

PIDP-10
SYSTEMS USER'S GUIDE



THIS IS WORK-IN-PROGRESS
but will be updated weekly from here on

Table of Contents

Introduction.....	4
Credit where credit is due.....	4
The PiDP-10 concept.....	6
Using the PiDP-10.....	7
Installing.....	7
Updating the PiDP-10 software.....	8
Current issues.....	8
Quick summary.....	9
PDP commands through the GUI Desktop.....	10
PDP View.....	11
Using a laptop as a remote PiDP-10 terminal.....	12
PiDP-10 Special functions under PAR STOP.....	13
Basics: Getting into ITS.....	14
ITS walkthrough: Light Entertainment on the Mainframe.....	15
Introduction.....	15
Starting the ITS Walkthrough.....	15
Logging in on the Knight TV.....	18
Hacking spacewar and using tvwar.....	19
Adding a VT-52 terminal session, some Lisp therapy.....	21
Multiple jobs.....	23
The Ultimate 1970s AI demo: SHRDLU.....	24
Remote Knight TV sessions from your laptop.....	26
Multi-Player Maze War.....	27
Gosper's Game of Life: the original.....	28
Digging deeper into ITS.....	28
Shutting down gracefully.....	28
Bubble Universe - Democoding on the Color Scope.....	30
Front panel operations – a primer.....	32
Indicator lights.....	32
Switches.....	32
Preparation.....	33
Using the front panel.....	33
Depositing and examining data into memory.....	34
Program Control.....	34
Debugging with the front panel: breakpoints.....	34
PDP-10 Machine Language.....	35
Machine code on the simh command line.....	36
Assembly in ITS: MIDAS and DDT.....	36
A first look at TOPS-10.....	37
Starting up.....	37
Login on a user terminal.....	38
Fortran.....	39
Assembly.....	40
Logging out and shutting down.....	40

TOPS-20: Manual section coming ASAP.....	42
Networking on the PDP-10.....	45
File exchange between ITS and Linux.....	45
Terminal sessions over Chaosnet: supdup.....	45
Chaosnet.....	45
Future Networking sections.....	46
Further reading.....	47
Appendix: Running the PiDP-10 software on X86 Linux.....	47
Appendix: assigned telnet port numbers for the PiDP-10.....	48
Appendix: A Multi-Pi PiDP-10.....	49

Relevant links for further information:

PiDP-10 & its constituent parts:

About the PiDP replica 'range': <https://www.obsolescence.dev/>

PiDP-10 details: <https://obsolescence.wixsite.com/obsolescence/pidp10>

PiDP-10 github: <https://github.com/obsolescence/pidp10>

RPDP for PiDP-10 github: <https://github.com/obsolescence/rpdp>

PDP-10 simulator source code: <https://github.com/rcornwell/sims>

ITS:

ITS Reconstruction Project sources & research: <https://github.com/PDP-10/its>

Modern ITS manual in the making: <https://github.com/larsbrinkhoff/its-manual/wiki>

Another useful reference: <https://its.victor.se/wiki/start>

TOPS-10:

TOPS-10 how-to & FAQ: <https://www.quentin.org.uk/tops-10-faq/>

TOPS-20:

PiDP-10 Google Group's thread: <https://groups.google.com/g/pidp-10/c/-UTX6Jx0bXc>

DEC Manuals:

Essentials: lower left-hand side of <https://obsolescence.dev/pidp10>

Bitsavers Collection: <http://www.bitsavers.org/pdf/dec/pdp10>

Operating the Front Panel:

Youtube demonstration: <https://youtu.be/XFmEHmySNT0>

Test & Learn: github.com/obsolescence/pidp10/wiki/PiDP%E2%80%9010-testing

Introduction

Thank you for adopting a 1960s mainframe!

The purpose of the PiDP-10 project is to keep hands-on knowledge of the PDP-10 alive, and to help build a new user community to the only personable mainframe in history.

This manual is meant as a first introduction, as many of the PiDP-10 resources are online for in-depth information. Links to those are provided; and the manual focuses on being a quick get-to-know guide for the machine. To get you started.

Much attention is given to the **ITS** operating system and all the surrounding MIT AI Lab hardware. The next page shows you just how much hardware there is to play with, inside the PiDP-10.

But equally important, DEC's own **TOPS-10** operating system is fully supported too, it is the other boot option for the PiDP-10. We hope that there will be a lively new community around TOPS-10 as well.

The PiDP-10 is a model KA10. But separate simulators are added for other PDP-10 models (KI, KL, KS) and thus, **TOPS-20** can also be run on the PiDP-10. The front panel will not be entirely complete for these other PDP-10 systems, but in fact, most of them did not even have a front panel. So we'd argue the PiDP-10 is not at a disadvantage.

Credit where credit is due

The PiDP-10 wraps the work of many people into its physical front panel.

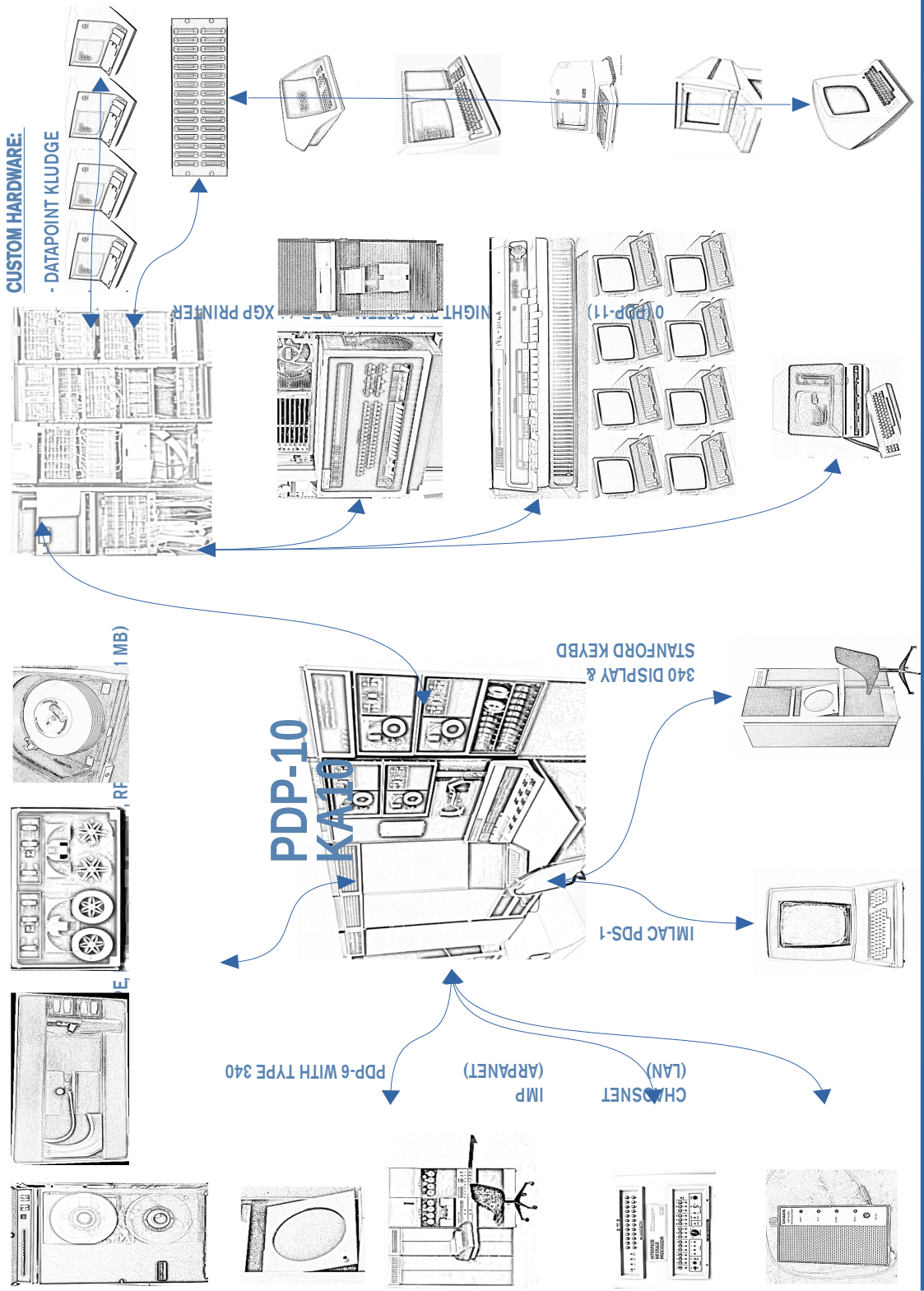
@rcornwell wrote the PDP-10 KA10 simulator (the Engine), @larsbrinkhoff did the heavy lifting on the ITS Reconstruction Project, @aap added the PDP-6, Knight TV and many other elements.

But they built on a community of dozens, going back a long time. Ken Harrenstein's emulator in 1992 was the start, and he did the first work on a Personal ITS distribution. In the early 2000's ITS was also running on Bob Supnik's SIMH.

Huge strides were made by @rcornwell adding ITS support to his KA10 emulator. Work with @larsbrinkhoff during fall 2017 resulted in ultimately booting this very accurate KA10 simulator successfully on January 8, 2018. Much of the following AI lab emulation followed from of this.

@eswenson1 and @atsampson did almost everything related to Maclisp, Macsyma, Muddle, and Zork. @bictorv is the cherished champion of Chaosnet. Without the effort of @lisper et al we wouldn't have the wealth of backup data to draw from. And ITS would only have been a whisper of a ghost of its former grandeur. And of course, ITS preservation started already in the 80s with @bawden and many others, many of whom are true heroes of computer history.

ITS RECONSTRUCTION PROJECT: HARDWARE OVERVIEW



The PiDP-10 concept

The goal was to set up the PiDP-10 such that it is:

- **Dual-hearted.** The real CPU offers you a normal Pi, the ‘soft CPU’ a PDP-10. The PiDP-10 is both *at the same time*.
- **Configurable.** The PDP-10 can be used *either* as an ‘all-in-one’ on the Pi’s HDMI monitor (easy to start with, everything is on your screen), *or* (more authentic) ‘headless’. In the latter approach, you connect from remote terminals. The HDMI display is now only used for the type 340 graphics display. The remote terminals can be real serial terminals, terminal simulator programs run on a laptop (see `rpdp` further on), or plain telnet sessions. The simulation can be controlled from a ssh command line, through the `pdp` and `pdpcontrol` commands. So the PiDP-10 can live on a bookshelf, if you wish.

It helps to know how the PiDP-10 actually works:

1. The entire package is located in `/opt/pidp10`.
`bin/` contains all binaries, `systems/` contains disk images and boot scripts for each boot option, `install/` contains the install script, `src/` the source code.
2. When you log in to the Pi, `pdpcontrol start` is run automatically. Look inside the script that does this: `opt/pidp10/bin/pdpcontrol.sh`
3. `Pdpcontrol` calls the `scansw10` program to read the front panel switches¹. The bootscript corresponding to the number read from the switches is looked up in the text file `systems/selections`, then passed to the emulator (`simh`) to configure for that OS.
 - You can add your own customised boot options simply by adding subdirectory entries to `systems/selection`. System subdirectories must contain a `boot.pidp` (used for the PiDP) and `boot.pi` (used for ‘Naked Pi’s’, without PiDP-10 hardware).
4. `Simh` is run inside a `screen` virtual terminal, in the background. You can call it up whenever you have a need to use it. Just run the command `pdp`. And `Ctrl-A d` to leave.
5. The chosen bootscript configures the simulated hardware, assigning disk image files to simulated disk drives, simulated serial ports to telnet ports. Have a look at `/opt/pidp10/systems/its/boot.pidp`

At this stage, then, you have a `simh` emulator running in a hidden virtual ‘screen’ terminal, listening for terminals plugged in to telnet ports. What you connect to the telnet ports is up to you. It could be the standard Linux telnet on the Pi or on another computer. It could also be a nice graphical emulator of a VT 52 (run on the Pi, or on a laptop). Whatever it is, it connects through a telnet port².

¹If you do not like this inauthentic step, you can take it out of the ‘`pdpcontrol.sh`’ script that does this.

²If you have a real serial terminal, you can reconfigure the bootscript to not use telnet but an actual Pi serial port.

Using the PiDP-10

Installing

Prepare a regular Raspberry Pi SD card using the Raspberry Pi Imager program. Make sure you pick the **64 bit** version of the OS. At the time of writing, the latest OS version was '**Bookworm**'; older versions of the OS will **not** work. Boot up the Pi, and get Wifi going. Now, do:

```
cd /opt
sudo git clone https://github.com/obsolescence/pidp10
opt/pidp10/install/install.sh
```

The install script will ask you a lot of questions. Just say 'yes' to all of them, except perhaps the 'install source code' and 'install source code dependencies' sections; they take a lot of time and disk space. It is always safe to re-run the install script, answering 'no' to questions means no action is taken. Answering 'yes' will update that aspect of the setup.

- No PiDP-10 front panel? You can also run the software in just a 'Naked Pi'.
- No Pi at all? You can also get the PiDP-10 package going on a regular Linux computer. See the Appendix.

Set the bottom row of switches to select the operating system that you want to boot:

- Setting SR 35 (rightmost switch, bottom row) will prepare to boot ITS;
- Setting no switches (or not having a PiDP-10 front panel!) will run the Blinky demo

(see next page for all OS boot options)

Now reboot. The front panel will light up when you log in again on your Pi. The tell-tale sign that you have chosen the ITS boot option is a quick running lights effect on the bottom row of lights on the front panel.

If you actually don't have a PiDP-10 front panel plugged into the Pi: type '`pdpcontrol stop`', wait a few seconds, type '`pdpcontrol start 1`'. The 1 tells the simulator it should not read the switch settings but just boot option 1, ITS.

The PDP-10 is now running the boot option you selected, waiting for you to connect through simulated Teletypes or terminals to its ports.

The following assumes you run a Pi with the GUI on a HDMI monitor. None of that is necessary, you can also run the PiDP-10 'headless', but this is the Quick Start after all:

- Double-click the View icon on the top left of the Pi's GUI desktop, choose 'Execute' (alternatively, type '`pdp view`' on the Pi's command line).

The PDP View window will pop up. You can open and close it as you wish, it will not interfere with the running PDP-10 system. It is **not** the PDP-10 simulator. Just a convenient view into it.

Updating the PiDP-10 software

As of early May 2024, the PiDP-10 is quite new and we add features/squash bugs regularly. **To keep your software up to date**, this will generally suffice:

```
cd /opt/pidp10
git pull
```

If you hit a problem, however,

- re-run the install script.
- or just delete the entire pidp10 subdirectory and reinstall as per the previous page

Of course, the above will Never Be Necessary™, but just in case ;-)

Current issues

2024-05-01 Autostart does not show the type340 display window

Occurs when you did a headless install.

Issue:

The Pi just started to use Wayland instead of X11 for its GUI. But Wayland wants GUI programs to autostart from its own wayfire.ini, instead of the normal .profile config file that, oh, everyone used since about the seventies <grumble>. So we need to support one method of autostarting the PiDP-10 for Wayland, and another way for X11. Brilliant.

