even if a built-in command has the same name: if you copy in a program called `CLS.COM`, `A:CLS` runs it rather than the built-in `CLS`.

The program is loaded at address `0100h` and may use memory up to the start of the operating system, about 60 KB. If a file is too large to fit, the CCP prints `No Memory` and loads only the part that fits.

7.6 Batch files: SUBMIT and AUTOEXEC.TXT

A `.SUB` file is a list of commands, one per line. Run it with `SUBMIT NAME` or just `NAME`; the CCP feeds the lines to itself one at a time, showing a `$` instead of `>` in the prompt while it does. `SUBMIT` replaces `$1`, `$2` and so on in the file with the words you type after the file name.

The emulator also looks for a file called `AUTOEXEC.TXT` in the top folder of Pocket CPM's storage, the one that contains the `A` folder (§6.3). If it is there when CP/M starts, its first line (up to 125 characters) is run as if you had typed it. A one-line `AUTOEXEC.TXT` containing `DIR` gives you a directory listing every time you open the app; one containing a `.SUB` file's name runs a whole batch of commands.

7.7 Error messages

Message	Meaning
`NAME?`	Not a built-in command, and no `NAME.COM` or `NAME.SUB` was found. Also shown, with the offending word, for a bad parameter: `USER 20` gives `20?`.
`No file`	`DIR`, `TYPE`, `ERA`, `REN` or `DUMP` found no matching file.
`Bdos Err on X: Select`	Drive X: doesn't exist (no folder for it). Press any key to return to the drive you were on.
`Bdos Err on X: R/O`	The drive has been set read-only (with `STAT X:=R/O`).
`No Memory`	The program is larger than the free memory.
`Source not found`, `Dir full`, `Disk full`	Problems during `COPY`.
`-- Press any key, ^C to quit --`	`TYPE` or `LDIR` has filled a page.

CHAPTER 8

Transferring Files

Sooner or later you will want to run something that didn't come with the app: a game you remember, a compiler, a BASIC program you typed in from a magazine. And you may want to take your own work out again. This is where the iPhone and Android versions of Pocket CPM differ most, so this chapter takes them one at a time and then compares them.

The principle is the same on both: CP/M's drives are folders (Chapter 6), so getting a file into CP/M means putting it in the right folder, and getting it out means copying it from there.

	iPhone and iPad	Android
Copy files in, on the device	Files app: move or copy into **Pocket CPM › A › 0**	⋮ › **Import File**: one file at a time, always into drive **B:** user 0
Copy files in, from a computer	Finder (Mac) or Apple Devices / iTunes (Windows)	`adb push`, Android Studio, or USB file transfer where the device shows `Android/data`
Choose the drive and user area	Yes: any folder you like	Import File: no (always B0:). Computer: yes
Copy files out, on the device	Files app: share, copy or move from **Pocket CPM**	Not possible from the app
Copy files out, to a computer	Finder or Apple Devices / iTunes	`adb pull` or Android Studio, or USB where allowed

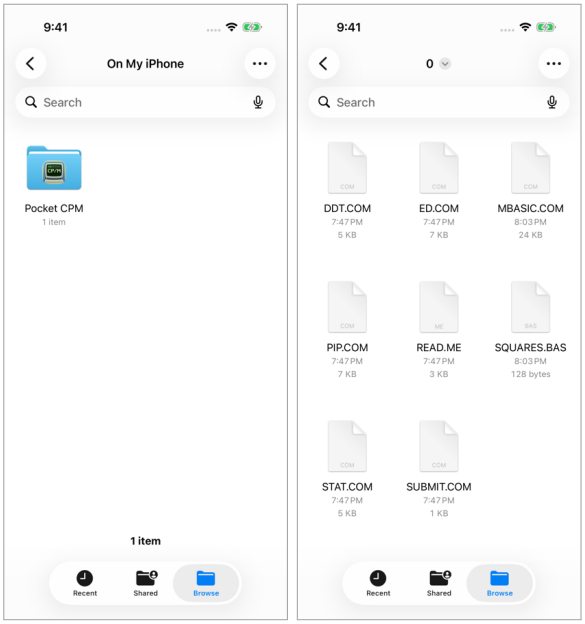
8.1 Before you start: preparing files

- **Use CP/M names.** Up to eight characters, a dot, up to three characters of extension, no spaces: `ADVENT.COM`, not `Colossal Cave (1977).com`. Capitals are traditional and safest. Rename files on your phone or computer before you copy them in.
- **Unpack archives first.** Software archives often come as `.ZIP` files; both platforms can unzip them (the iPhone Files app does it with a tap). CP/M-era formats such as `.LBR` libraries, or "squeezed" files whose extension has a `Q` in the middle (`.DQC`, `.CQM`), need CP/M tools such as `NULU`, `LT` or `USQ`: copy the tool and the archive in and unpack inside CP/M.
- **No disk images.** `.DSK` and `.IMG` files can't be used directly. Extract their contents on a computer first (with a tool such as `cpmtools`), then copy the individual files in.

- **Text files need CP/M line endings.** CP/M text files end each line with carriage return *and* line feed (CR LF). Files made on a Mac, on Linux or on a phone usually end lines with LF alone, and many CP/M programs can't read them: MBASIC, for one, stops with `Line buffer overflow` when you load such a `.BAS` file. Convert line endings to CR LF ("Windows" line endings) in a text editor on your computer before copying the file in. Files saved by CP/M programs themselves are already correct.
- **Check your licence.** Much CP/M software is free to use today, but not all of it. The **CP/M Software** item on the iPhone menu (Chapter 5) opens one large archive.

8.2 iPhone and iPad: using the Files app

Pocket CPM shares its storage with the Files app, which shows it as a folder called **Pocket CPM** under **On My iPhone** (or **On My iPad**). Inside are the drive folders, starting with **A**, and inside each drive, the user-area folders.



The Files app on iPhone. Left: ***On My iPhone****, with the* ***Pocket CPM*** *folder. Right:* ***Pocket CPM*** *›* ***A*** *›* ***0****, drive A: user 0, showing the six files that come with the app and a copied-in* `MBASIC.COM`*.*

8.2.1 Copying a file in

1. **Get the file onto your iPhone.** For example: download it in Safari (it goes to **Downloads**); save an email attachment with **Save to Files**; receive it by AirDrop; or put it in iCloud Drive from a Mac or PC.
2. **Open the Files app** and find the file. If it is a `.zip`, tap it once to unpack it into a folder.

