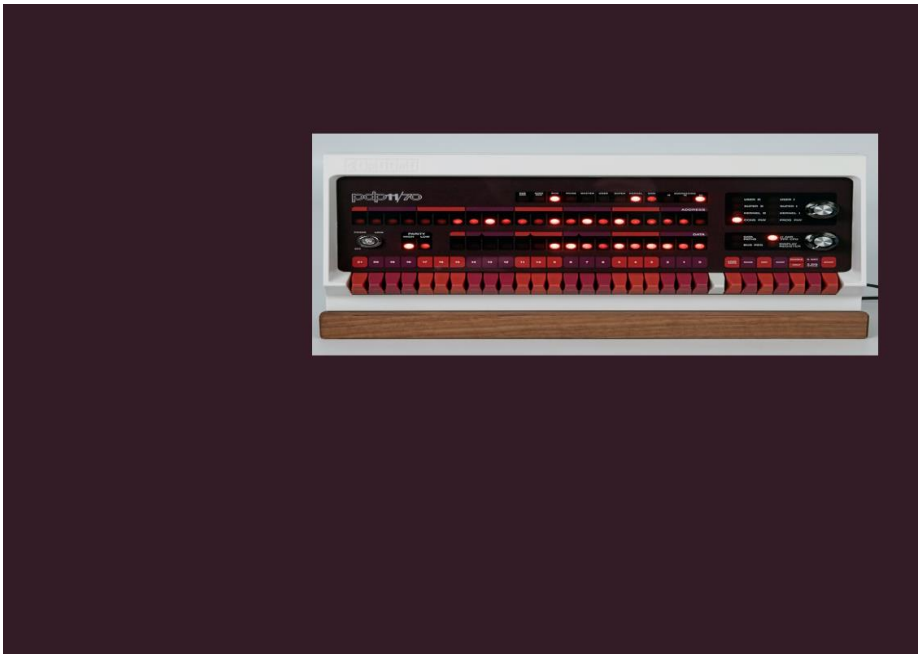
PiDP-11 USER MANUAL

DRAFT 29 April 2019



obsolescence.wix.com/obsolesce

Contents

Introduction

This manual will get a first-time user started on the PiDP-11, and then provides an overview of its set-up. In other words: a quick overview of what it is, how to use it, and how it works.

It is not a replacement for the web site: the Building Instructions and 'Hacks' are not included here for instance. It also does not replace the simh manuals – the go-to reference for tweaking the simulated PDP-11's hardware configuration. Lastly, each OS has its own set of manuals on bitsavers.org.

But this manual is probably where you want to start. It will make you comfortable using the PiDP-11 and the Raspberry Pi hidden inside, then gives a quick tour through the PDP-11 if you are new to the machine. The last section adds a cookbook-style list of things you probably should know about later on.

The PiDP was born out of the simple desire to bring PDP-11s into the living room. I hope that the PiDP-11 will bring new users and new activity to the PDP-11 architecture. Keeping the real machines going becomes quite hard, and so much can still be done in the PDP-11 playground that it should not fade away. It's just hard to decide where to start your adventure into the PDP-11 world. But that will be your problem.

The PiDP-11 also harbours hardware-hacking ambitions. As a motivation to actually do something with the PDP-11, consider it as a very powerful Arduino-style platform; it can control modern I2C peripherals (see the web site). As a society, we clearly need PDP-11s involved in home automation, for example. Plenty of things to do…

Acknowledgements. The PiDP-11 is based on the hard work of others. I was just the Master of Plastics at the end of a long road, giving a physical form to the real genius that lies inside: Bob Supnik's simh emulator, honed to perfection over decades, with Mark Pizzolato and many others adding continuous development to this day. Jörg Hoppe, whose BlinkenBone project gave a front panel interface to simh, and who shaped simh's behavior much closer to the blinky bits on the front end. And then, many early users of the PiDP-11 fixed my bugs and added capabilities. Mike Hill gave a lot of direction to the early project, Ian Schofield dug deep to fix bugs and to add graphics capabilities that are beyond what's in the regular simh. Mark Matlock provided a rich RSX-11 environment that includes Johnny Billquist's RSX-11 TCP/IP networking suite; Terry Kennedy provided bug reports and server space. Angelo Papenhoff provided polished versions of Ancient Unices, Stephen Casner did some serious debugging, [and so many others need to be named here but I need to get this manual out today!].

About simh

Let it be clear: the genius inside the PiDP-11 is simh, the emulator for many historically important computer systems. It has its roots in the MIMIC simulation system, designed in the late 1960's. Bob Supnik continued the work through the decades, with many contributors along the way. In recent years, Mark Pizzolato has been the driving force. See github.com/simh/simh .