Solution:

- If you installed the PiDP-10 headless, but *later on* decide to use the HDMI monitor to view the graphics displays on the default Wayland setup:
 - Run the install script again from inside the GUI, and only say Yes to "Automatically start the PiDP-10 ...?". The install script will adjust the autostart.
- If you at some point decide to switch from Wayland to X11 (because you hate Wayland):
 - run the install script again and only say Yes to "Automatically start the PiDP-10 ...?".

Quick summary

Normally, you set the data switches before powering up (or restarting pdpcontrol) to choose a boot option.

Currently available boot options:

```
-----  
001   ITS           KA10  
002   TOPS-10      KA10  
200   TOPS-20      KL10  
-----
```

Any other number on the data switches will start Blinky – a little blinkenlights demo program.



If you are not yet familiar with octal front panels: the 9 switches above form a 3 digit octal number. So 200 means setting the ‘2’ switch on the left-hand group of three dark blue switches; 070 would mean you set all three switches on the middle group of three light blue switches ($4+2+1=7$).

The PDP-10 simulator is started in the background when you log in to the Pi.

Only if you selected ITS: the ITS boot configuration keeps things very authentic. You have to press **STOP**, then **READ IN** on the front panel to get the system booted off the virtual paper tape bootstrap. This is only when you start the PiDP-10 through the data switches on the front panel, not if you start it using `pdpcontrol start 1`

pdpcontrol: from the Linux command line, starts, stops, restarts the PDP-10.

```
pdpcontrol stop          - halts the PDP-10 simulator  
pdpcontrol start        - starts with the boot option set at the data switches  
pdpcontrol start x      - starts with the bootscript x, ignoring the data switch settings  
                        (used if you have a Naked Pi without PiDP-10 hardware!)  
pdpcontrol              - asks you what to do, interactively
```

pdp: when the PDP-10 is running, manages various hardware.

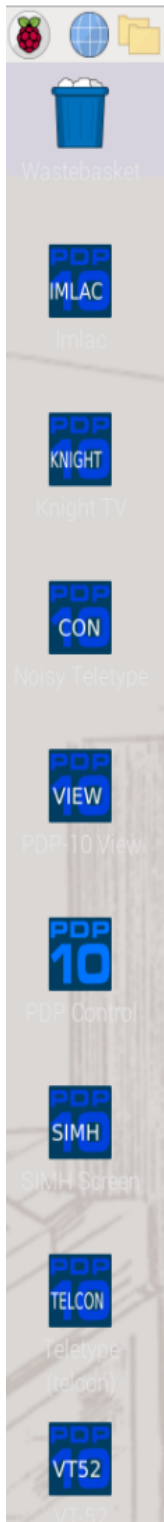
```
pdp con                 - starts a noisy Teletype simulator on the console (only on HDMI display)  
pdp telcon              - starts a quiet telnet session on the console instead (anywhere)  
pdp vt52                - starts a VT52 terminal (only on HDMI display)  
pdp tvcon               - starts a Knight TV terminal (only on HDMI display)  
pdp ?                   - see all the other terminal and display options: imlac, vt05, dp3300, etc  
pdp                     - brings you into the simh simulator itself  
                        CTRL-E: halts the simulator and brings you into the simh command line  
                        continue: continues the simulator  
                        CTRL-A d: leaves the simh simulator to run in the background again  
pdp view                - starts an all-in-one overview
```

Using a laptop to get a terminal on the PiDP-10

For this, install the `rpdp` package. See ‘Using a laptop as a remote PiDP-10 terminal’.

PDP commands through the GUI Desktop

The desktop icons on the Pi's desktop can be used instead of typing `pdp` and `pdpcontrol` on the command line, if you prefer. It's the same thing, really. Each desktop icon represents one of the `pdp` commands; `pdpcontrol` is represented by the plain 'PDP-10' icon.



Quick overview of how to go through startup using the desktop icons: suppose you've booted up with 001 on the data switches to boot ITS, then:

- Press **STOP**, then **READ IN** on the front panel to get the system booted off the virtual paper tape bootstrap (this step is only for boot option 001, ITS)
- Double-click the CON or TELCON desktop icons to get the Teletype (noisy or quiet)
- Type `its<esc>g` to boot on the Teletype when you see DSKDMP
- Double-click the TVCON, VT-52 or IMLAC icons to get a user terminal.

Again, this does nothing else than regular `pdp` commands from the command line, and the same icons are in the Pi's Apps drop-down menu if you prefer that.

If double-clicking desktop icons gives you an annoying 'Execute?' dialog box:

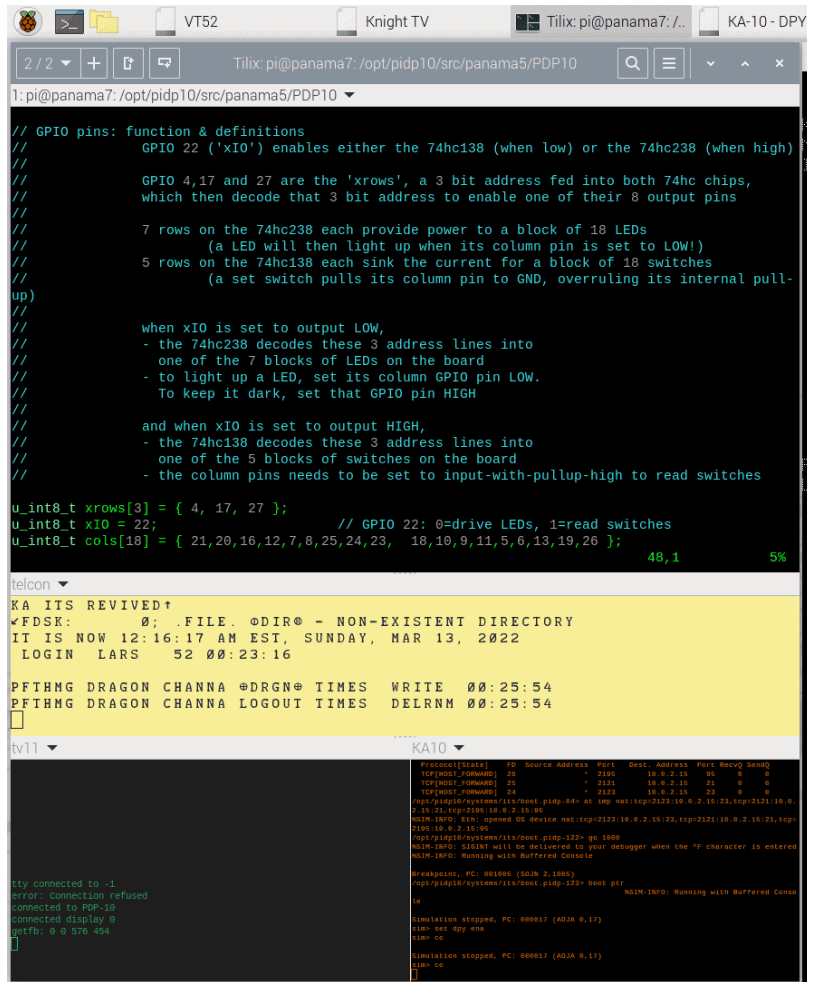
- Launch the Pi's File Manager,
- go to the Edit-->Preferences menu item,
- enable "Don't ask options on launch executable file'.

It seems there is no failsafe way to automate this for you in the install script (let me know if you do have a way!)

PDP View

pdp view is *just* a handy all-in-one widget when you use the Pi's HDMI monitor. It does nothing that the pdp and pdpcontrol commands won't do for you on the command line.

- the **yellow** window is the Teletype on PDP-10's console port. Required for the boot process, and will regularly print out system status messages.
- at the **bottom right** is a view on the PDP-10 simulator. Useful to check its status, but also, CTRL-E will bring you into the simh command line, which is very powerful, including PDP-10 debugging and a mini-assembler/disassembler. Ctrl-Shift-+ increases font size, and there is a maximise icon to zoom in.
- at the **bottom left** is a view on the PDP-11 running the Knight TV terminals. Not very useful, but just to check if any Knight TV terminals are active.



The large window at the top is simply a generic Linux command line, to use as you see fit. Three obvious uses for PDP-10 purposes:

- enter **pdp** commands: For example, pdp tvcon gets you a Knight TV terminal.
- enter **pdpcontrol** commands: Control the PDP-10 itself. Principally, pdpcontrol stop and pdpcontrol start. One special function: no matter what the front panel switches are set to, for example, pdpcontrol start 1 will boot ITS.
- **use as a PDP-10 terminal.** Type telnet localhost 10018 to use the top window for a simple login to the ITS system. In this case, it's a plain-vanilla telnet login. Once logged in to ITS (see below) use :tctype vt to tell ITS to use the right terminal control codes.

So, consider pdp view a handy all-in one gadget on top of the actual pdp and pdpcontrol commands.

Using a laptop as a remote PiDP-10 terminal

Most of this manual assumes you do everything on the PiDP-10 itself. But the more ‘normal’ approach would be to place the PiDP-10 on a bookshelf, with perhaps a little HDMI monitor just for the type340 and/or Colour Scope display.

User terminals would then be run on your laptop, from the sofa (or, if you set up Linux’s networking for it, remotely from anywhere else). Remote control actually takes two parts: (1) use ssh to issue `pdpcontrol start` and `stop` commands; (2) use the `rpdp` package for any user terminals. A PDP-10 is a time-sharing system: users can have many terminals open from anywhere.

(1) ssh connection for pdpcontrol

Make sure that ssh is enabled on your Pi. The install script does this for you, if you said ‘yes’ to that. Then, assuming the name of your PiDP-10 is ‘pdp10’, just do ssh pi@pdp10.local from your laptop. Now you have the Pi’s command line to issue `pdpcontrol stop/start x` commands.

(2) remote terminal connections: rpdp

On a Linux laptop (or Windows 11 laptop with the WSL subsystem installed), do this to install the `rpdp` package:

```
cd /opt
sudo git clone https://github.com/obsolescence/rpdp
/opt/rpdp/install/install-rpdp.sh
```

The install script will then ask you for the hostname of your PiDP-10 (pdp10, for example) and the user name on the Pi (pi, for example) and set things up accordingly.

Use:

Just like the `pdp` command on the PiDP-10 itself, so type `rpdp vt52` for a VT-52 terminal, etc. Don't forget to press F11 for full-screen mode.

```
rpdp con          -- Noisy & Slow Teletype console
rpdp telcon      -- Quiet & Fast Teletype console
rpdp             -- Go into the simh simulator, Ctrl-A d to leave
rpdp tvcon       -- Knight TV terminal
rpdp vt52        -- DEC VT-52 terminal
rpdp dp3300      -- Datapoint 3300 terminal
rpdp imlac       -- Imlac terminal/minicomputer
rpdp tek         -- Tektronix 4010 storage tube graphics terminal
```

Don't forget, if you want to switch Teletypes in mid-flight, leave the first Teletype properly (Ctrl-], quit) or the line will not be free for the new one.

Of course, you can also just use plain old telnet itself: `telnet pidp10.local 1025` gets you to the console's Teletype (assuming your PiDP-10 has hostname `pidp10`); `telnet pidp10.local 10018` to a regular ITS user terminal.

PiDP-10 Special functions under PAR STOP

You can skip this page if you are new to the PiDP-10.

These are functions found on a real KA10's Maintenance Panel, which you will not normally need.

The **PAR STOP** switch has no function in the PiDP-10 replica, because we can be confident that there will never be a parity error that would actually trigger this function (i.e., stop on parity error).

Therefore, **PAR STOP** is reconfigured to be an 'escape' switch. As long as it is enabled, it allows you to see and set switches normally found on a real KA10's maintenance panel:

Maintenance panel functions with **PAR STOP** set:

- **READ IN** switch:
 - See the readin device: the current value of the readin device is displayed in the MA register, bits 27-33 as long as **PAR STOP** is enabled.
 - Set the readin device: set address switches 27-33 to form the readin device number, then press **READ IN** to store it on the virtual maintenance panel.
 - Just as in the real panel, the two least significant bits cannot be set. Readin addresses always end in either 0 or 4.

Note: you can also set the readin device with "dep readin #" on the simh command line, or simh boot script. See boot.pidp in /opt/pidp10/systems/its for an example, where it is set at 104 to select the paper tape reader.

- **CONTINUE** switch:
Toggles the MI disable switch, which shows up as bit 25 of the MA when PAR STOP is set.
- **EXAMINE** switch:
Loads the upper 4 address switches into the repeat rate control. This is shown in bits 20-23.
- **STOP** switch:
Will power the machine off (exit simh).



Later in 2024, we will make this replica Maintenance Panel for hardcore KA-10 aficionados.

With the explicit comment that *you do not need it!*

Normal PDP-10 operation leaves the Maintenance Panel switches untouched. It is just an ornament.

Basics: Getting into ITS

We will assume that you installed the PiDP-10 software, after setting switch SR35 to boot up the ITS operating system. You will see a quick ‘running lights’ effect on the bottom row of front panel lights. This is just a tiny program that the PDP-10 is running, waiting for you to do more meaningful things. Like load the bootstrap off paper tape.

If you have a PiDP-10:

- Press the **STOP** switch to halt the PDP-10, and then the **READ IN** switch. This triggers a circuit in the PDP-10 to start reading a paper tape, and then run the last instruction found on that tape. It’s the standard way to bring up a PDP-10.

If, instead, you have a ‘Naked Pi’, no PiDP-10 front panel:

- Do a `pdpcontrol stop`, wait a few seconds, then `pdpcontrol start 1`.

You will see DSKDMP appear on the yellow Teletype window.

It was just loaded from the paper tape reader, waiting for your input.

Type `its<return><esc>g` and DSKDMP will comply by booting ITS off disk. Some messages pass by, including ‘TV11 IS DOWN’, then, ‘TV11 IS UP’. This means that the PDP-11 that supports the Knight TV terminals is connected.

From here, you are free to use ITS. On any terminal, you need to ‘CALL’ the computer

- On the Teletype: Press `CTRL-Z` (CALL) gets you to the command line.
- On a Knight TV Terminal: run `pdp tvcon` from the Linux command line. A Knight TV terminal will appear on screen. Hit `F1` (CALL) to get into ITS. **F1 on a Knight TV terminal is the same as CTRL-Z on any other terminal.**
- On any other terminal type: on the Linux command line, do `pdp ?` to see the available terminals. `pdp vt52` will get you a vt52 terminal window, for instance. Again, press `CTRL-Z` (‘CALL’) to get into ITS.
- If you use PDP View, in its the top window above the yellow Teletype: type ‘`telnet localhost 10018`’. Again, `CTRL-Z` to get into ITS.

For this Quick Overview, we keep it at the above possibilities. But there are many others:

- You can also run the terminals remotely, on a laptop, and log in from there. Typical commands would be `rpdp telcon`, and `rpdp tvcon` (note the r-for-remote). See the section on remote connections for setup details.
- You can also simply telnet into the Pi from anywhere, for instance `telnet pidp10 10018`. You can also get the system console Teletype from a remote machine: `telnet pidp10 1025` (this assumes your Pi is named `pidp10`, obviously, and that you have logged out of any other telnet session using the 1025 console port).
- You can use real serial terminals over a serial port.

See the section ‘ITS walkthrough: Light Entertainment on the Mainframe’ for next steps.

ITS walkthrough: Light Entertainment on the Mainframe

Introduction

This section is simply a walk-through demonstration of some fun or interesting programs on ITS to get you familiarised with the system. All we want to do is give a first flavour.