3. **Touch and hold the file**, then tap **Move** (or **Copy**, to keep the original where it is). To do several files at once, tap ··· › **Select**, select them, and tap the folder button at the bottom.
4. In the destination picker, go back to **On My iPhone**, then tap **Pocket CPM** › **A** › **0**.
5. Tap **Move** (or **Copy**).
6. In Pocket CPM, type `DIR`. The file is there. If it is a program, type its name to run it.

To put a file on another drive, create the drive's folder first: in **Pocket CPM**, tap ··· › **New Folder**, name it with a single capital letter (`B`), then inside it create a folder named `0`. CP/M can now use `B:`.

> **Tip.** On iPad you can open the Files app beside Pocket CPM in Split View and drag files straight into the **A** › **0** folder.

8.2.2 Copying a file out

1. In the Files app, open **On My iPhone** › **Pocket CPM** and the drive and user-area folder that holds the file (for example **A** › **0**).
2. Touch and hold the file, then choose **Share** to send it by AirDrop, Mail or Messages, or **Copy** / **Move** to put it in iCloud Drive or another folder.

Text files open in Quick Look when you tap them, which is handy for reading a printer file (`LST.TXT`, §6.4) or a BASIC program saved with `SAVE "NAME",A` (§10.2).

8.3 iPhone and iPad: from a Mac or PC

You can also copy files over a USB cable, without iCloud.

On a Mac (macOS Catalina or later):

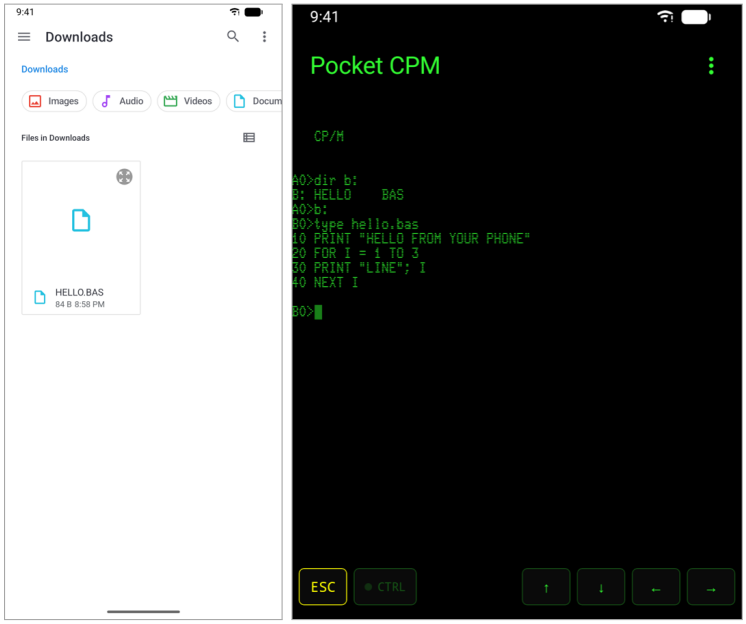
1. Connect the iPhone or iPad with a cable, and unlock it. Tap **Trust** if asked.
2. In Finder, select the device in the sidebar, then click **Files**.
3. Find **Pocket CPM** in the list of apps. Drag files from a Finder window onto it.
4. The files land in the top level of the **Pocket CPM** folder, beside the **A** folder. On the device, open the Files app and move them into **Pocket CPM** › **A** › **0** as in §8.2, steps 3 to 5.

To copy files out, click the disclosure triangle beside **Pocket CPM** in the **Files** list and drag files to your Mac.

On Windows, use the Apple Devices app (or iTunes on older versions of Windows): select the device, choose **File Sharing**, select **Pocket CPM**, and add or save files in the same way.

8.4 Android: Import File

The Android app has an **Import File** item on its ⋮ menu. It opens Android's standard file picker, and copies the file you choose into Pocket CPM's storage.



*Importing on Android. Left: the system file picker, opened by ⋮ › **Import File**. Right: the imported file appears on drive B:, user 0.*

8.4.1 Copying a file in

1. **Get the file onto your phone**, for example by downloading it (it goes to **Downloads**) or saving it from email or Google Drive.
2. In Pocket CPM, tap ⋮ at the top right, then **Import File**.
3. The file picker opens. Find the file; tap ☰ (top left) to switch between **Recent**, **Downloads**, Google Drive and other places.
4. **Tap the file.** The picker closes and you are back at the terminal. The app shows no message, but the file has been copied.
5. The file is on **drive B:, user 0**, under the name it had on your phone. Type `DIR B:` to see it, or `B:` to switch to drive B:.

Three things to know about **Import File**:

- **It always copies to `B0:`.** You can't choose the drive or user area. Use `COPY` (§7.3) to move the file afterwards: `COPY B:ADVENT.COM A:ADVENT.COM`, then `ERA B:ADVENT.COM`.
- **Programs on B: need a drive prefix.** The command processor looks for programs on the current drive and on `A0:` (§7.5), not on B:. Either switch to B: first, type `B:ADVENT`, or copy the program to A:. Data files a program opens on the current drive are easier to use if you work on B:.
- **One file at a time, and same name wins.** You can pick only one file per import, and a file of the same name already on B0: is replaced without warning.

8.4.2 Copying a file out

The Android app has no export command. Its storage is in `Android/data`, which recent versions of Android hide from the Files app and from other apps. To get files out you need a computer (§8.5).

8.5 Android: from a computer

The storage folder is:

```
Android/data/com.pymblesoftware.pocketcpm/files/
```

on the phone's shared storage, with the drive folders `A` to `P` inside.

With adb (the Android Debug Bridge, part of Google's free Android SDK Platform Tools): turn on **Developer options** and **USB debugging** on the phone, connect it, and use:

```
adb push ADVENT.COM /sdcard/Android/data/com.pymblesoftware.pocketcpm/files/A/0/
adb pull /sdcard/Android/data/com.pymblesoftware.pocketcpm/files/A/0/SQUARES.BAS .
```

`push` copies a file in (here straight to `A0:`, so it can be run from anywhere), and `pull` copies one out. `adb shell ls` lists what is there. **Android Studio's Device Explorer** does the same with drag and drop.

With a USB cable and no developer tools: connect the phone, choose **File transfer** in the USB notification, and browse the phone's storage from your computer (on a Mac, with Google's or another MTP client). Many phones show `Android/data/com.pymblesoftware.pocketcpm/files` this way and let you copy files in both directions; some newer ones hide it. If yours hides it, use **Import File** to bring files in and adb to take them out.

8.6 Checking a transfer

After copying a program in, `LDIR /C` lists every file with its exact size in bytes and a checksum (§7.3). Compare the size with the original. Programs (`.COM` files) taken from CP/M disks are almost always a whole number of 128-byte records long; one that isn't may have been damaged or converted on the way, for example by a mail program that treated it as text.