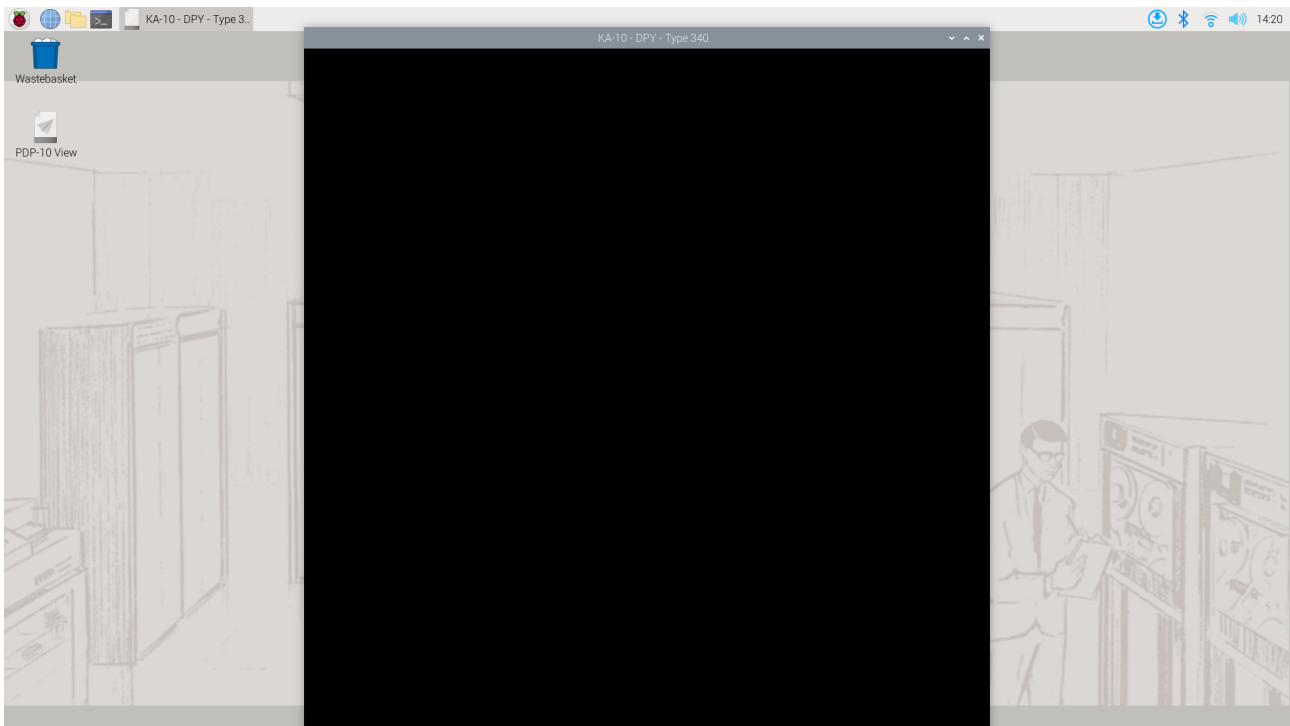
Also, we use `pdp view` on the Pi's HDMI monitor as our main terminal here. But that is not necessary – in fact, most of the time, you'd use a laptop with terminal emulator to connect to the PiDP-10. We'll demonstrate that too.

For more in-depth information:

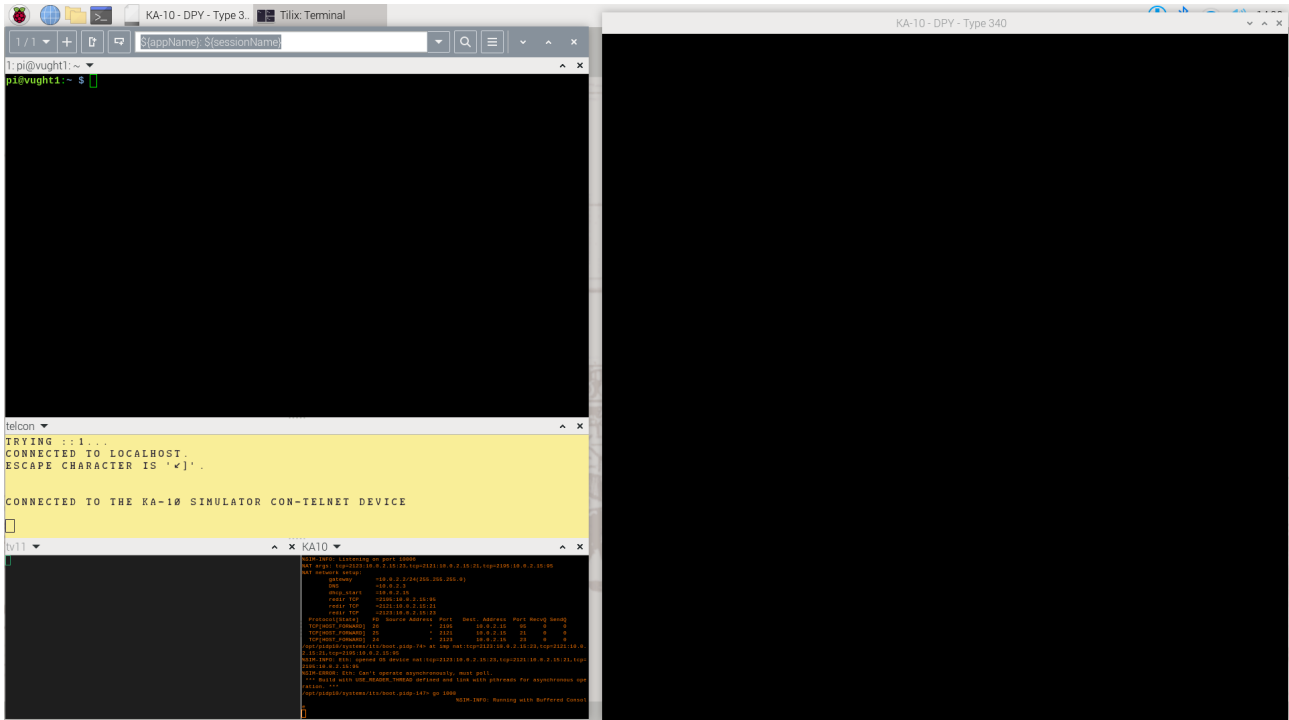
- A modern ITS manual is progressing here: <https://github.com/larsbrinkhoff/its-manual/wiki>
- Another useful reference: <https://its.victor.se/wiki/start>
- Online help inside ITS. Try `:HELP`, `:INFO`, and the documents in the `.INFO` directory.

Starting the ITS Walkthrough

Set the PiDP-10 to boot up with ITS (by setting the rightmost of the 36 data switches), and power up. You will see the front panel light up and see the Pi's desktop, with a big type340 graphics display (you can minimise its window if you want):



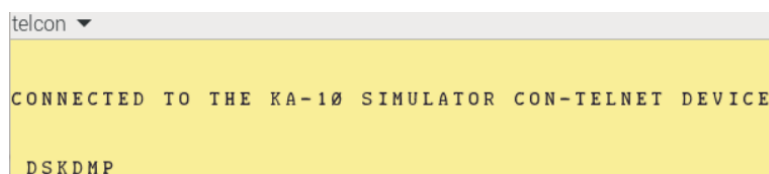
Double-click to execute the PDP-10 View icon on the desktop. This gives you an all-in-one view of the PiDP-10:



The yellow terminal is the simulated Teletype on the PDP-10's console port. We moved the type340 graphics display to the right. The PDP-10 is running on air at the moment, and to bring up the system: press the **STOP** switch, and then the **READ IN** switch. The latter triggers a PDP-10 to read DSKDMP from paper tape and starts running it.

- I. If you run on a 'Naked Pi', i.e. without the PiDP-10 front panel, no need to press any switches (as you will not have any!).
- II. Note that the PDP View window is just a handy collection of terminal windows. It's not really part of the PDP-10 simulation, you can do everything on just a regular command line as well. For instance, when you run a 'headless' PiDP-10. As mentioned earlier, `pdp` and `pdpcontrol` are the real PiDP-10 commands you need.
- III. It gets old rather quickly, but for demonstration purposes you could also have used the 'real time' Teletype simulator (real time, meaning, as slow and noisy as the original) instead of this all-in one PDP View. Instead of starting `pdp view`, on the Linux command line, type `pdp con` and that will be used as the PDP-10's console instead of PDP View. And on a headless PiDP-10 (on a ssh connection for instance), you should use `pdp telcon` instead. Or just `telnet localhost 1025`. You can switch between these Teletype options mid-flight, as long as you leave the telnet connection orderly (`Ctrl-]`, then `quit`)

You will see DSKDMP report itself on the Teletype:

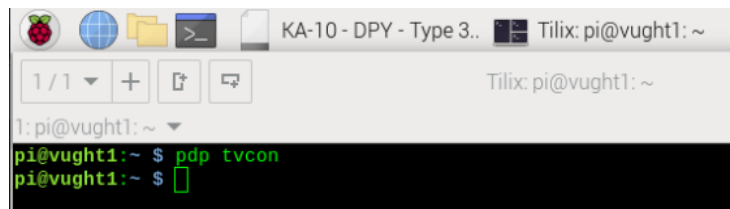


Type `ITS <esc> G` to boot into ITS, and you'll arrive at this stage:

```
telcon
KA ITS 1651 IN OPERATION AT 15:21:38
KA ITS 1651 SYSTEM JOB USING THIS CONSOLE.
TV 11 WENT DOWN -- 15:21:38
LOGIN TARAKA 0 15:21:39
TOP LEVEL INTERRUPT 200 DETACHED JOB -> 4, USR:TARAKA
201314/ 0 1 TARAKA 15:21:43
TV 11 IS UP - 15:21:43
```

You can press `Ctrl-Z` to sign in to ITS from the Teletype. From here, the Teletype is just like any other terminal on ITS, with the exception that ITS prints out regular system state updates on it as well.

But Teletypes are no fun. Instead, on the Linux command line, type `pdp tvcon`, and a Knight TV terminal pops up. We'll start off pretending you are an MIT student, for them this was the default way of getting access to the PDP-10.



```
KA-10 - DPY - Type 3.. Tilix: pi@vught1: ~
1/1 + [ ] [ ]
Tilix: pi@vught1: ~
1: pi@vught1: ~
pi@vught1: ~ $ pdp tvcon
pi@vught1: ~ $
```

The Knight TV terminals consisted of a TV tube and a rather nice custom keyboard; 16 of them were connected to a PDP-11, which acted as the terminal multiplexer, sharing memory with the PDP-10. Note that on the Teletype, you saw a message TV-11 is up, to confirm that the PDP-11 is indeed up and running. The status of the TV-11 is shown at the bottom left of PDP View.



Knight TVs were amazingly powerful terminals – not only could they do graphics, but students could view other’s displays, or even take over control from their own keyboards. Keep in mind that the things below were done in 1972, long before such features were available anywhere else.

Logging in on the Knight TV

Hit F1 to get into ITS. F1 on the Knight TV is the same as CTRL-Z on the Teletype console and on all terminals. It’s just that the Knight keyboard had a dedicated ‘CALL’ key for this.

```
KA ITS.1651. DDT.1548.
TTY 64
You're all alone, Fair share = 99%
Welcome to ITS!

For brief information, type ?
For a list of colon commands, type :? and press Enter.
For the full info system, type :INFO and Enter.

Happy hacking!
:login lars█
```

ITS does not do security; passwords and access restrictions just stand in the way of hackability. But it was polite to login. The link below helps you to set up your own account, with mail and your own work directory. But for now, you’ve stolen the lars account to use... Type

```
:login lars
```

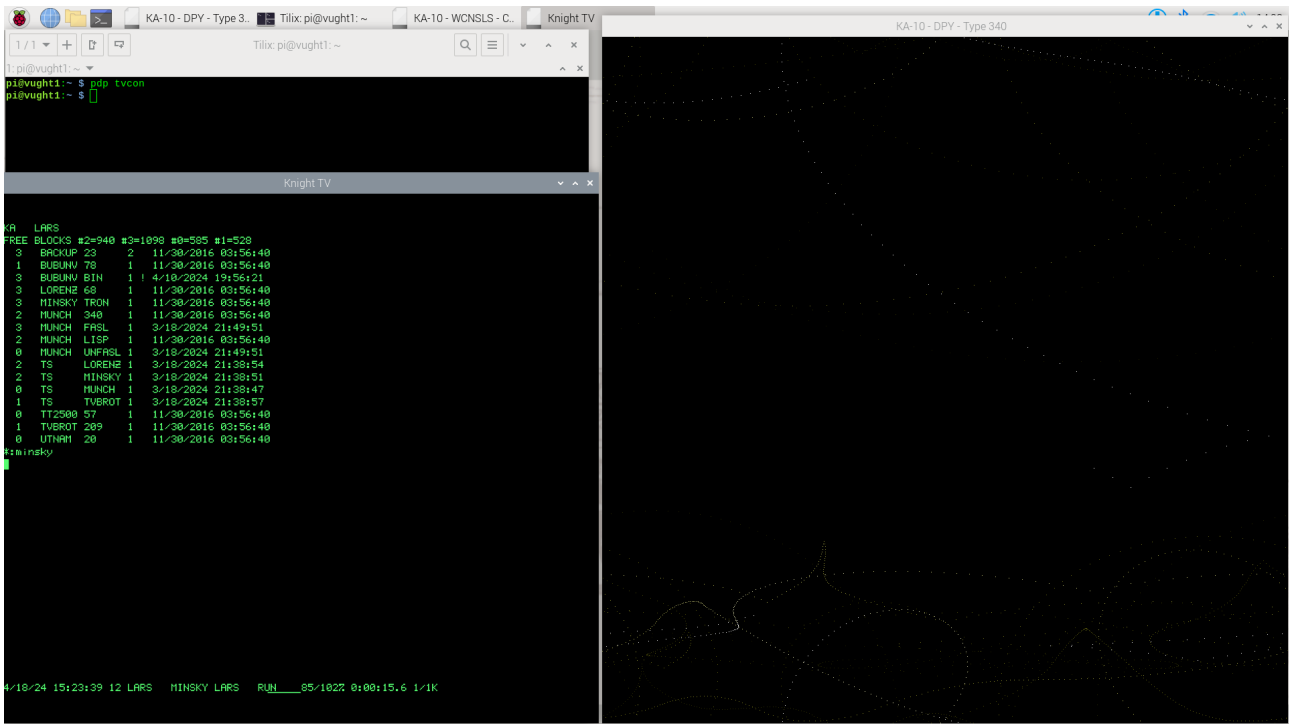
Follow the instructions on the ITS Wiki to set up your own user account, with your own directory, mail, etc.: (<https://github.com/larsbrinkhoff/its-manual/wiki/Adding-a-new-ITS-user>)

You are now in HACTRN, the ITS debugger which, interestingly, doubles as the shell. Press Ctrl-F to list the directory of this user. ITS generally has long-form commands (:login, :listf) and alt-mode shortcuts (<esc>u, Ctrl-F). Using the long-form versions is generally considered to be for Turists and Lusers... noobs.

It is also very Turisty, even Luserish, to say <esc>. The proper ITS term is **Altmode**. Altmode is Esc. And the regular Alt key on your keyboard does not play a role in ITS life. We stick to saying <esc> here, because it is just a quick walk-through. No point adding confusion.

Run the Minskytron demonstration program, by typing :minsky . Although the very fine dots are not clearly visible in the printed screenshot below, the type340 display will come alive. Play with setting the data switches to influence the Minskytron patterns. The source code is in the same directory, and this might be a nice basis for building your own display demo hack some day.

See <https://www.masswerk.at/minskytron/> for more on Minskytron. It is a deep rabbit hole to fall in to.



Abort Minskytron using `F1` (or `Ctrl-Z` on any terminal other than the Knight TV), which drops you into HACTRN, the ITS debugger/command line interpreter again. The screen shot does not show it, but it would be good form to type `:KILL` so as to end the Minskytron job, and not just leave it suspended. It is also often useful to clear the screen by pressing `Ctrl-L`.

Hacking spacewar and using tvwar

Now, change to the games directory by entering `:cwd games`

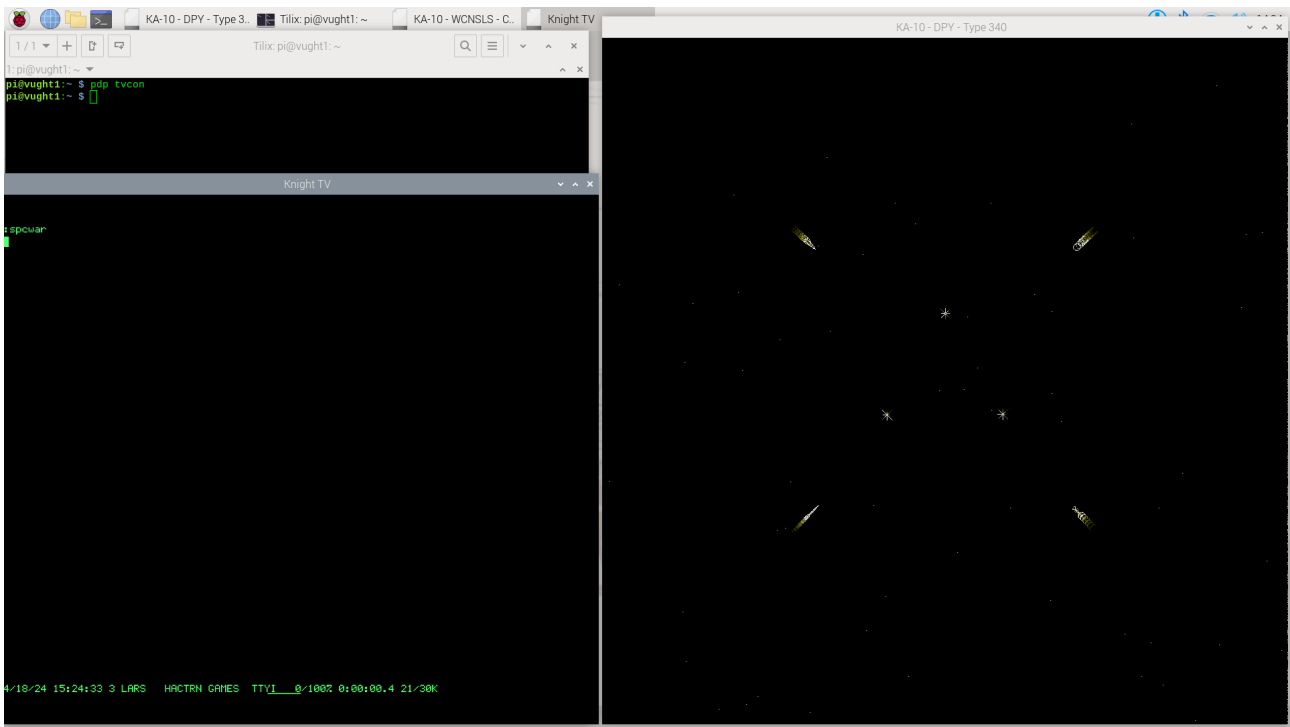
```

0 TT2500 57 1 11/30/2016 03:56:40
1 TVBROT 209 1 11/30/2016 03:56:40
0 UTHAM 20 1 11/30/2016 03:56:40
#:minsky
↑Z
243) SOJGE 1,243 :kill
#:cwd games
#

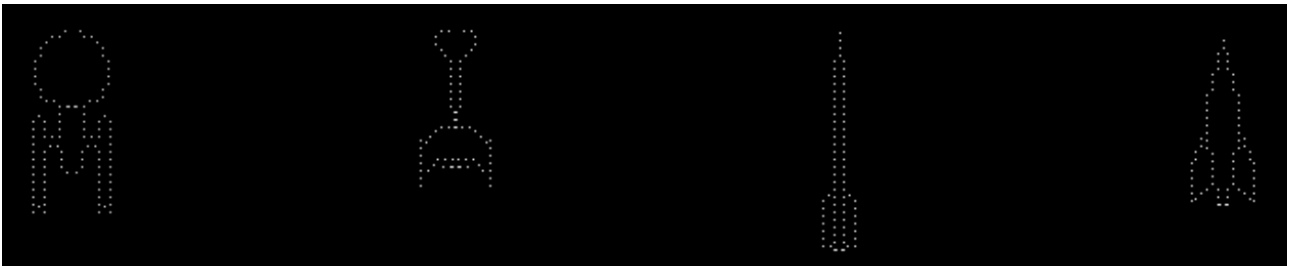
```

List the games directory by typing `Ctrl-F`, just to get an idea of what is there. Filenames in ITS consist of two 6-character fields; if the first field contains 'TS' (for timesharing), that denotes an executable which runs under ITS. One of them is TS SPCWAR.

So type `:spcwar` to start this vintage (because, first ever) video game in its PDP-10 incarnation:



You will need USB joysticks to actually play the game; but even without, you get a pretty good impression of it. By default, PDP-10 spacewar starts up in the fancy version with three suns and four spaceships:



This gives us a nice opportunity to show a bit more of HACTRN, the debugger that is also the command line. Press F1 to interrupt the program, and then:

- Type `<esc>Ctrl-K` to load its symbol table(!)
- Type `:lists` to print out all the symbols
- Type `nsuns/=1` to inspect the 'number of suns' symbol and change it to 1
- Type `nships/=2` to change to a two-player games
- `<esc>P` proceeds with the program execution

Wait a bit, and once the current players perish and a new game commences, you'll see the single sun with two spaceships as it was in the earlier PDP-1 version.

From which we draw the general conclusion that hackers do not really need a Settings menu if the a debugger is the command line.

Hit F1 to halt Spacewar, and `:kill` to remove its job.



The problem at the AI Lab, of course, was that there was only one type340 display, and it stood next to the PDP-10. Not next to a student's Knight TV terminal. Therefore, a TV War game was also written. Type `:tvwar` to start it.



Player 1 controls his spaceship with Z,X,C for rotation, V for acceleration, and `<space>` to fire torpedoes. Player 2 uses [,],\, P and ;. But note: use capital letters.

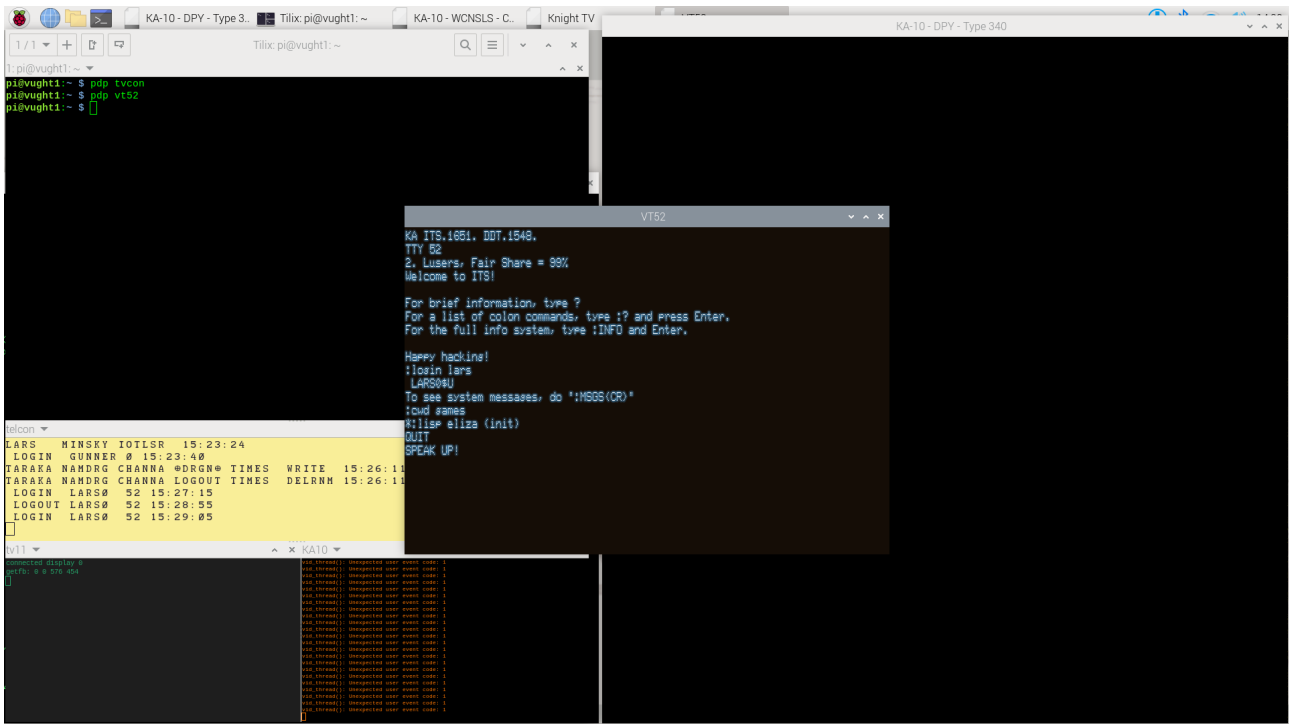
There's also Dazzle Dart, another very early video game to play. Alas, you really do need 4 players, each with a USB joystick. `:games;dazdrt`

Adding a VT-52 terminal session, some Lisp therapy

Not everyone used a Knight TV terminal. Regular serial terminals were around as well (the Datapoint 3300, `pdp dp3300`, for instance). But users would log in from around the world over ARPANET.

Let's use an industry-standard DEC VT-52 terminal. Type `pdp vt52` on the linux command line, at the top of PDP View, and let us use that to run Eliza, the very early AI therapist written in Lisp.

Press `F11` to make the VT-52 full-screen (you can do that on the Knight TV terminals too).



In the VT-52, press `Ctrl-Z`, type `:login lars` to log in, `:cwd games` and then `:lisp eliza (init)`. Now tell Eliza what is bothering you, and end your sorrows with a double return. You need a bit of experimentation in how you formulate your issues, but many have profited from Eliza once they did.

Press `Ctrl-Z`, and type `:kill` to leave Eliza behind. `Ctrl-L` to clear the screen perhaps, then `:cwd lars` to get back to Lars's home directory again. Then, `:emacs munch lisp` will let you edit a simple ASCII-art version of munching squares. Emacs, of course, was born on ITS and thus is the preferred editor. You might want to try Teco, the other editor, to find out why.

Google on emacs keystrokes, but for now, `CTRL-V` does a page down – where you see that munching-squares will start the program. `Ctrl-X Ctrl-C` brings you back to HACTRN.

Then, do

- `l` `Ctrl-K` (lower-case l, this is the *altmode equivalent* of Turistish `:lisp`)
- answer `n` to the Alloc? Question
- `(load "munch lisp")`
- `(munching-squares)`

This version of munching squares left an audience of ITS veterans distinctly unimpressed, when we did this demonstration recently. Which was fair enough, it was just to show lisp and the VT-52. So, press `Ctrl-Z`, `:kill`, `:munch` and watch a much more beautiful munching squares on the type340. `Ctrl-Z`, then `:kill` once you have enough. Here, too, you'll find the source code in the same directory if you want to play with it³.

³There is much more to Munching Squares than you'd think. This is where we introduce **HAKMEM**, which gets you started: <http://www.hakmem.org/#item146>. HAKMEM, if we don't mention it anywhere in this manual, is the collective notebook of the ITS hackers. Dive in, it is where the PiDP-10 wants to lead you ;-)

To graciously leave ITS from this terminal, type `:logout` . Then, to virtually walk away from your VT-52, type `Ctrl-]`, then `quit`. Just to demonstrate that we used a telnet connection to hook up the VT-52 with ITS. Close the VT-52 window.

Multiple jobs

Select the Knight TV terminal window again, and perhaps press `Ctrl-L` to clear the screen.

Type `:peek` to run the ITS system monitor utility, the equivalent of `top` in Linux. Press `?` to get an overview of its display pages. But more interestingly, press `Ctrl-Y` to let Peek use the `type340`; and then type `p` to proceed- meaning, let Peek run on the `type340` but proceed with other jobs on the Knight TV terminal itself.

```

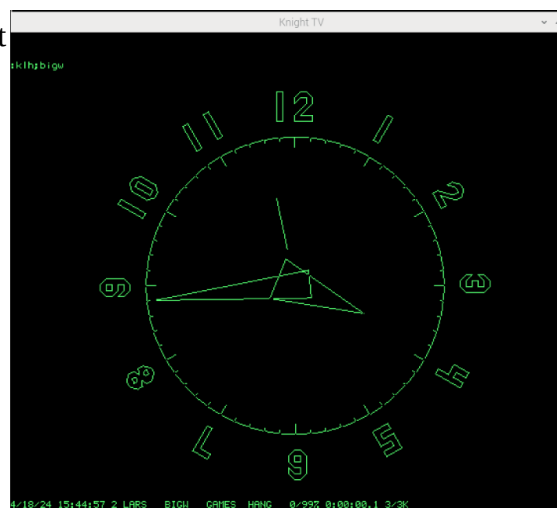
KA ITS 1651 Peek 631 4/18/2024 15:41:04 Up time = 19:04
Memory: Free=442 Runnable Total=11 Out=2 Users: High=14 Runnable=1
Index Username Jobname Sname Status TTY Core Out ZTime Time Pts
0 SYS SYS SYS HANG ? 71 0 0% 1
1 CORE JOB CORE UUO ? 0 0 0%
2 LARS HACTRN GAMES TTY1 T64 30 9 0%
3 LARS PEEK GAMES +SLEEP < D 11 2 0%
4 TARAKA CHAURL CHAURL 1010 ? DSN 29 0 0% .VALUE
5 .BATCH BATCHN .BATCH SLEEP ? 126 23 0%
6 TARAKA NARDRG NARDRG HANG ? 29 0 0%
7 TARAKA PAPSBU PAPSBU HANG ? 1 0 0%
8 TARAKA JOB_07 SYS HANG ? 3 0 0%
9 PPTHNG DRAGON DRAGON HANG ? 6 0 0%
10 GUNNER GUNNER GUNNER _SLEEP ? 12 3 0%
Fair Share 100% Totals: 318 0% 3
Logout time = 41 Lost 0% Idle 96% Null time = 18:26
HYPFRACED
HKLH;TINYW
pp
H:ListJ
H:PEEK R 2
H:TINYW R 12
H:job
H:PEEKJJ
H:ListJ
H:PEEK R 2
H:TINYW R 12
4/18/24 15:41:11 3 LARS HACTRN GAMES TTY1 0/100% 0:00:00.5 21/30K
  
```

To demonstrate the multi-tasking from here, type `:klh;tinyw`

and note how a small analog clock appears on the Knight TV. Press `Ctrl-F` to see that TinyWatch keeps running in the background as well, whilst you continue in HACTRN.