CHAPTER 9

The Software: What Comes With It and What to Add

Pocket CPM comes with the classic Digital Research utilities on drive A:, user 0, the same six files on iPhone, iPad and Android. Everything else, BASIC, compilers, editors and games, you find and copy in yourself. This chapter covers both: what is there from the start, and how to add the rest.

9.1 What comes with the app

File	What it is
`PIP.COM`	Peripheral Interchange Program: copies files and devices
`ED.COM`	Digital Research's line editor
`STAT.COM`	Disk and file statistics; sets file attributes
`DDT.COM`	Dynamic Debugging Tool: debugger and patcher
`SUBMIT.COM`	Runs batch files
`READ.ME`	The legal notice and short instructions for copying in software

Both platforms install the same six files into `A0:` the first time the terminal opens, and put back any that are missing every time after that (§6.5). The only difference is the text of `READ.ME`: each platform's copy gives the file-copying instructions for that platform.

9.2 PIP: copying files

`PIP` (Peripheral Interchange Program) is CP/M's general copier. Its commands look like assignments, destination first:

```
A0>pip
*B:=A:SQUARES.BAS
*
A0>
```

Run without arguments, `PIP` shows a `*` prompt and takes one command per line; an empty line ends it. You can also put the command on the CP/M command line: `PIP B:=A:SQUARES.BAS`.

Command	Effect
`PIP B:=A:*.BAS`	Copy every `.BAS` file from A: to B:
`PIP NEW.TXT=OLD.TXT`	Copy a file under a new name
`PIP ALL.TXT=ONE.TXT,TWO.TXT`	Join two files into one
`PIP CON:=README.TXT`	Show a file on the screen (like `TYPE`)
`PIP LST:=REPORT.TXT`	"Print" a file, to `A0:LST.TXT` (§6.4)
`PIP B:=A:*.*[V]`	Copy everything and verify the copies
`PIP A:=B:FILE.COM[G3]`	Copy from user area 3 (`[G3]`) into the current user area

The built-in `COPY` command (§7.3) is simpler for a single file. `PIP` is the tool for wildcards, devices and user areas.

9.3 ED, STAT, DDT and SUBMIT

ED is the line editor Digital Research supplied with CP/M: powerful, terse and unforgiving. `ED NOTES.TXT` opens (or creates) a file and shows a `*` prompt. Type `#A` to load the whole file, `I` to start inserting text (end with Ctrl-Z), `B#T` to type the whole file from the beginning, and `E` to save and exit. `Q` quits without saving. Most people today prefer to write text on their phone or computer and copy it in (Chapter 8).

STAT reports on disks and files. `STAT` alone shows the free space on each drive in use; `STAT *.*` lists files with their sizes in records and kilobytes; `STAT DSK:` describes the drive; `STAT USR:` lists the user areas that hold files. `STAT NAME.COM $R/O` makes a file read-only (and `$R/W` undoes it).

DDT is the Dynamic Debugging Tool, for 8080 machine code: `DDT PROG.COM` loads a program, `L` lists (disassembles) it, `D` dumps memory, `S` changes bytes, `T` traces instructions, `G` runs it and `GO` returns to CP/M. After patching a program in memory, leave DDT with `GO` and use the built-in `SAVE` command to write it back (§7.2).

SUBMIT runs a batch file of commands (§7.6). A file `BUILD.SUB` containing

```
ECHO Building $1
ERA $1.BAK
STAT $1.*
```

is run with `SUBMIT BUILD REPORT`, which substitutes `REPORT` for `$1` on every line.

9.4 Adding software

The real fun of CP/M is the thousands of programs written for it. Adding one takes three steps.

1. **Find it.** The **CP/M Software** item on the iPhone and iPad menu opens `cpmarchives.classiccmp.org`, one of the largest CP/M archives (on Android, open the same address in your browser). Other long-standing sources include the Walnut Creek CP/M CD-ROM collection and the Interactive Fiction Archive's CP/M games.
2. **Prepare it.** Unpack archives, give files CP/M names, and convert text files to CR LF line endings (§8.1). Many programs come with companion files, such as overlays, data files or a configuration program, so copy the whole set, not just the `.COM` file.
3. **Copy it in** (Chapter 8):
   - **iPhone and iPad:** with the Files app, into **Pocket CPM › A › 0** (or any drive and user-area folder).
   - **Android:** with ⋮ **› Import File**, which puts each file on **drive B:, user 0**. Run programs there as `B:NAME`, or switch to `B:` first, or copy them to A: with `COPY B:NAME.COM A:NAME.COM` so they can be run from anywhere (§7.5). With a computer, `adb push` can copy straight into `A/0` (§8.5).

Make sure you have the right to use what you copy in: much CP/M software is free today, but not all of it.

Popular things to try:

Program	What it is
Microsoft BASIC (`MBASIC.COM`)	The BASIC interpreter most CP/M owners used (§9.5)
Turbo Pascal 3	Borland's famous fast Pascal compiler and editor
BDS C, Small-C	C compilers
Microsoft MACRO-80 (`M80`, `L80`)	A macro assembler and linker
WordStar 3 or 4	The word processor that defined the era
dBASE II, SuperCalc	Database and spreadsheet
Colossal Cave Adventure, Zork	Text adventures (Zork needs its `.DAT` story file as well as the `.COM`)
NSWEEP	A popular file-management utility

When a program asks what terminal you have, choose VT100 or ANSI (§4.5).

9.5 MBASIC

Microsoft BASIC-85 isn't included with Pocket CPM; once you have copied `MBASIC.COM` in (recipe 10.1), it is the easiest way to start programming. Version 5.29 was the BASIC interpreter most CP/M owners used. Start it with `MBASIC`, or with `MBASIC NAME` to load and run `NAME.BAS` straight away.



A short program in MBASIC, after copying `MBASIC.COM` in: on iPhone (left) and Android (right).

The essentials:

Command	What it does
`10 PRINT "HI"`	A line with a number is stored as part of the program
`LIST`	Shows the program; `LIST 100-200` shows part of it
`RUN`	Runs the program; Ctrl-C stops it
`NEW`	Clears the program from memory
`SAVE "NAME"`	Saves it as `NAME.BAS` in BASIC's compact format
`SAVE "NAME",A`	Saves it as plain text, which you can read with `TYPE` and copy out
`LOAD "NAME"`	Loads `NAME.BAS`
`FILES`	Lists the files on the current drive
`SYSTEM`	Leaves BASIC and returns to CP/M

MBASIC shows `39224 Bytes free` at start-up: plenty for a BASIC program. Recipe 10.1 shows how to copy it in on each platform, and recipes 10.2 and 10.3 use it.