Enter `:emacs test` to just start up one more job, and type `Ctrl-X Ctrl-C` to leave it in the background, too.

You now have three jobs, and you can see them with `:listj`



Note that only one of the jobs has a * in front of it. This is considered the current job, you can rotate through the list of jobs with `:job` and then use `:contin` to continue it. Or just type `:contin emacs`.

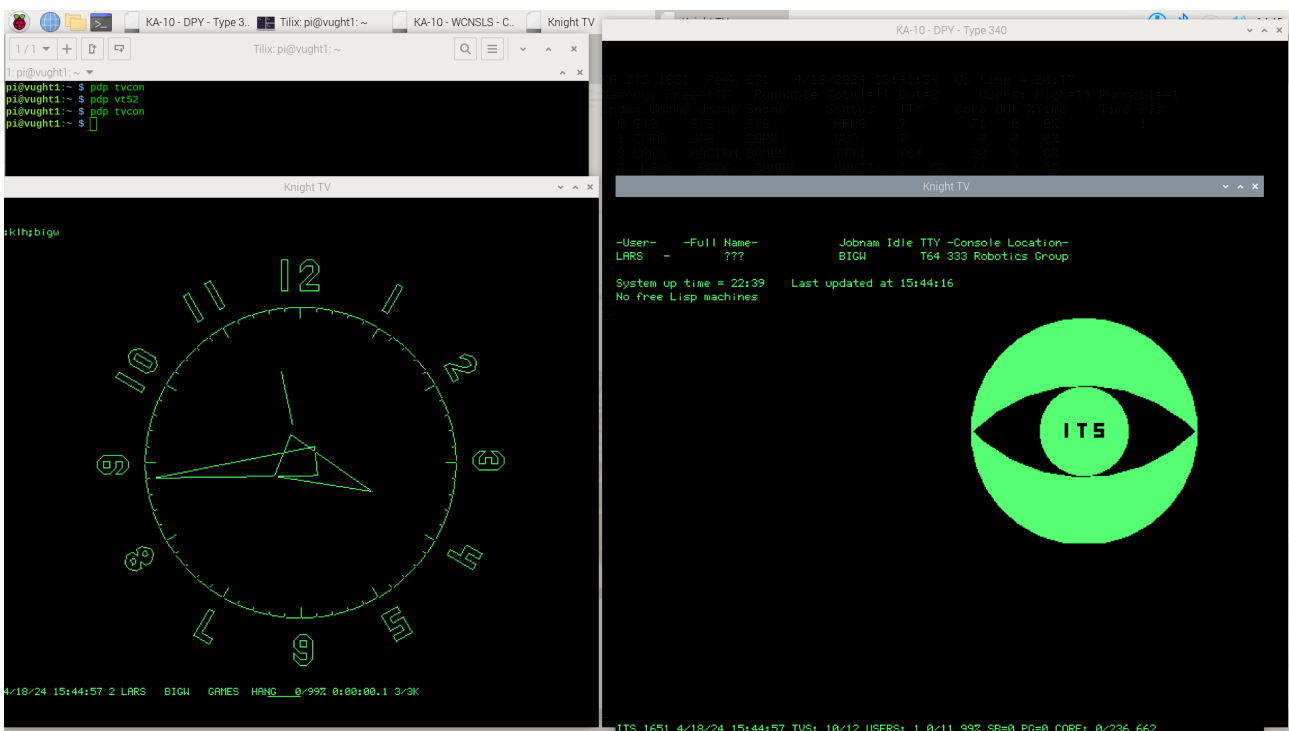
Of course, you should look up the altmode keystrokes at some point to migrate from Luser (or perhaps, Turist) to hacker status.

Clean up all the jobs with a repeated `:kill` command, and to leave this Knight TV terminal in a decorative state, do `:klh;bigw`.

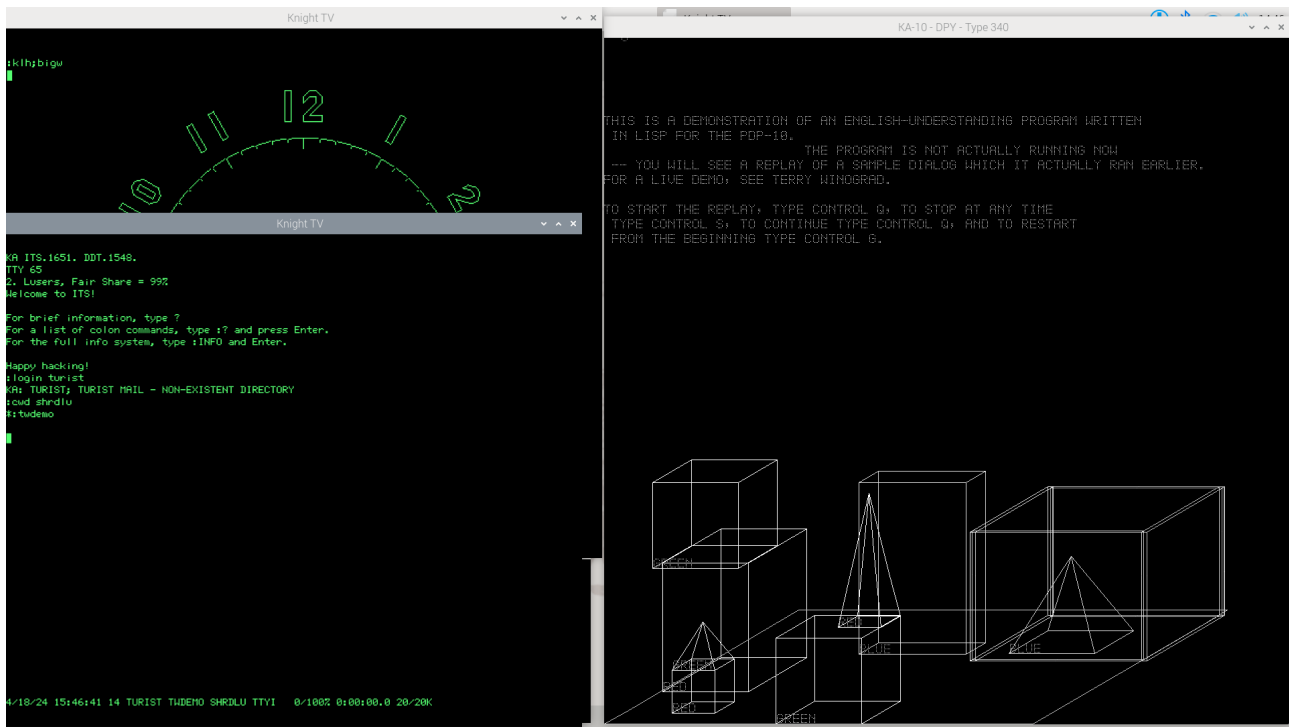
The Ultimate 1970s AI demo: SHRDLU

It is very much worth googling for more background information on Shrdlu – this AI, living in a simple 3D world of a table full of blocks, boxes and pyramids, has a ‘real’ understanding of its world. And its English language parser makes it accept surprisingly complex commands, and respond in a manner that shows actual insight into its little world. Keep in mind: this was in 1969...

Start up a second Knight TV terminal (by entering `pdp tvcon` on the Linux console):



Again, log in to ITS (F1, `:login tourist`) and go to the Shrdlu directory (`:cwd shrdlu`). Then, run the demo with `:twdemo` and start it off by typing `Ctrl-Q` in the Knight TV terminal window. Move the window aside so you have the type 340 window in full view again:



At some point, use F1 and `:kill` to terminate the demo. Sadly, the original *interactive* Shrdlu has not been recovered. We have only the scripted demo. Some Googling will tell you the story, and why it is not likely the unscripted version will ever be brought back...

Remote Knight TV sessions from your laptop

So far, we have demonstrated simulated terminals on the Pi's HDMI monitor. But it will often be more convenient to run a terminal from your laptop, seated in a comfortable recliner chair. The PDP-10, after all, is a mainframe-class computer; terminals can be anywhere.

Getting the terminal simulator programs on your Linux laptop/desktop: see the section 'Connecting with remote terminals to the PiDP-10' for setup details. This will work on Windows 11 as well, if you install the WSL Linux subsystem.

Connecting to the PiDP-10: After you've installed the `rpdp` package from github, just use `rpdp tvcon` for a Knight TV on your laptop. Pressing `F11` will make the terminal simulators use full-screen mode. `rpdp` has the same functionality as `pdp` on the Pi itself. So you have `rpdp telcon`, `rpdp vt52`, `rpdp imlac`, etc.



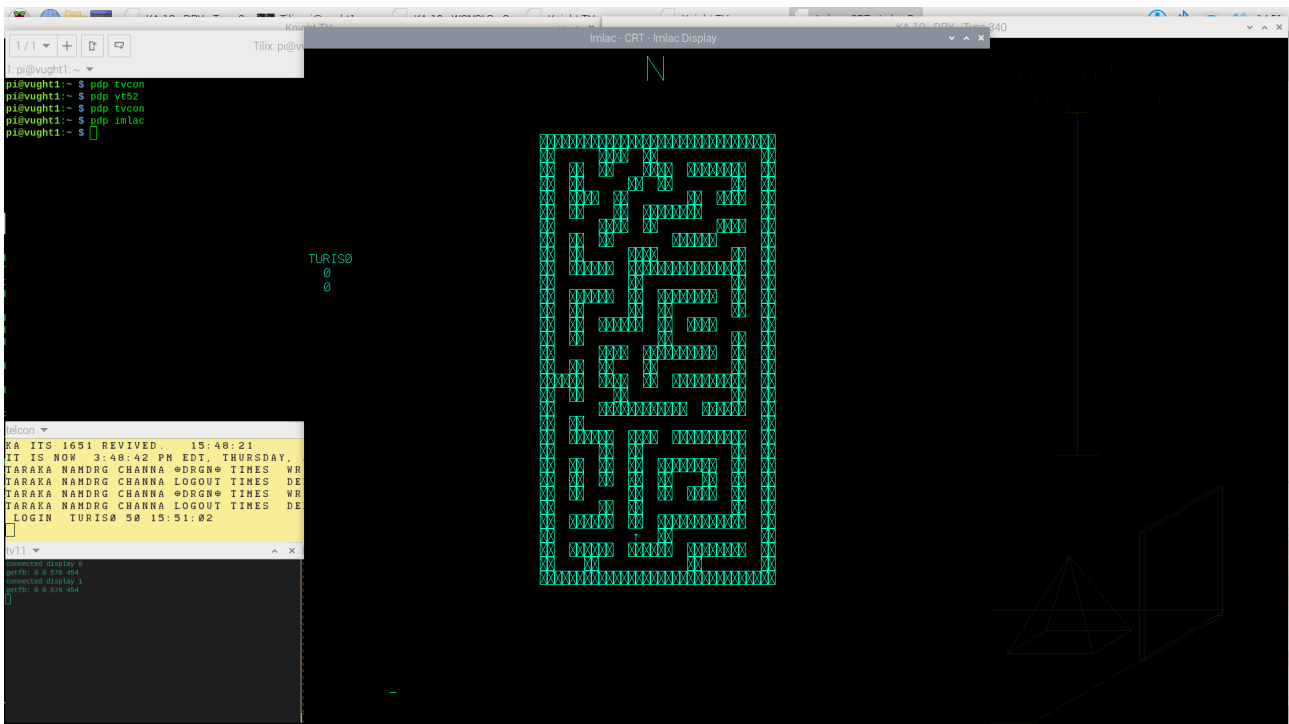
Above: a Knight TV'd laptop running MacHack VI chess on the PiDP-10 (Don't forget to press `F1` to get into ITS, then run `:games; ocm`, and enter `ps` to let it play against itself). If you prefer, another chess version will use the type340 to display the board. Run `:games; c`, and type `FANCYTAB2<enter>`. Press `F1`, do `:kill`, then `:logout` and you can close the Knight TV terminal window. You can run multiple Knight TV terminals on the same laptop.

Multi-Player Maze War

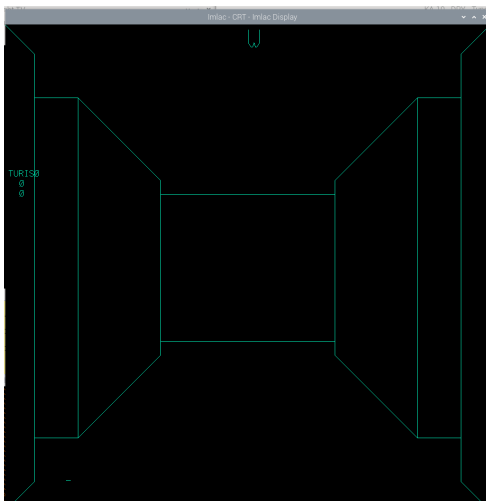
Maze War might be the first 3D FPS game. Assuming you have prepared the Imlac simulator on a second Linux machine as per the above section, here are the steps.

On the PiDP-10 itself, start one Imlac simulator by running `pdp imlac` on the Linux command line. And on the Linux laptop/desktop, start another Imlac with `./imlac ./imlac.simh`, and make sure you have edited the `imlac.simh` config file to replace the reference to `localhost` to `pidp10.local` – or whatever you chose for your PiDP-10's network name.

Type `Ctrl-Z`, and `:login tourist` (on the laptop) and `:login lars` (on the PiDP-10 imlac window) to get into ITS. To tell ITS that you are on an Imlac, type `:tctyp oimlac`, and on both machines, go to the games directory with `:cwd games`, and run `:maze c` (the C indicates you want Cheat mode).



The `Tab` key will show you the maze with your position in it, the cursor keys let you move, and `<esc>` or `<space>` lets you shoot at your opponent when you encounter him.



Once you are done, it is generally polite to clean up before you leave your Imlac terminal:

```
Ctrl-Z  
:kill  
:logout
```

– and then you can just close the Imlac simulator.

Gosper's Game of Life: the original

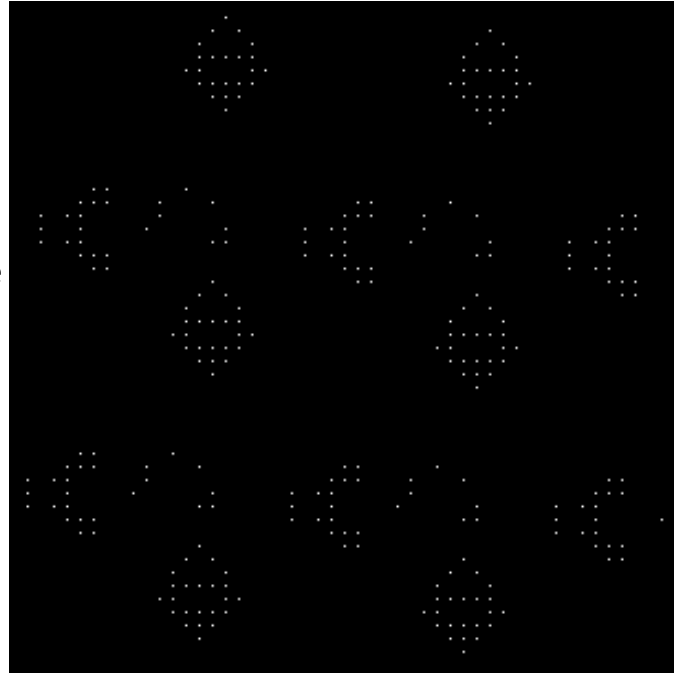
If you know your computer history, Life will need no introduction. The author of the PDP-10 Game of Life was Mike Speciner; and Bill Gosper (RWG)'s famous experimentation files were also recovered. This is where the Glider and other interesting patterns were first discovered. Worth Googling.

```
:games;mlife (to run it, the type340 will be  
the display, your current Knight TV the console)
```

Then, type (in the Knight TV window, where Mlife shows an underscore waiting for input):
I RWG;LIFE DEMO (caveat: you have to type uppercase input).

Hit (upper-case) R repeatedly to go through all the patterns in Gospers saved workbook, and hit space repeatedly to let the life evolution step through generations.

To see the source code, do
:emacs rwg;mlife



And, as we get to the end of the walkthrough, a side step: it can be useful to see a bit more of the PiDP-10 simulator setup. If you ran the install script to also download the ITS source code (you can rerun the script to add the sources if you want), look in /opt/pidp10/src/its/src/rwg . This is the input for the reconstructed ITS disk image, but it is also handy to know all files are available for review and just spelunking around in Linux.

Digging deeper into ITS

<2024-05-02: More to come>

Shutting down gracefully

You would normally type this on the Teletype console, but it works on any terminal. If the Teletype does not respond, type a `Ctrl-Z` to call ITS again. Then, to bring down ITS in an orderly manner, type:

```
:lock  
5down
```

y
Ctrl-C

If there are no users still logged in, the shutdown will commence immediately. If there are any users still logged in, or some of their jobs still running, you'd have to wait 5 minutes. So it makes sense to :kill and :logout properly.

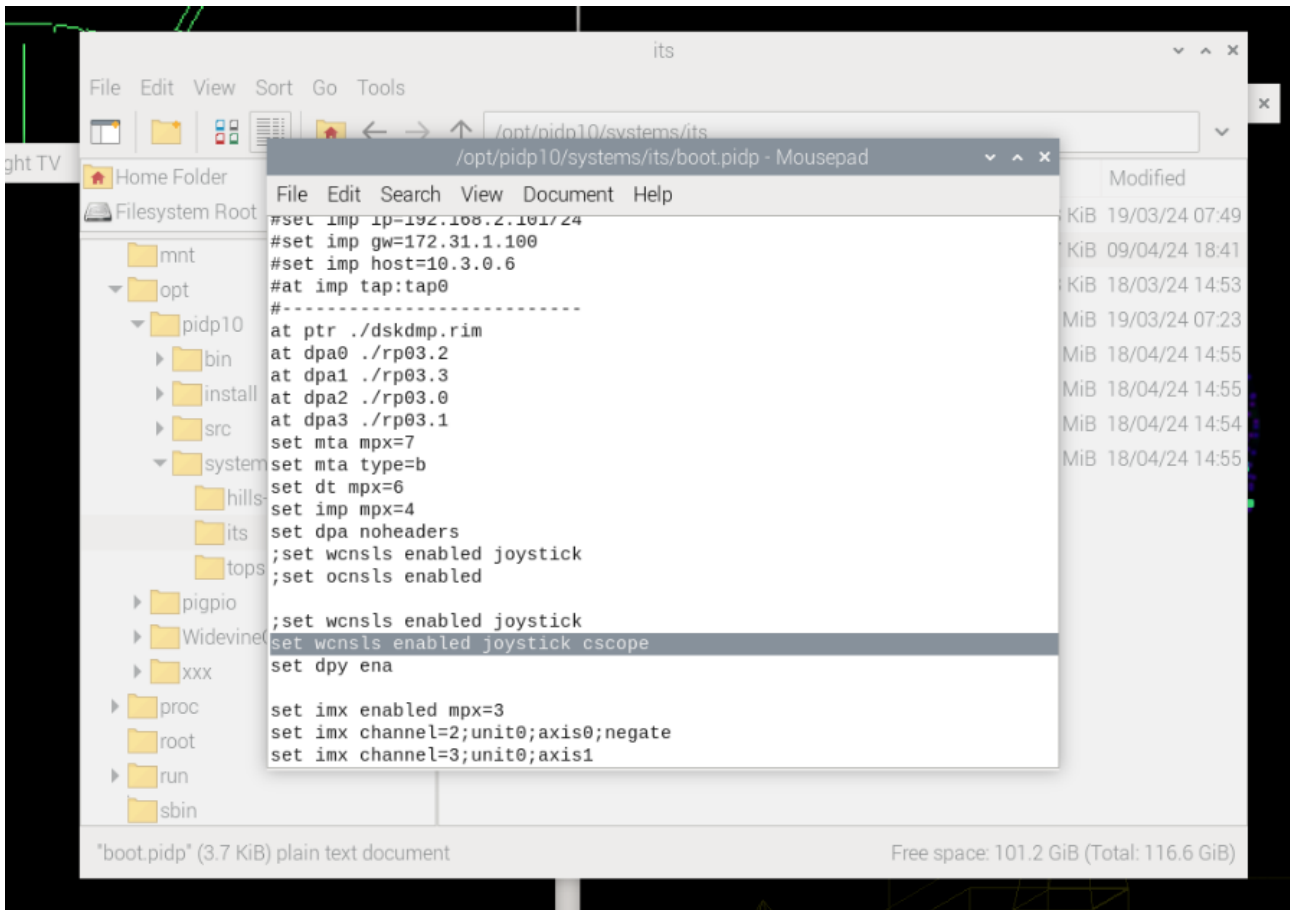
Generally, nothing bad occurs after a disorderly shutdown (if you do not wait for the five minutes). But trouble could occur. If it does, run the PiDP-10 install script to copy back a fresh set of ITS disk images.

One more thing... democoding!

Bubble Universe - Democoding on the Color Scope

DEC delivered a color scope – although smaller, effectively a color version of the type340 – to the AI Lab. Only two of these were ever made, but it is a nice target for writing graphics demos. At the time of writing this manual, we’re not sure whether we should enable the ‘Cscope’ display window at startup or not. So if you do not see a smaller-sized ‘color scope’ window, you can enable it by editing the ITS boot script file. Which, in fact, is useful to know about anyway.

The boot scripts for PiDP-10s are named `boot.pidp`, although if you run a Naked Pi (no PiDP-10 hardware attached), the `boot.pi` scripts are used instead. So if you do not see the Cscope window just yet, simply edit `/opt/pidp10/systems/its/boot.pidp` (or `.pi`) to enable the Cscope window by uncommenting the highlighted line with a semicolon. You’ll then want to comment out the line directly above. You’ll have to reboot for changes to take effect; and the Cscope window will only appear once ITS is booted up.



And note that similarly, the line below the highlighted one causes the type340 to appear or not. There are many more customisation options in the boot file, which is why we like to show it in this ‘walkthrough’.

Good – assuming you have the Cscope window on your display, there is a newly-written piece of democode to highlight the possibilities. Lars Brinkhoff wrote it in PDP-10 assembler, in late 2023,

based off a thread here: <https://retrocomputingforum.com/t/the-marvellous-bubble-universe-graphical-animation/3651>

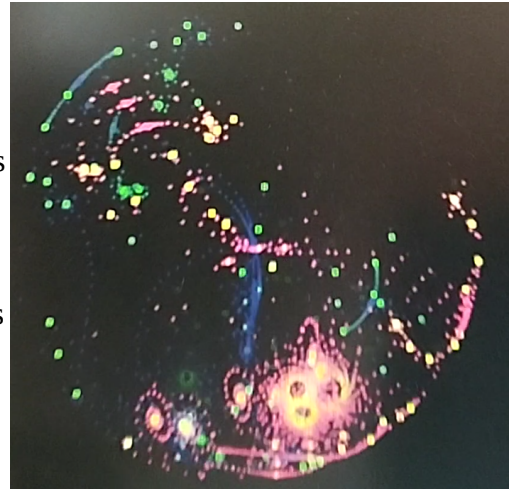
Go back to one of the Knight TV terminals that you still have open. Type `F1` and `:kill` in case anything is still running on it, and do `:cwd lars` to go to his directory.

Then:

```
:emacs bubunv > (to open the latest version – ITS does versioning! - of the source code)
```

Hit `Ctrl-V` to page through the 6 screens of PDP-10 assembly, `Ctrl-X Ctrl-C` to leave emacs. Use the Midas assembler to assemble the source into a binary:

```
:midas bubunv (bubunv without extension will let ITS pick the latest source version)
```



Hit `Ctrl-F`, and you will now see a BUNUNV BIN binary. Create a new job, load the binary in it and let it run:

- `bubunv<esc>j`
- `<esc>l bubunv bin`
- `<esc>g`

And again, stop with `F1` and `:kill`.

Front panel operations – a primer



This is just a summary to get you started.

For a thorough description, DEC's manuals are the best source: page 102-109 (2-84 – 2-90).

http://www.bitsavers.org/pdf/dec/pdp10/1970_PDP-10_Ref/1970PDP10Ref_Part1.pdf ,

Indicator lights

Top row: Program Counter – 18 bits. Shows the address of the currently executed instruction when running; when manually operated, shows the currently selected address the front panel is set work on.

Middle row: Instruction Indicator – 36 bits. A very crude approximation to give a first idea:

- **IR: left 18 bits.** Shows the instruction being executed, or just having executed. The 36 bit instruction is neatly broken up into separate bit fields (see next chapter), thus, you can see here (from left to right), 9 bits that indicate the instruction, 3 bits to indicate which of the 16 registers/accumulators this instruction works on, 1 bit to indicate whether the address after it is an indirect address reference, 3 bits to indicate which register is used as an index
- **MA: right 18 bits,** holds the address this instruction is working on.

Bottom row: Memory indicator. Shows the 36-bit value in the memory address being referenced. But 'referenced' can mean different things, for which we refer to the manual – or watch Lars Brinkhoff's Youtube movie.

Switches

Bottom row: Data switches. If front panels are new to you, these can be used to toggle in the binary value to store into some memory location. Also, programs can read the switch register to see which switches a user sets

Top right: Address switches. The 18 switches on the top right are use to form an address. Either an address you want to deposit the data switch contents in to, or the address of a word in memory you would like to examine. Or the address you want to set a break point on!

Top left: Operating switches: the 20 switches on the top left. Strictly speaking, there are 10 operating switches (toggle switches) and 10 operating keys (momentary switches). We will go into them now.

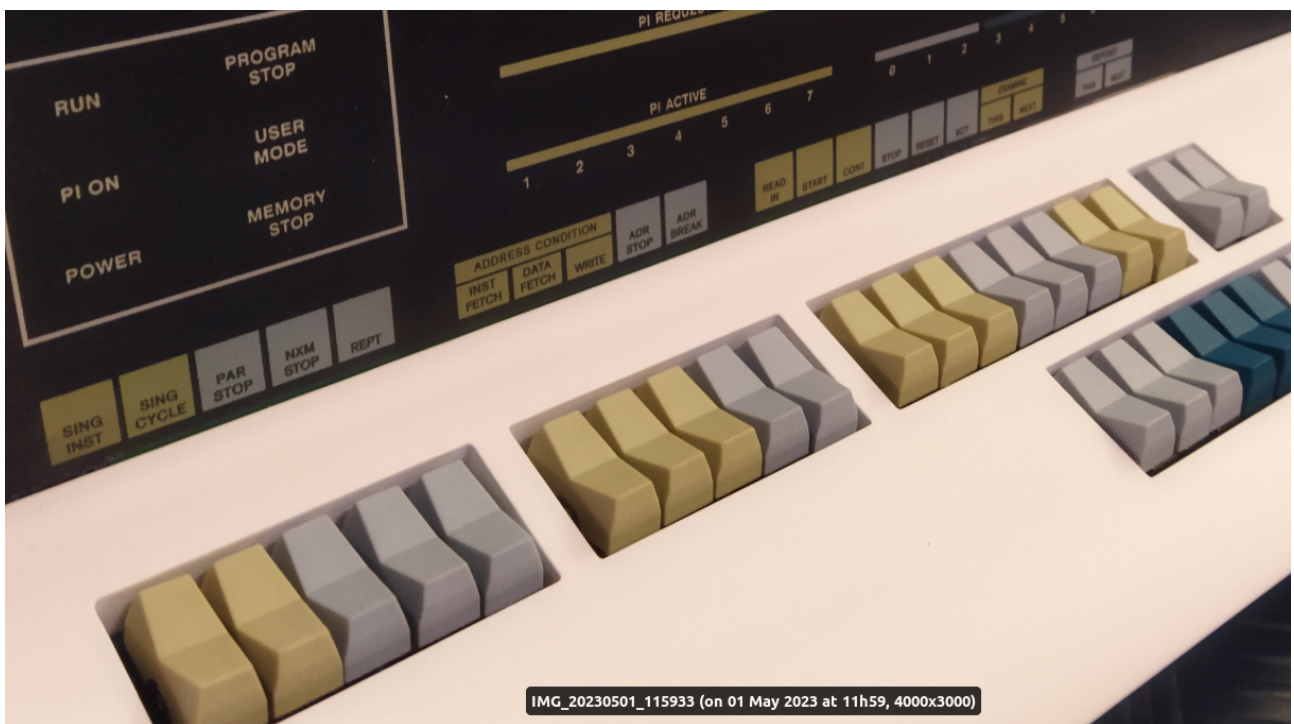
Preparation

Playing with the front panel when Hill's Blinky is running is nice and harmless. Doing it when running an actual operating system might prove destructive if you don't yet know your way around.

So (re)boot the PDP-10 into Blinky, by not setting any of the data switches. You can restart the Raspberry Pi of course, but it's quicker to enter `pdpcontrol stop`, followed by `pdpcontrol start` after a few seconds. And if you want to leave whatever is on the data switches unchanged, just do `pdpcontrol start 0` to let the PiDP-10 ignore the actual value on the data switches, and boot Hill's Blinky regardless.

Using the front panel

If you are used to the front panels of simpler computers, you'll be in for a pleasant surprise with the PDP-10's. It contains some advanced debugging features. Below is an extract of the relevant section of the PiDP-10 wiki (<https://github.com/obsolescence/pidp10/wiki/PiDP%E2%80%90testing>) to give an overview.



The nice thing is that this overview also is a first introduction of the `simh` command line. It is a very useful tool for writing and debugging short PDP-10 programs, as you'll see during the explanation of the front panel.

Lars Brinkhoff also made a very effective Youtube movie about the PDP-10 front panel's operation. See it [here \(link\)](#). It is a quick way to demistify the front panel's operation.