If MBASIC stops with `Line buffer overflow` when loading a `.BAS` file, the file has Unix line endings; convert it to CR LF (§8.1).

CHAPTER 10

Recipes

Step-by-step answers to the things people most often want to do with Pocket CPM. Each recipe says which platform it is for when that matters.

10.1 Install MBASIC

Pocket CPM doesn't include Microsoft BASIC on either platform, but if you have `MBASIC.COM` (for example from a CP/M archive, §9.4) it takes a minute to install. The same steps work for any other program.

On iPhone or iPad:

1. Save `MBASIC.COM` to your iPhone: from Safari it goes to **Downloads**; from a Mac, AirDrop it.
2. Open the **Files** app, find `MBASIC.COM`, touch and hold it, and tap **Copy**.
3. Go to **On My iPhone › Pocket CPM › A › 0** and tap **Copy** (§8.2).
4. In Pocket CPM, type `DIR`. `MBASIC COM` is in the list.
5. Type `MBASIC`. You should see `BASIC-85 Rev. 5.29` and `Ok`.

On Android:

1. Save `MBASIC.COM` to your phone, for example by downloading it (it goes to **Downloads**).
2. In Pocket CPM, tap ⋮ › **Import File** and choose `MBASIC.COM` (§8.4). It is copied to **drive B:, user 0**.
3. Copy it to drive A: so it can be run from anywhere, and tidy up B::

```
A0>copy b:mbasic.com a:mbasic.com
A0>era b:mbasic.com
```

4. Type `DIR`. `MBASIC COM` is in the list. Type `MBASIC` to start it.

(With a computer you can skip step 3: `adb push MBASIC.COM` straight into `…/files/A/0/`, §8.5.)



`DIR` after copying `MBASIC.COM` into drive A:, user 0, on iPhone (left) and Android (right). The other five files are the ones the app comes with.

Because `MBASIC.COM` is in `A0:`, you can start it from any drive or user area.

10.2 Write, save and list a BASIC program

1. Type `MBASIC` to start BASIC.
2. Type the program, pressing **Return** after each line:

```
10 FOR I = 1 TO 5
20 PRINT I, I * I
30 NEXT I
```

3. Type `RUN`. BASIC prints the numbers 1 to 5 and their squares.
4. Save it as text: `SAVE "SQUARES",A`. BASIC adds `.BAS`.
5. Type `SYSTEM` to return to CP/M, and `TYPE SQUARES.BAS` to see the saved file.



The program saved as text and shown with `TYPE`.

The `,A` matters: without it BASIC saves in a compact "tokenised" form that only BASIC can read. With it, the file is plain text that you can copy out and read on any device (recipe 10.4).

To come back to it later: `MBASIC SQUARES` loads and runs it; or start `MBASIC` and type `LOAD "SQUARES"`, then `LIST`.

10.3 Type in a BASIC program on your phone and run it

Typing a long listing on a phone keyboard into the terminal is slow. It is easier to write it in a notes or text-editor app and copy the file in.

1. Write the program in a text editor, one numbered line per line:

   ```
   10 PRINT "HELLO FROM YOUR PHONE"
   20 FOR I = 1 TO 3
   30 PRINT "LINE"; I
   40 NEXT I
   ```

2. Save it as plain text with a CP/M name: `HELLO.BAS`. Make sure the file uses CR LF line endings (§8.1); on a phone, the simplest way is to write it on a computer, or in an editor that offers "Windows (CRLF)" line endings.

3. **iPhone:** move it into **Pocket CPM › A › 0** with the Files app and type `MBASIC HELLO`.

   Android: use ⋮ › **Import File**; it lands on B:, so type `B:` and then `MBASIC HELLO` (MBASIC itself, copied in by recipe 10.1, is found on `A0:`; the program is loaded from the current drive, B:).



On Android: back at the terminal after choosing the file (left), and the imported program running from drive B: (right).

10.4 Get a file out of Pocket CPM

iPhone and iPad: in the Files app, open **On My iPhone › Pocket CPM › A › 0**, touch and hold the file, and choose **Share** (to send it by AirDrop, Mail or Messages) or **Copy** (to paste it into iCloud Drive or another folder).

Android: the app can't export files. Connect the phone to a computer and run

```
adb pull /sdcard/Android/data/com.pymblesoftware.pocketcpm/files/A/0/SQUARES.BAS .
```

or use Android Studio's Device Explorer, or USB file transfer if your phone shows `Android/data` (§8.5).

10.5 Organise files in user areas

User areas are a good way to keep projects apart.

```
A0>user 1
A1>pip squares.bas=a:squares.bas[g0]
A1>dir
A: SQUARES  BAS
A1>era squares.bas
```

`[G0]` tells PIP to take the source file from user area 0. Programs such as `MBASIC` and `PIP` are still found while you are in user area 1, because the command processor also looks in `A0:` (§7.5). On your device, user area 1 is the folder `A/1`.

10.6 Clear the screen

The built-in `CLS` doesn't clear the screen in this version (§7.3). Two ways that do:

- **In MBASIC:** `PRINT CHR$(27);"[2J";CHR$(27);"[H"` sends the VT100 "erase screen" and "cursor home" sequences. Put it at the start of your programs, for example as line `5`.
- **From CP/M:** a one-line `.SUB` file won't help (CP/M has no command that prints raw characters), but a two-line MBASIC program saved as `CLS.BAS` will: run it with `MBASIC CLS`, ending the program with `SYSTEM` so it returns to CP/M.

```
10 PRINT CHR$(27);"[2J";CHR$(27);"[H";
20 SYSTEM
```

Form feed (Ctrl-L, character 12) also clears the screen, so `PRINT CHR$(12);` works too.

10.7 Start every session with your own command

Create a text file called `AUTOEXEC.TXT` containing one command, such as `DIR` or `STAT`, and copy it into the top level of Pocket CPM's storage, beside the `A` folder: on iPhone, **On My iPhone › Pocket CPM**; on Android, `Android/data/com.pymblesoftware.pocketcpm/files/` (with adb, §8.5). The next time CP/M starts it runs that command before showing the prompt (§7.6). To run several commands, put them in a `.SUB` file in `A0:` and make the `AUTOEXEC.TXT` command its name.



With `DIR *.COM` in `AUTOEXEC.TXT`, the Android app lists the programs on drive A: as soon as it starts (here the five that come with the app, plus a copied-in `MBASIC.COM`).

10.8 Make another drive (iPhone and iPad)

1. Open the Files app and go to **On My iPhone › Pocket CPM**.
2. Tap ··· (top right), then **New Folder**, and name it `B`.
3. Open **B**, create another folder inside it named `0`.
4. In Pocket CPM, type `B:`. The prompt becomes `B0>`.

Repeat for `C` to `P` if you want more drives. On Android, drives A: to P: already exist.

10.9 Check that a copied program arrived intact

`LDIR /C` lists every file with its size in bytes and a 16-bit checksum:

```
A0>ldir mbasic.com /c
```

Compare the size with the original on your computer. If you know the checksum from another RunCPM system, compare that too (§8.6).

CHAPTER 11

Troubleshooting and FAQ

11.1 The screen and keyboard

The keyboard has disappeared. Tap anywhere on the black terminal and it comes back.

The text is too small to read. Pinch outwards on the terminal to zoom in, and drag to move around (§4.2). Landscape helps too.

I typed a command and got the command back with a question mark. CP/M didn't recognise it: either it is misspelt or the program's `.COM` file isn't on the current drive, the current user area, or `A0:` (§7.5). Type `DIR` to check. A program on drive B: must be run as `B:NAME` or from drive B: (§8.4).

I pressed an arrow key and got `[A?`. The arrow keys send VT100 sequences for full-screen programs; the command line doesn't understand them (§4.4). Use Ctrl-W to recall the last command and Ctrl-A / Ctrl-F to move along the line.

Ctrl-C does nothing. At the CP/M prompt, Ctrl-C only works on an empty line. In programs, it depends on the program: MBASIC stops a running program with it, but many programs ignore it.

`CLS` doesn't clear the screen. In this version the built-in `CLS` does nothing visible (§7.3). Use a program that sends `ESC [ 2 J`, such as MBASIC (§10.6).

`VER` prints a blank line. Also expected in this version (§7.3).

A full-screen program prints junk like `[2J`, `=`, or `Y` followed by letters. The program was installed for a different terminal, probably an ADM-3A, Kaypro or VT52. Run its installation program and choose VT100 or ANSI (§4.5). On Android, **Show ESC Log** lets you see exactly what the program sent (§5.5).

My typing comes out scrambled on Android, and the screen was cleared. You have probably visited the **Settings** screen. In this version, coming back from Settings rebuilds the terminal and starts a second copy of CP/M alongside the first; from then on each key you type goes to one copy or the other, so commands arrive with letters missing (`dir` becomes `DI?`). Close Pocket CPM from the recent-apps screen and open it again, and avoid the Settings screen until a later version fixes this (its options don't do anything yet, §5.4). The rest of the menu is safe to use. If typing is scrambled without a visit to Settings, check that your keyboard's auto-correct and suggestions are off.

11.2 Drives and files

`Bdos Err on B: Select`. Drive B: doesn't exist. On iPhone and iPad only drive A: exists to begin with: create a folder named `B` in **Pocket CPM** with the Files app, and a folder named `0` inside it (§8.2). On Android all drives A: to P: exist from the start, so this only happens if the folders have been deleted; restarting the app recreates them. Press any key after the message to carry on.

I copied a file in but `DIR` doesn't show it.

- Check it is in a user-area folder, `A/0`, not loose in `A` or at the top of **Pocket CPM**.
- Check you are looking in the right place: `DIR` shows the current drive and user area only. Try `DIR B:` or `USER 0`.
- On Android, **Import File** puts files on **B:**, not A: (§8.4).
- Check the name is a valid CP/M name: eight characters, a dot, three characters, no spaces (§8.1).

A deleted program came back. Bundled files are copied into `A0:` whenever they are missing, so erasing one only lasts until the next start (§6.5).

I erased a file by mistake. `ERA` deletes the file from your device. There is no undelete. If you have a copy in a backup of your device, restore it from there.

Where is my printer output? In `A0:LST.TXT` (§6.4), for programs that print through CP/M's printer call, such as `PIP LST:=FILE.TXT`. The file is started afresh the first time something is printed after the emulator starts, so copy it out before you print again in a new session. MBASIC's `LLIST` and `LPRINT` output is discarded (§6.4); to get a BASIC listing out, save the program as text with `SAVE "NAME",A` and copy `NAME.BAS` out instead.

MBASIC says `Line buffer overflow` when I load a program. The `.BAS` file has Unix-style line endings (LF only). Convert it to CR LF on a computer and copy it in again (§8.1).

11.3 Programs

A program I copied in says it can't find a file, or isn't configured. Many CP/M programs come with companion files: overlays, data files (Zork, for example, needs its `ZORK1.DAT` story file beside `ZORK.COM`), or a configuration program to run once to choose your terminal. Copy the program's whole distribution onto the same drive and user area, and when asked for a terminal type choose VT100 or ANSI (§4.5).

After `EXIT` nothing works. `EXIT` stops CP/M completely. Close the app fully and open it again (§7.4).

A program has hung. Try Ctrl-C (tap **CTRL**, then **c**), then ESC. If neither works, close the app fully and reopen it. Any files the program had already saved are safe.

A program says it is out of memory, or the CCP says `No Memory`. About 60 KB is available to programs, as on a well-equipped CP/M 2.2 machine. Programs written for CP/M 3 with banked memory, or for MP/M, won't run.

A program needs CP/M 3 (CP/M Plus) or MS-DOS. Pocket CPM runs CP/M 2.2 programs for the 8080 and Z80. Programs for CP/M 3 that use its extra system calls, for CP/M-86, or for MS-DOS won't work.

11.4 Frequently asked questions

Is it really CP/M? The operating system calls (the BDOS and BIOS) are emulated in C by the RunCPM emulator rather than running Digital Research's original machine code, and the command processor is RunCPM's own. To programs, though, it behaves like CP/M 2.2, and the bundled utilities (`PIP`, `ED`, `STAT`, `DDT`, `SUBMIT`) are the original Digital Research programs.

How fast is it? Much faster than any real CP/M machine: the emulated Z80 runs as fast as your device allows, not at the 2 to 4 MHz of the originals. Games that time themselves by counting instructions may run too fast.

Can I use a disk image? Not directly; extract the files first (§8.1).

Can I print? Not to a real printer. Printer output goes to `A0:LST.TXT`, which you can then share or print from your phone (§6.4).

Does it need the internet? No. Only the web links on the menu do.

Does it collect any data? The app asks for no permissions. Its menu links open the publisher's web site, which has its privacy policy.

Where can I get help? Use **Feedback** (iPhone) or **Email Feedback** (Android) on the menu to email the publisher. On iPhone, **irc channel** opens the `#pocket-cpm` chat channel.

APPENDIX A

iPhone and Android Compared

The iPhone and iPad version (1.0.0) and the Android version (1.0.1) run the same emulator core, so everything inside CP/M (the prompt, the built-in commands, the file layout, program behaviour) is identical. The apps around it were written separately. This table lists every difference a user is likely to notice; each was checked in the apps' source code and on screen.