Depositing and examining data into memory

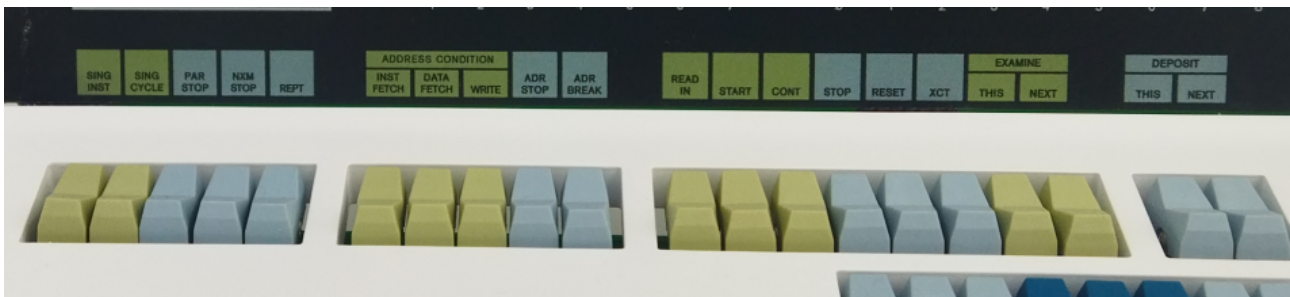
Obviously, a simple use of the front panel is to fill memory with data (for instance a small program). Set a value on the data switches, set an address on the address switches. Press **STOP** to halt the machine, press **EXAMINE THIS** to see what is currently stored in the selected address, and press **DEPOSIT THIS** to replace that value with the value set on the data switches. Press **EXAMINE THIS** again to see that, indeed, the value from the data switches is stored in memory.

If you enter a series of words into memory (a little program for instance), it would be annoying to have to set the address switches for every new word. Therefore, you can use **EXAMINE NEXT** and **DEPOSIT NEXT** to just walk through memory word-by-word, either to look at what is in memory, or to enter new values from the data switches.

See this page on the PiDP-10 wiki to practice a bit with Examine & Deposit: ([link](#))

Program Control

After inspecting and changing data in memory, the front panel's practical use is of course on running and stopping programs.



The section (please follow this [link](#)) of the Wiki contains a practical exercise: how to enter a little program in the simh command line, how to start, stop, restart or continue the running program's execution; and how to single-step through its instructions as it runs.

Unsurprisingly, this is done with the **START**, **STOP**, **CONT** switches. Setting **SING INST** will single step, every time you press **CONT** it will only execute the next instruction and stop again. But if you set **REPT**, and keep depressing **CONT**, it will just slowly step through the instructions so you can follow along.

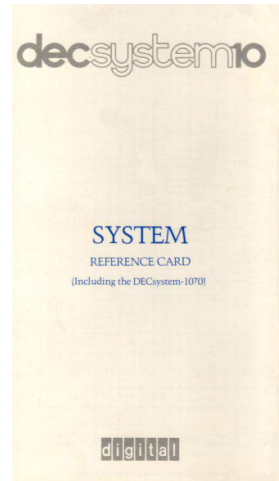
The **XCT** switch has a special function: it executes the data on the data switches straight away as an instruction, without needing any memory address to store the instruction in.

Debugging with the front panel: breakpoints

<forthcoming, or see DEC's documentation for now>

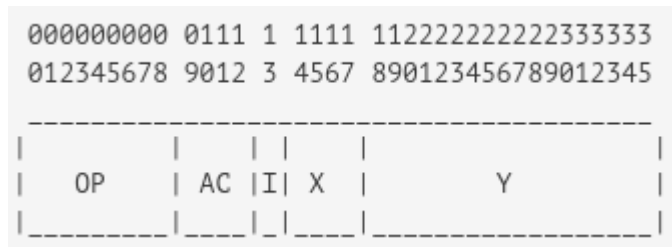
PDP-10 Machine Language

After playing with the front panel, it is a natural moment to get a first flavour for PDP-10 machine code. The first step is to print out the famous Reference Card for the PDP-10. No shirt pocket should be without it.



As the previous section explained, the middle row of front panel lights give you a view of instructions in binary. We can now start to understand them. To quote from HAKMEM:

The PDP-10 is a word oriented machine. Words contain 36 data bits, numbered (left to right) 0 to 35. Every machine instruction is one word. [...] Most instructions have the format:



- OP = operation code
- AC = accumulator field
- I = indirect bit
- X = index field
- Y = address field
- DEV = device code
- IOP = input-output operation code

The last page of the Reference Card, now permanently located in your shirt pocket, shows you the OP codes (we only print part of the table here) in octal, which translates easily into the 9 bit OP field above. To remind you of octal digits:

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

INSTRUCTION CODES								
	--0	--1	--2	--3	--4	--5	--6	--7
00-	(ILLEGAL)							
01-	USER DEFINED UOO'S							
02-	(UNIMPLEMENTED USER OPERATIONS)							
03-								
04-	CALL	INIT	LEFT FOR SPECIAL MONITORS				CALLI	
05-	OPEN	TTCALL	RESERVED FOR DEC				IN	OUT
06-	SETSTS	STATO	GETSTS	STATZ	INBUF	OUTBUF	INPUT	OUTPUT
07-	CLOSE	RELEAS	MTAPE	UGETF	USE-TI	USE-TO	LOOKUP	ENTER
10-	UJEN							
11-	DFAD	DFSB	DFMP	DFDV				
12-	DMOVE	DMOVN	FIX		DMOVE/M	DMOVNM	FIXR	FLTR
13-	UFA	DFN	FSC	IBP	ILDB	LDB	IDPB	DPB
14-	FAD	-L	-M	-B	FADR	-I	-M	-B
15-	FSB	-L	-M	-B	FSBR	-I	-M	-B
16-	FMP	-L	-M	-B	FMPR	-I	-M	-B
17-	FDV	-L	-M	-B	FDVR	-I	-M	-B
20-	MOVE	-I	-M	-S	MOVS	-I	-M	-S
21-	MOVN	-I	-M	-S	MOVNM	-I	-M	-S
22-	IMUL	-I	-M	-B	MUL	-I	-M	-B
23-	IDIV	-I	-M	-B	DIV	-I	-M	-B
24-	ASH	ROT	LSH	JFFO	ASHC	ROTC	LSHC	
25-	EXCH	RT	ACRIP	ACRIN	IPST	IECI	YCT	MAP

This gives you some first basics. Read further on <http://www.hakmem.org/pdp-10.html#IFormat> . Then, for a primer on the PDP-10 instruction set, there is great (not just good) documentation to be found in the Books section of the PiDP-10 web site. Down in the right column of <https://pidp.net/pidp10> , and many more manuals on Bitsavers: <https://bitsavers.org/pdf/dec/pdp10/> .

Machine code on the simh command line

Another, very user-friendly way to just play with PDP-10 machine code, is to go into the simh simulator and work on its command line. As long as you've booted up Blinky, you can do no harm. And the combination of the physical front panel plus the simh command line is very nice to play with. So:

In PDP View: click on the ^ icon of the KA10 window to maximise it, and then type `Ctrl-Shift-+` a few times to increase the font size. When not using PDP View, you can type `pdp` on the linux command line.



You're now in simh. Press `Ctrl-E` to stop the simulation.


Some pointers of useful commands:

<code>e 200-210</code>	octal data dump of memory addresses 200-210
<code>e -m 200-210</code>	disassembled printout
<code>d 202 move 0, 475</code>	quick assembly entry
<code>d 202 200000000475</code>	entry in octal
<code>e -2 202</code>	dump binary values
<code>e sw</code>	shows the value of the 36-bit switch register (bottom row of panel)
<code>e pc</code>	shows the program counter

But there are many, many more. Simh is the debugger and monitor of your dreams in fact. You can trace, set breakpoints, view CPU register history over the last few steps – many things that make PDP-10 assembly easier. You can also copy and paste series of commands, or load them in from a text file. `/opt/pidp10/systems/hills-blinky/boot.pidp` is a nice example.

Type `help` for simh's help pages; and there are two manuals for it. The PDP-10 specific at https://github.com/rcornwell/sims/blob/master/doc/ka10_doc.doc, and the simh command line manual at https://github.com/open-simh/simh/blob/master/doc/simh_doc.doc.

Simh is wrapped in the virtual terminal utility 'screen'. Which adds comforts and frustrations, one tip you will need is that `Ctrl-A [` lets you scroll up in recent output, `<esc>` brings you back to the simh command line. Googling for a primer on screen might be good.

Other than that, to close this first flavour on PDP-10 assembly and machine language: Type `continue` on the simh command line to resume the PDP-10 simulation, and type `Ctrl-A d` to leave the simh view and let it run in the background again (or press the  icon in the window bar when using PDP View).

Assembly in ITS: MIDAS and DDT

<forthcoming>

A first look at TOPS-10

Tops-10 was DEC's own operating system for the PDP-10. And as such, used much more widely than ITS. You might argue, it is less fun for people new to the PDP-10 in the 21st century.

Less fun, yes, but hugely influential. It defined what an operating should be for an interactive computer. And as such, everything from OS/8 to CP/M to MS-DOS builds on the model of TOPS-10. But TOPS-10 was seriously multi-user, multi-tasking, and quite a bit more powerful than much of the operating systems it inspired.

We will suffice with a brief walk-through to get you a first look. Having the TOPS-10 Operating System Commands Manual at hand will be useful: <http://bitsavers.trailing-edge.com/scandocs.trailing-edge.com/tops10-aa-0916d-tb.pdf> .



The full documentation library for TOPS-10 is here: <https://bitsavers.org/pdf/dec/pdp10/TOPS10/>
Another reference is Quentin's TOPS-10 FAQ: <https://www.quentin.org.uk/tops-10-faq/#qaef-172>

2024-04-27: the new set of disk images for TOPS-10 from Richard Cornwell was added to the PiDP-10. Rerun the PiDP-10 install script and just say Yes to the item 'download TOPS-10 disk images' to bring your PiDP-10 up to date.

Starting up

Set the data switches to 002 octal and boot up. Or, do `pdpcontrol stop`, wait a few seconds, do `pdpcontrol start 2`. This will bring up the TOPS-10 boot configuration.

Later on, inspect the boot script at `/opt/pidp10/systems/tops603/boot.pidp`, it will help to demystify things and get you more comfortable with the simh setup.

If you use PDP View, you will see this (you might have to close and re-open PDP View):

```
telcon ▼  
CONNECTED TO LOCALHOST .  
ESCAPE CHARACTER IS '^]' .  
  
CONNECTED TO THE KA-10 SIMULATOR CON-TELNET DEVICE
```

And if you do *not* use PDP View (because you're running headless perhaps), just do `pdp telcon` to connect a Teletype to the PDP-10.

Hit return to get the PDP-10's attention on the Teletype, and follow this example:

```
WHY RELOAD: SA  
DATE: 04-23-78
```

TIME: 1212
STARTUP OPTION: QUICK

The boot process prints progress, and takes some time. then, you're done with the Teletype on the PDP-10 console until it's time to bring the system down again.

Login on a user terminal

On the linux command line, log in on a proper terminal: `telnet localhost 2020`.
If you prefer, you can also use the VT-52 terminal simulation (or Datapoint, VT-05, etc):
`/opt/pdp10/bin/vt52 -B telnet localhost 2020 &`

In fact, you can open multiple telnet, VT-52 or other terminal connections, all to port 2020. It is a multi-user system after all, and after logging in on the second or third terminal, do a `sysstat` again. You'll see the occupied TTY's for each user.

In the new terminal, hit return to get the PDP-10's attention, and then type
`LOGIN 1,2` ; password is FAILSA. This will bring you to TOPS-10's 'dot' command prompt.

Here are the pre-installed users with their passwords:

[1,2]	FAILSA
[6,6]	MAINT
[7,7]	OPER
[10,*]	DIST
[100,100]	DEMO1

As you logged in as the administrator, you could see that simply by going into REACT:

```
R REACT
R (to read the user database)
T (to type it out, including passwords if you wish)
```

REACT is also where you should create an account for yourself. See the manuals.

Type `SYSTAT` to get a first overview of the system;
`DIR[1,1]` shows the MFD (the root directory if you will);
`DIR[1,4]` then shows the directory with lots of software installed.

As we are superficial 21st century people, first some light entertainment:
`R ADVENT` (and `Ctrl-C` once you're done)

Let's get a quick flavour of programming under TOPS-10. The Timesharing Handbook is a good companion to have at hand for this: https://bitsavers.org/pdf/dec/pdp10/TOPS10/1970_PDP-10_Timesharing_Handbook.pdf

Fortran

Installed on the disk images are quite a few programming languages: Cobol, Fortran, BCPL, Algol, Basic, APL.

A quick Hello, world in Fortran, found under the 100,100 account:

```
K/F
LOGIN 100,100 (password: demo1)
DIR

TYPE FTEST.FOR
COMPILE FTEST.FOR
LOAD FTEST
START
SAVE HELLO
RUN HELLO
```

This shows a difference between system programs and programs you load from your disk:

```
R file (system program)
RUN file (load from current disk)
```

But now comes the challenge! Editing a file. TECO is the classic editor, the default editor that you'll find pretty much on any PDP computer.

```
TECO FTEST.FOR
```

The teco manual is here: [http://www.bitsavers.org/www.computer.museum.uq.edu.au/pdf/DEC-10-UTECA-A-D%20decsystem10%20Introduction%20To%20TECO%20\(Text%20Editor%20And%20Corrector\).pdf](http://www.bitsavers.org/www.computer.museum.uq.edu.au/pdf/DEC-10-UTECA-A-D%20decsystem10%20Introduction%20To%20TECO%20(Text%20Editor%20And%20Corrector).pdf)

Mastering TECO is the hallmark of a proper DEC user. But – well. 21st century users might not want to climb the steep learning curve.

Not to worry, you can also run the relatively more comfortable SOS:

```
SOS FTEST.FOR
H (for help)
P (for print)
... and from there, we leave you to the manuals! Ctrl-C, Q to quit SOS
```

The SOS manual is here: http://www.bitsavers.org/pdf/dec/pdp10/TOPS10_softwareNotebooks/vol01/DEC-10-USOSA-A-D_SOS_Users_Guide_Jul75.pdf

As you'll have seen, it requires quite some determination to work with these editors.

Assembly

TYPE HELLO.MAC to see a simple Hello, world example.

Still logged in as user 100,100, copy hello.mac to test1.mac, and another copy as test2.mac:

```
R PIP
TEST1.MAC=HELLO.MAC
TEST2.MAC=HELLO.MAC
Ctrl-C
```

now, you could follow the steps you saw with the Fortran compiler (COMPILE, results in a .REL file, LOAD then turns it into a .SAV file). But this is quicker, and gives you an indication there's some intelligence in the build commands in TOPS-10:

```
LOAD TEST1.MAC
SAVE TEST1.MAC
RUN TEST1.MAC
```

Even quicker is EXECUTE, it will compile and run (but not save as an EXE program file!):

```
EXECUTE TEST2
```

Debugging with DDT

TYPE TEST1.MAC ...and note the code starts at the label START.

```
DEBUG TEST1
```

will load DDT, as well as the symbol table for you. From here, you will recognize things from ITS DDT. For instance, as the symbol table is already loaded by the DEBUG invocation of DDT, START/ Ctrl-J Ctrl-J will list you through the program.

Obviously, Ctrl-C to exit.