Feature	iPhone and iPad	Android
Version and system	1.0.0; iOS / iPadOS 26.2 or later	1.0.1; Android 9 (API 28) or later
Start-up	Typed-out *Legacy Software Notice*; tap to skip, or it moves on after about 20 seconds	App icon briefly, then straight to the terminal
Title bar	No title; status bar hidden; round ⋮ button	**Pocket CPM** title and ⋮ button; status bar shown
Terminal size	80 × 24	80 × 25
Terminal font	System monospaced font	Glass TTY VT220
Tab character	Ignored	Moves to the next 8-column tab stop
132-column mode (`ESC [ ? 3 h`)	Clears the screen; stays 80 columns	Switches to 132 columns
Cursor position report (`ESC [ 6 n`), status (`ESC [ 5 n`)	Not answered	Answered
Cursor sequences `ESC [ n E`, `F`, `G`	Not supported	Supported
CTRL latch	Works with any key (keeps the low 5 bits)	Works with A–Z; other keys are sent unchanged
Back button	(none)	Doesn't hide the keyboard on the terminal screen
Phosphor colour item	**Phosphor: Green / Amber / White**	**Screen: GREEN / AMBER / WHITE**
CRT Glow	Subtle halo	Stronger halo

Feature	iPhone and iPad	Android
Display settings remembered	Yes: colour, glow and scanlines persist	No: reset to green, glow off, scanlines off at every launch
Settings screen	None	**Settings**: terminal type and display mode (preview only; no effect yet). Returning from it resets the terminal and starts a second CP/M session (§11.1)
ESC log and Raw Dump	Not available	**Show ESC Log**, **Raw Dump**
Terminal identity (`ESC [ c` reply)	Always VT100	**Identity**: VT100, VT52, ADM3A or DUMB
Web and help items	**CP/M Software**, **YouTube Playlist**, **Rate App**, **Share App**, **Support**, **Privacy Policy**, **Feedback**, **irc channel**, **More Apps**	**Share App**, **Email Feedback**, **More Apps**, **YouTube Playlist**
Storage location	App's Documents folder (**On My iPhone › Pocket CPM**)	`Android/data/com.pymblesoftware.pocketcpm/files/`
Drives at first launch	A: only (folder `A/0`)	A: to P:, each with folders `0` to `15`
Copying files in on the device	Files app, into any drive / user folder	⋮ › **Import File**: one file at a time, always to B0:
Copying files in from a computer	Finder (Mac); Apple Devices / iTunes (Windows)	adb, Android Studio, or USB file transfer where `Android/data` is visible
Copying files out	Files app (share, copy, move); Finder / Apple Devices	Computer only (adb, Android Studio, USB where allowed)
`READ.ME` text	Copying instructions for the iPhone Files app	Copying instructions for Android

The same on both: the RunCPM-based emulator (Z80, CP/M 2.2, internal CCP 3.3); the `A0>` prompt and all built-in commands (Chapter 7); the folder layout of drives and user areas; the six files that come with the app (`DDT.COM`, `ED.COM`, `PIP.COM`, `STAT.COM`, `SUBMIT.COM` and `READ.ME`), copied into `A0:` on first run and restored when missing; `AUTOEXEC.TXT`, `LST.TXT` and `PUN.TXT`; the key bar (**ESC**, **CTRL**, four arrows); pinch-to-zoom and drag; VT100 cursor, erase and attribute sequences; green, amber and white phosphors with the same colours; scanlines; and English as the only language.

APPENDIX B

Quick Reference

B.1 Built-in commands

Command	Does
`?`	List the built-in commands
`DIR [pat]`	List files (`DIR *.COM`, `DIR B:`)
`LDIR [pat] [/C]`	List files with sizes (and checksums)
`TYPE file`	Show a text file, a page at a time
`ERA pat` / `DEL pat`	Erase files (no confirmation)
`REN new=old`	Rename a file
`COPY src dst`	Copy a file
`SAVE n file`	Save `n` 256-byte pages from `0100h`
`USER n` / `n:`	Change user area (0–15)
`X:` / `Xn:`	Change drive (and user area)
`PAGE [n]`	Lines per page for `TYPE` and `LDIR` (0 = no pause; default 22)
`DUMP addr` / `DUMP file`	Hex dump of memory or a file
`POKE addr val`	Write a byte to memory (hex)
`ECHO text`	Print text
`VOL [X:]`	List user areas on a drive, with their `INFO.TXT` labels
`CLS`, `VER`	Accepted, but do nothing visible in this version
`EXIT`	Halts CP/M; the app must be closed and reopened
`name [args]`	Run `name.COM` (or `name.SUB`) from the current drive/user, then `A0:`

B.2 Line-editing keys (at the prompt, in ED and PIP)

Key	Does	Key	Does
Ctrl-H, Delete	Delete left	Ctrl-G	Delete under cursor
Ctrl-A	Cursor left	Ctrl-F	Cursor right
Ctrl-B	Start / end of line	Ctrl-K	Delete to end of line
Ctrl-X	Delete to start of line	Ctrl-U	Delete whole line
Ctrl-W	Recall last command	Ctrl-R	Retype line
Ctrl-E	New screen line	Ctrl-C	Abandon (empty line only)
Return	Enter the line	Ctrl-Z	End of input (ED, PIP)

B.3 The key bar

Key	Sends
ESC	`1Bh`
CTRL then a key	The control character (Ctrl-A = `01h` … Ctrl-Z = `1Ah`)
↑ ↓ → ←	`ESC [ A`, `ESC [ B`, `ESC [ C`, `ESC [ D`

B.4 Terminal escape sequences

`ESC` is character 27 (`1Bh`); `n`, `r`, `c` are decimal numbers. In MBASIC, send them with `PRINT CHR$(27);"[2J";`.

Sequence	Meaning	iPhone	Android
`ESC [ r ; c H` or `f`	Move cursor to row `r`, column `c` (from 1)	✓	✓
`ESC [ H`	Cursor home	✓	✓
`ESC [ n A` / `B` / `C` / `D`	Cursor up / down / right / left `n`	✓	✓
`ESC [ n E` / `F`	Cursor to start of line `n` down / up		✓
`ESC [ n G`	Cursor to column `n`		✓
`ESC [ J`, `ESC [ 0 J`	Erase to end of screen	✓	✓

Sequence	Meaning	iPhone	Android
`ESC [ 1 J`	Erase to start of screen	✓	✓
`ESC [ 2 J`	Erase whole screen	✓	✓
`ESC [ K`, `ESC [ 0 K`	Erase to end of line	✓	✓
`ESC [ 1 K` / `ESC [ 2 K`	Erase to start of line / whole line	✓	✓
`ESC [ 0 m`	Normal characters	✓	✓
`ESC [ 1 m` / `2` / `4` / `5` / `7` / `8` m	Bold / dim / underline / blink / reverse / invisible	✓	✓
`ESC [ 22/24/25/27/28 m`	Turn the above off	✓	✓
`ESC [ 30–47 m`	Colours	ignored	ignored
`ESC [ ? 5 h` / `l`	Reverse / normal screen	✓	✓
`ESC [ ? 3 h` / `l`	132 / 80 columns	clears only	✓
`ESC [ c`	Identify (reply depends on Identity, §5.6)	VT100	✓
`ESC [ 5 n` / `ESC [ 6 n`	Status / cursor position report		✓
`ESC 7` / `ESC 8`	Save / restore cursor	✓	✓
`ESC D` / `ESC M` / `ESC E`	Index / reverse index / next line	✓	✓
`ESC c`	Reset and clear screen	✓	✓
Ctrl-L (`0Ch`)	Clear screen	✓	✓
Tab (`09h`)	Next tab stop		✓

B.5 Useful BDOS functions for programmers

Load the function number into register C, the parameter into DE (or E), and `CALL 0005h`.

C	Function	C	Function
1	Console input	2	Console output (E)
5	Printer output (to `A0: LST.TXT`)	6	Direct console I/O

C	Function	C	Function
9	Print string ending in `$` (DE)	10	Read console line (DE)
11	Console status	12	CP/M version
14	Select drive (E)	15 / 16	Open / close file (DE = FCB)
17 / 18	Search first / next	19	Delete file
20 / 21	Read / write next record	22	Make file
23	Rename file	25	Current drive
26	Set DMA address	32	Get / set user area
33 / 34	Read / write random	35	File size

B.6 Where things are

Item	iPhone and iPad	Android
Storage	**On My iPhone › Pocket CPM**	`Android/data/com.pymblesoftware.pocketcpm/files/`
Drive A:, user 0	`A/0`	`A/0`
Bring a file in	Files app → `A/0`	⋮ › **Import File** → B0:
Take a file out	Files app → Share	`adb pull`
Printer output	`A/0/LST.TXT`	`A/0/LST.TXT`
Start-up command	`AUTOEXEC.TXT` beside `A`	`AUTOEXEC.TXT` beside `A`

APPENDIX C

Glossary

8080. Intel's 8-bit microprocessor of 1974, the processor CP/M was written for.

8.3 name. A file name of up to eight characters, a dot, and an extension of up to three: `MBASIC.COM`.

A0>. The prompt: drive A:, user area 0, ready for a command.

adb. The Android Debug Bridge, a command-line tool from Google that copies files to and from an Android device connected to a computer (§8.5).

ADM-3A. A cheap and popular "dumb" video terminal from Lear Siegler (1976), with a very small set of control codes. Many CP/M programs offer it as a choice.

ANSI. The standard (ANSI X3.64) behind the VT100's escape sequences; "ANSI terminal" usually means "VT100-compatible".

ASCII. The 7-bit character code CP/M uses for text.

AUTOEXEC.TXT. A file whose first line the emulator runs as a command when CP/M starts (§7.6).

BDOS. Basic Disk Operating System: the part of CP/M that programs call (at address `0005h`) for console, file and disk services.

BIOS. Basic Input/Output System: the machine-specific part of CP/M that talks to the hardware. In Pocket CPM it is emulated.

CCP. Console Command Processor: the part of CP/M that shows the prompt and runs commands (Chapter 7).

.COM file. A CP/M program, loaded at `0100h` and run by typing its name.

Control character. A character below 32 (or 127), typed with the Ctrl key, that controls the terminal or a program rather than printing.

CP/M. Control Program for Microcomputers: Digital Research's operating system for 8080 and Z80 computers. Pocket CPM emulates version 2.2.

CR LF. Carriage return and line feed (characters 13 and 10), CP/M's end-of-line marker in text files.

CRT. Cathode-ray tube: the glass screen of old terminals, imitated by the app's glow and scanline effects.

DDT. Dynamic Debugging Tool, Digital Research's machine-code debugger (§9.3).

Digital Research. Gary Kildall's company, which published CP/M.

DMA address. The memory address where CP/M reads or writes the next 128-byte disk record.

Drive. One of CP/M's disks, `A:` to `P:`. In Pocket CPM, a folder (Chapter 6).

ED. CP/M's line editor (§9.3).

Emulator. A program that imitates another computer. Pocket CPM emulates a Z80 CP/M machine.

Escape sequence. The ESC character followed by others, telling the terminal to move the cursor, clear the screen and so on (§4.5).

FCB. File Control Block: the 36-byte structure a CP/M program uses to name and access a file.

Files app. Apple's file manager on iPhone and iPad, through which you reach Pocket CPM's storage (§8.2).

Import File. The Android menu item that copies a file into drive B:, user 0 (§8.4).

LST:. CP/M's printer device. In Pocket CPM, output to it goes to `A0:LST.TXT`.

MBASIC. Microsoft BASIC-85 for CP/M. Not included with the app; copy it in yourself (§9.5, §10.1).

Phosphor. The coating inside a CRT that glows. Green (P1), amber (P3) and white (P4) phosphors were common; the app imitates all three (§5.1).

PIP. Peripheral Interchange Program, CP/M's file copier (§9.2).

Record. CP/M's unit of disk storage: 128 bytes.

RunCPM. The open-source CP/M 2.2 emulator by Marcelo Dantas on which Pocket CPM's emulator core is based.

STAT. CP/M's disk and file statistics program (§9.3).

SUBMIT. CP/M's batch-file runner (§7.6, §9.3).

TPA. Transient Program Area: the memory, starting at `0100h`, where programs run. About 60 KB in Pocket CPM.

Transient command. A program on disk (a `.COM` file), as opposed to a built-in command.

User area. One of sixteen subdivisions (0 to 15) of each CP/M drive. In Pocket CPM, a subfolder of the drive folder (§3.4, §6.2).

VT52 / VT100. DEC video terminals of 1975 and 1978. The VT100's escape sequences became the standard that Pocket CPM's terminal follows.

Warm boot. Returning from a program to the CP/M prompt, reloading the command processor; Ctrl-C at the prompt triggers one.

Wildcard. `?` (any one character) or `*` (any characters) in a file name pattern (§3.5).

Z80. Zilog's 1976 microprocessor, compatible with the 8080 and with extra instructions. Pocket CPM emulates a Z80.