Logging out and shutting down

At each user terminal:

```
K/F
```

Once all users are logged out, in practice you're safe to just switch off the PiDP-10 (comes with no guarantee). Officially, there is a proper shutdown procedure though: on the Teletype console, do:

```
R OPSER
```

:KILL ALL

<*to be continued as your writer climbs the learning curve*>

TOPS-20: Manual section coming ASAP

<below is placeholder text, copied from the PiDP-10 Google Group.
This is just for now, a proper walkthrough will appear here ASAP>

guide:<https://groups.google.com/g/pidp-10/c/-UTX6Jx0bXc>

Tops 20 is a bit harder to boot. To the Boot> prompt you need to enter TOPS20: then it will as the structure name, enter "tops20" here. I am not sure why this is required. I have added same setting as klh10, and am using the same boot loader. Note boot loaders are specific to each version of TOPS 20.

```
> telnet localhost 1025
```

to which I was rewarded with:

```
BOOT V11.0(315)
```

```
BOOT>
```

I entered "tops20:" and got:

```
BOOT> tops20:
```

```
[BOOT: Loading] [OK]
```

```
----
```

Connected to the KL-10 simulator CON-TELNET device

```
BOOT V11.0(315)
```

```
BOOT>tops20:
```

```
[BOOT: Loading] [OK]
```

System structure not found. What is its name? TOPS20:

```
[TOPS20 mounted]
```

```
Panda Distribution, PANDA TOPS-20 Monitor 7.1(21733)-4 Internet: Loading host names [OK]
```

System res

Date and time is: Saturday, 20-April-2024 2:21AM

Why reload? SA

Run CHECKD? NO

DDMP: Started

[KNILDR: Loading mic20-Apr-2024 02:22:12 Internet: Network My-Network on, Output on rocode version 1(172) into Ethernet channel 0]

SYSJOB 7A(88)-4 started at 20-Apr-2024 0222

SJ 0: @LOGIN OPERATOR

SJ 0: @ENABLE

SJ 0: \$SYSTEM:STSJ1

20-Apr-2024 02:22:13 SYSJB1: SYSJB1 started.

SJ 0: \$^ESET LOGIN ANY

SJ 0: \$OPR

[NCP]: Waiting for ORION to start

20-Apr-2024 02:22:14 SYSJB1: Job 0:

20-Apr-2024 02:22:14 SYSJB1: Job 0: Panda Distribution, PANDA TOPS-20 Monitor 7.1(21733)-4

20-Apr-2024 02:22:14 SYSJB1: Job 1:

20-Apr-2024 02:22:14 SYSJB1: Job 1: Panda Distribution, PANDA TOPS-20 Monitor

% [Logger 20-Apr-2024 02:22:14]: Started at 20-Apr-2024 02:22:13

7.1(21733)-4

20-Apr-2024 02:22:14 SYSJB1: Job 2:

20-Apr-2024 02:22:14 SYSJB1: Job 2: Panda Distribution, PANDA TOPS-20 Monitor 7.1(21733)-4

20-Apr-2024 02:22:14 SYSJB1: Job 3:

20-Apr-2024 02:22:14 SYSJB1: Job 3: Panda Distribution, PANDA TOPS-20 Monitor 7.1(21733)-4

20-Apr-2024 02:22:14 SYSJB1: Job 4:

20-Apr-2024 02:22:14 SYSJB1: Job 4: Panda Distribution, PANDA TOPS-20 Monitor 7.1(21733)-4

20-Apr-2024 02:22:14 SYSJB1: Job 5:

20-Apr-2024 02:22:14 SYSJB1: Job 5: Panda Distribution, PANDA TOPS-20 Monitor 7.1(21733)-4

20-Apr-2024 02:22:15 SYSJB1: Job 1: @LOGIN OPERATOR

20-Apr-2024 02:22:15 SYSJB1: Job 3: @LOGIN OPERATOR

20-Apr-2024 02:22:15 SYSJB1: Job 0: @LOGIN OPERATOR

20-Apr-2024 02:22:15 SYSJB1: Job 2: @LOGIN OPERATOR

20-Apr-2024 02:22:15 SYSJB1: Job 2: @ENABLE

20-Apr-2024 02:22:15 SYSJB1: Job 3: @ENABLE

```

20-Apr-2024 02:22:15 SYSJB1: Job 4: @LOGIN OPERATOR
20-Apr-2024 02:22:15 SYSJB1: Job 5: @LOGIN OPERATOR
20-Apr-2024 02:22:15 SYSJB1: Job 0: @ENABLE
20-Apr-2024 02:22:15 SYSJB1: Job 0: $RESOLV
20-Apr-2024 02:22:15 SYSJB1: Job 1: @ENABLE
20-Apr-2024 02:22:15 SYSJB1: Job 2: $SMTJFN
20-Apr-2024 02:22:15 SYSJB1: Job 3: $MMAILR
20-Apr-2024 02:22:15 SYSJB1: Job 4: @ENABLE
20-Apr-2024 02:22:16 SYSJB1: Job 4: $IMAPSV
20-Apr-2024 02:22:16 SYSJB1: Job 5: @ENABLE
20-Apr-2024 02:22:16 SYSJB1: Job 1: $NETSRV
20-Apr-2024 02:22:16 SYSJB1: Job 5: $FTS
SJ 0:
SJ 0: 02:22:14      -- Can't rename file SPOOL:OPERATO
20-Apr-2024 02:22:16 SYSJB1: Job 5: FTS>TAKE FTS.CMD
20-Apr-2024 02:22:16 SYSJB1: Job 5: [FTS20: FTS event 38: spooler started]
R-SYSTEM.LOG --
SJ 0:      Current LOG file will be appended
SJ 0: OPR>TAKE SYSTEM:SYSTEM.CMD
SJ 0:
SJ 0: 02:22:16      --ORION logging disabled by job 1 OPERATOR at terminal 13--
SJ 0:
SJ 0: 02:22:16      --Output display for OPR modified--
SJ 0:
SJ 0: 02:22:16      --Output display for OPR modified--
SJ 0:
SJ 0: 02:22:17      --Output display for OPR modified--
SJ 0:
SJ 0: 02:22:17      --Output display for OPR modified--
SJ 0:
SJ 0: 02:22:16      Batch-Stream 0 -- Set Accepted --
SJ 0:
SJ 0: 02:22:16      Batch-Stream 1 -- Set Accepted --
SJ 0:
SJ 0: 02:22:16      Batch-Stream 2 -- Set Accepted --
SJ 0:
SJ 0: 02:22:16      Batch-Stream 3 -- Set Accepted --
SJ 0:
SJ 0: 02:22:17      Batch-Stream 0 -- Set Accepted --
SJ 0:
SJ 0: 02:22:17      Batch-Stream 1 -- Set Accepted --
SJ 0:
SJ 0: 02:22:17      Batch-Stream 2 -- Set Accepted --
SJ 0:
SJ 0: 02:22:1      02:22:18 From operator terminal 13 on no7      Batch-Stream 3 -- Set Accepted --
SJ 0:
SJ 0: 02:22:17      Batch-Stream 0 -- Startup Scheduled --
SJ 0:
SJ 0: 02:22:17      Batch-Stream 1 -- Startup Scheduled --
SJ 0:
SJ 0: 02:22:17      Batch-Stream 2 -- Startup Scheduled --
SJ 0:
SJ 0: 02:22:17      Batch-Stream 3 -- Startup Scheduled --
SJ 0:
SJ 0: 02:22:19      --SEND command completed--
SJ 0: OPR>
SJ 0: 02:22:23      -- Device status file initialization error --
SJ 0:      Changed CPU
SJ 0:      SYSTEM:DEVICE-STATUS.BIN file will be reset
SJ 0:
SJ 0: 02:22:23      -- Structure Status Change Detected --
SJ 0:      Previously mounted structure TOPS20: detected
SJ 0:
SJ 0: 02:22:23      -- Structure Status Change Detected --
SJ 0:      Structure state for structure TOPS20 is incorrect
SJ 0:      EXCLUSIVE/SHARED attribute set incorrectly
SJ 0:      Status of structure TOPS20: is set:
SJ 0:      Domestic, Unregulated, Shared, Available, Dumpable
SJ 0:

```

Well look at that... I hit ^C in the console window and it gave me a command prompt!

Panda Distribution, PANDA TOPS-20 Monitor 7.1(21733)-4

This system is for the use of authorized users only. Usage of this system may be monitored and recorded by system personnel.

Anyone using this system expressly consents to such monitor^C

```

@
@? Command, one of the following:
ATTACH DAYTIME INFORMATION LOGIN LOGOUT SYSTAT TERMINAL UNATTACH
or user name
@systat
Sat 20-Apr-2024 02:34:30 Up 0:21:39

```

0+8 Jobs Load av 0.39 0.36 0.23

No operator in attendance

System is remedial

Job	Line	Program	User	Origin
9*	5	SYSTAT	Not logged in	

1	13	OPR	OPERATOR	
2	DET	SYSJB1	OPERATOR	
3	14	RESOLV	OPERATOR	
4	15	NETSRV	OPERATOR	
5	16	SMTJFN	OPERATOR	
6	17	MMAILR	OPERATOR	
7	20	IMAPSV	OPERATOR	
8	21	FTS	OPERATOR	

@

So now I just need to create/learn some login accounts & passwords. :-). In the meantime I can play from the console.

Note: The user terminal (telnet localhost 10018) still doesn't work. Maybe someone here can shed some light on why that is (and maybe a list of WHEEL account/passwords) ;-). Here's what I see when I telnet into a user terminal:

```
@login operator
Password: dec-20
@zork
```

My problem is that I have a well-configured KI-10 (not KL-10) image of TOPS-10 that I built using the Blinkenbone emulator:

<https://retrocmp.com/projects/blinkenbone/simulated-panels/228-virtual-pdp-10-ki10-panel-the-holy-gral-of-blinkenlights-on-your-desktop>

I'd like to try to get this running on the PiDP-10 as it claims to have a KI-10 CPU version. But when I try to run it, it hangs up. If I can get the manual for the version of SIMH that the PiDP-10 runs on I think I can work it all out. Without that, I'm just making random stabs in the dark.

So if anyone can point me at the SIMH PDP-10 Simulator Usage manual for what's running in the PiDP-10 it would be greatly appreciated.

Networking on the PDP-10

The PiDP-10 comes with a working Chaosnet installed, so that the Pi and the PDP-10 can talk to each other. Two immediate uses present themselves:

File exchange between ITS and Linux

The utility program `mlftp` will do this, connect from Linux to ITS over Chaosnet, and read/write files. A minimal example:

```
/opt/pidp10/bin/mlftp -w its test ~/test
```

...will store the file in the `.TEMP.` directory of ITS. Run `mlftp` without arguments to see its options.

Or:

```
mlftp -w its -- "-READ- -THIS- GAMES;" read-this.txt
```

...to show an example with full ITS filenames. Note that the semicolon demarks the directory.

Terminal sessions over Chaosnet: supdup

Supdup was a groundbreaking virtual terminal; it is worth reading up on it online, because there is much more than meets the eye. But the simplest example:

```
/opt/pidp10/bin/supdup its
```

Note that *this* `supdup` is a Linux version of `supdup`, it is not a simulator. There is also a `supdup` in ITS itself, which will allow you to visit other ITS systems. `:supdup` will get it running. It is a very elegant way to connect to other systems over the global Chaosnet, but setting up networking beyond the host Pi is another section.

Chaosnet

Chaosnet was MIT's a LAN technology from 1975. The name refers both to networking hardware and to a protocol. You can have Chaosnet-over-IP, but also Chaosnet-over-Unix-sockets, and much more.

The PiDP-10 starts a minimal Chaosnet connection between ITS and the host Pi in the ITS bootscrip. It runs the external program `cbridge`, something you could do on the command line as well,

```
/opt/pidp10/bin/cbridge -c /opt/pidp10/bin/cbridge.conf
```

Cbridge can do much more, and `cbridge.conf` holds the key to everything. See <https://github.com/bictory/chaosnet-bridge>

Future Networking sections

(Work in progress, hints for the moment are:)

- SYSNET; TELNET 752 - TELNET client for TCP
- SYSNET; CHTN 15 - TELNET client for Chaosnet

ITS can be part of the global HECnet network, use TCP/IP, and other modern niceties. It can also be part of a re-created ARPANET. The IMP device is part of the simulation. Future updates of the manual will give much more detail (as we explore the practical setups).

Further reading

<https://github.com/obsolescence/pidp10/wiki/PiDP%E2%80%90testing>

<https://its.victor.se/wiki/start>

<https://chaosnet.net/>

Appendix: Running the PiDP-10 software on X86 Linux

During development, it turned out to be handy that the whole PiDP-10 software package also runs on a regular Linux laptop or desktop. In fact, it seems that it will also run on Windows 11 (not 10) if you install the WSL system.

The install is less smooth than on a Pi, it is basically a by-product of development. Still, if you want to try, it will not mess up your computer. As long as you say No to these install questions, nothing gets modified and everything can be removed by deleting `/opt/pidp10`. The install options to say No to, then, are:

- Automatically start the PiDP-10 core when logging in? → just say N
- Add a DEC flavour to the Pi's desktop? → just say N
- Reduce telnet release time? → just say N

- Install required dependencies for running the PiDP-10? → here, you have to say Y

After the install process has completed, do `cd /opt/pidp10/bin`
and type `tar xzvf binaries-for-X86-64.tar.gz`

This will overwrite the Raspberry Pi's ARM binaries with X86-64 binaries, and things will run. Although the binaries are not exactly the latest ones, I update them irregularly.

So – buyer beware, but it will probably work fine, no harm trying. Once the project sees less updates, I will make a polished X86 linux version with less caveats.

Appendix: assigned telnet port numbers for the PiDP-10

ITS:

Console Teletype: port 1025.

Some the terminal port numbers are dedicated to a particular terminal type, such as Imlac or VT52, and only allow one connection. These numbers are in the range 10010-10021 (subject to change).

Other port numbers are generic, allow more than one connection, and go to some of the terminal controllers: 10000 is for the slow teletype controller, 100002 is for the medium speed "Datapoint kludge" terminal interface that the PiDP-10 simulates, and 10003 is for the very fast "Morton box" interface. There was quite a variety of hardware in the MIT AI Lab...

TOPS-10:

Console Teletype: port 1025.

User terminals (can be multiple): port 2020.

Appendix: A Multi-Pi PiDP-10

It is a hack – but a brilliant hack always fits the ITS spirit. Terry Kennedy did this, which I think is very nice:



Two less power-hungry Pi 3's (although he used a Pi 4 and a Pi 3) are running headless TOPS-10 and TOPS-20. Of course, they will not drive the front panel, that is reserved for the Pi 5 mounted in the regular way. But the two extra Pi's get their power off the USB ports of the Pi 5, and to connect to the two 'peripheral Pis', Terry uses:

```
PiDP-10/ITS: [on main graphic display]
or > ssh pidp10.local <- talk to pi
```

```
TOPS-10: > ssh tops10.local <- talk to pi (for boot)
> telnet tops10.local 1025 <- connect to console
> telnet tops10.local 2020 <- connect to user terminal
```

```
TOPS-20: > ssh tops20.local <- pi
> telnet tops20.local 1025 <- console
> telnet tops20.local 10085 <- user term (not impl.)
```

As long as you use the 'official power supply for the Pi 5', which is beefier, this runs fine (and should run fine) with any old Pi 3 (and apparently, a more power-hungry Pi 4 too). Many people have a stack of older Pi's lying around, this seems a neat way to add them to the replicated DEC computer room we aspire to :-)

Nice! The only thing I could add to this is that you can drill mount holes to fix the two extra Pi's (as long as you put some insulation tape over the metal screw heads). And that the extra Pi's could happily run the PiDP-8 or PiDP-11 simulators as well.