Index

Numbers

8080, 10, 40, 53, 60

A

adb, 28, 34, 38, 41, 44, 48, 55, 59, 60
ADM-3A, 18, 22, 23, 25, 51, 55, 60
AirDrop, 35, 44, 47
Android Studio, 28, 38, 47
Android/data, 27, 29, 34, 38, 47, 48, 55, 59
ANSI, 13, 17, 18, 23, 41, 51, 52, 60
Apple Devices, 27, 34, 36, 55
arrow keys, 1, 15, 17, 51
`AUTOEXEC.TXT`, 29, 33, 48, 55, 59
`AUTOEXEC.TXT`, 60

B

BDOS, 53, 60
BIOS, 10, 29, 53, 60

C

CCP, 1, 11, 12, 30, 31, 32, 33, 37, 48, 52, 53, 55, 60
`CLS`, 32, 33, 48, 56
`CLS`, 51
`.COM files`, 12, 31, 33, 38, 41, 51, 56
`.COM files`, 49, 60
control keys, 7, 13, 17, 24, 57
`COPY`, 12, 31, 33, 37, 40, 41, 56
CRT Glow, 20, 54
CTRL key, 1, 6, 7, 13, 15, 52, 54, 57

D

`DDT`, 6, 31, 53
`DDT`, 40, 61
`DEL`, 31, 56
Digital Research, 1, 6, 10, 11, 39, 40, 53, 60
`DIR`, 3, 6, 10, 12, 27, 33, 36, 37, 44, 48, 51, 56
`DIR`, 14, 31, 52
disk images, 1, 27, 34, 53
drive, 1, 2, 11, 26, 27, 31, 33, 34, 49, 52, 55
`DUMP`, 12, 32, 33, 56

E

`ECHO`, 32, 56
`ED`, 6, 13, 53, 57
`ED`, 40, 61
`ERA`, 12, 29, 31, 33, 37, 52, 56
`ERA`, 31
error messages
  Bdos Err, 11, 26, 33, 52
  Line buffer overflow, 35, 43, 52
  No file, 31, 33
  No Memory, 33, 52
ESC key, 1, 13, 15, 16, 17, 22, 25, 51, 52, 54, 57, 61
ESC log, 22, 51, 55
escape sequences, 13, 17, 22, 24, 60
`EXIT`, 30, 32, 56
`EXIT`, 52

F

file names, 27, 46, 52, 60
Files app, 1, 21, 27, 34, 35, 38, 41, 47, 49, 52, 55, 59, 61
Finder, 27, 34, 36, 55

I

iCloud Drive, 35, 47
Import File, 1, 22, 34, 36, 38, 41, 44, 47, 52, 55, 59, 61
`INFO.TXT`, 29, 32, 56

K

key bar, 1, 2, 6, 7, 13, 14, 16, 55
Kildall, Gary, 10, 61

L

`LDIR`, 12, 31, 33, 38, 49, 56
Legacy Software Notice, 4, 54
line editing, 13, 39, 40, 61
line endings, 35, 41, 43, 46, 52, 60
`LST.TXT`, 11, 29, 36, 40, 52, 53, 55, 58, 61

M

MBASIC, 12, 13, 16, 29, 32, 35, 41, 44, 45, 47, 48, 51, 52, 57
MBASIC, 48, 61
menu, 2, 4, 15, 19, 21, 51, 53

P

PAGE, 31, 32, 56
phosphor colour, 1, 19, 54, 61
PIP, 6, 13, 29, 39, 48, 52, 53, 57
PIP, 61
POKE, 32, 56
prompt, 1, 2, 5, 7, 10, 11, 12, 13, 17, 26, 30, 33, 40, 48, 49, 51, 54, 60
PUN.TXT, 29, 55

R

Raw Dump, 22, 55
READ.ME, 4, 6, 31, 39, 55
REN, 10, 12, 33, 56
REN, 31
RunCPM, 2, 11, 26, 30, 31, 50, 53, 61

S

SAVE, 12, 36, 40, 42, 45, 52, 56
SAVE, 31
scanlines, 20, 55
Settings screen, 21, 23, 51, 55
STAT, 6, 26, 33, 48, 53
STAT, 40, 61
.SUB files, 12, 29, 33, 48
SUBMIT, 6, 11, 12, 29, 32, 33, 53
SUBMIT, 40, 61

T

terminal identity, 22, 25, 55, 58
TPA, 61
TYPE, 4, 6, 10, 12, 15, 32, 33, 40, 42, 45, 56
TYPE, 23, 31

U

USER, 11, 12, 26, 27, 33, 52, 56
USER, 31
user area, 1, 2, 6, 11, 26, 27, 29, 31, 32, 34, 37, 40, 45, 48, 51, 52, 55, 56, 59, 60

V

VER, 32, 56
VER, 51
VOL, 29, 32, 56
VT100, 13, 15, 17, 22, 23, 25, 41, 48, 51, 52, 55, 58, 60
VT52, 18, 22, 25, 51, 55, 61

W

warm boot, 13
wildcards, 12, 40

Z

Z80, 1, 10, 53, 55, 60
zooming, 15, 23, 51



ABOUT THE PUBLISHER

Pymble Software

Pymble Software is a Sydney software company, in business since 2006, led by a developer with 35 years in the industry. We consult on software projects (apps, payment gateways, banking and security), make games and tools for iPhone and Android, and write about what we've learned.

Our apps include Pocket CPM, Pocket ARM, Web3 Inspector, iRelayChat, Shieldkeep, Orbital Fury, Noir Descent, Rescue Rapunzel, Plate Panic and Ticker Tape Tycoon.

Find them all, and our books, at pymblesoftware.com.

A 1979 computer in your pocket.
Here's how to drive it.

Pocket CPM runs CP/M 2.2, the operating system of the Kaypro, the Osborne and a thousand other early business computers, on an emulated Z80 inside your iPhone, iPad or Android phone. Type `DIR` at the `A0>` prompt and you are back in 1980.

This free companion guide explains every part of the app on both platforms, and the CP/M you need to get real work and real fun out of it. You'll learn how to:

- find your way round drives, user areas and the `A0>` prompt;
- use every built-in command and the bundled **PIP**, **ED** and **STAT**;
- add BASIC, compilers and games of your own, starting with **MBASIC**;
- **copy software in and out**, step by step, with the iPhone Files app or Android's Import File;
- tune the terminal: green, amber or white phosphor, CRT glow and scan lines;
- fix the things that trip people up, with recipes and a full iPhone-versus-Android comparison.

Written for anyone who remembers CP/M, and anyone curious about where personal computing began.

CP/M 2.2 · Z80 · iPhone, iPad & Android · Free Edition