Simh consists of two main parts. First, scp, the simh command processor. This is where CTRL-E brings you, and the command line is one of the things that makes simh so powerful. As a programming environment and debugger, it's hard to beat. Then, there is the actual PDP-11 simulator, which is what you see until you hit CTRL-E. You should read the simh users manual and the PDP-11-specific manual:
github.com/simh/simh/blob/master/doc/simh_doc.doc .
github.com/simh/simh/blob/master/doc/pdp11_doc.doc .

About the BlinkenBone project

The PiDP-11 is a BlinkenBone device. Created by Jörg Hoppe, BlinkenBone extends simh with front panel drivers. The project initially allowed you to **revive authentic front panels** with an adapter board. Useful if you have a real machine that has died; or for a museum that wants to present its exhibits 'alive' without silly electrical and maintenance bills. Later on, photorealistic on-screen **'virtual' front panels** were written in Java, for Windows and Linux.
The PiDP-11 is the third BlinkenBone option: a physical front panel replica. You can mix and match options; so you can set up the PiDP-11 to be just the front panel for a simh that is running on a fast PC.

1 BlinkenBoned: original & virtual front panels

2 UniBone: a Unibus card to integrate Linux and original PDP-11 hardware

The recent **UniBone** project goes a step further. It is a Unibus card, inserted into a real PDP-11 *either* to take over those peripherals of the machine that still work using a simulated CPU, *or* to provide simulated peripherals to the real CPU. For instance, it is possible to have a simulated PDP-11/20 take over the Unibus from an 11/05 and let the simulated CPU use the core memory and peripherals. The PiDP-11 front panel could be driven by (or drive, depending how you think of it) a UniBone card. The projects interact and will grow together. It is well worth exploring the http://retrocmp.com web site. You'll find detailed BlinkenBone documentation as well as many other interesting PDP-11 related things.

Section 1: Getting Started

Setting up the PiDP-11

This page assumes you have already set up your Raspberry Pi like you would do for any Pi. If you are new to the Pi World, first read the section *Setting up a Raspberry Pi*. It's what needs doing first. The PiDP-11 setup requires a 16GB or larger SD card.

The standard way to use a PiDP-11 is to use wireless **ssh** terminal sessions, **plus VNC** for optional graphics displays. Alternatively, just use the Pi's HDMI monitor and a USB keyboard. Or, if you have one, use a vintage serial terminal (plus maybe a HDMI monitor or VNC as the optional graphics display).

Download & Install

1 Make a /opt/pidp11 directory and go there:

```
sudo mkdir /opt/pidp11
cd /opt/pidp11
```

2 Download the pidp11 software:

```
sudo wget https://www3.ispnet.net/pidp11/pidp11.tar.gz
```

3 Unpack it so the software lives in its designated /opt/pidp11/ directory:

```
sudo tar -xvf pidp11.tar.gz
```

4 Run the **install** script so the PDP-11 autoboots when you switch on the Pi:

```
sudo /opt/pidp11/install/install.sh
```

5 Reboot and see the lights:

```
sudo reboot
```

Adding the PDP-11 operating systems collection

The above gives you the PiDP-11, but only a demo program (idled) to run. The next step is thus to download all the operating systems. **Download and unpack** the 'systems' collection of disk images:

```
cd /opt/pidp11
sudo wget https://www3.ispnet.net/pidp11/systems.tar.gz
sudo tar -xvf systems.tar.gz
```

Also, you could use `sudo wget https://www3.ispnet.net/pidp11/nankervis.tar.gz` to add that to the collection.

More things to tweak if you want:

- if you autoboot into the GUI, you need to open a terminal and type `~/pdp.sh` to 'grab' the PDP-11. Disable GUI autobooting: "Raspberry icon" ->Preferences-> Raspberry Pi Configuration. You can always start the GUI afterwards using the `startx` command.
- You can enable auto-login on the Pi, set that using `sudo raspi-config` and you'll no longer be bothered by Pi/Linux stuff; you'll just be dropped into the PDP-11 straight away.

Updating the pidp11 software

Usually, it will be the simplest & sufficient to rename the current /opt/pidp11 directory to something else (keep it as a backup), and run steps 1-3 from the top of this page. Then, move the systems directory from your old/renamed directory back into the new /opt/pidp11.

Using the PiDP-11

Booting different operating systems

Powering up the PiDP-11 with all SR switches down will boot a demonstration program from Mike Hill called *idled*. It's a little "PDP-11 poem in 64 words" that keeps the front panel lights busy – nothing else. Play with SR0-7 for different LED patterns. Look at the PDF and boot.ini in /opt/pidp11/systems/idled, they show some neat programming tricks and demonstrate how you can use the simh command line for programming, not just configuring, the PDP-11 hardware.

This is something harmless to boot in to, but normally you'd prefer to boot an operating system. This is done by setting an octal number on the front panel SR switches, and pushing on the Address rotary knob. It contains a hidden switch function (not present on real 11/70s!) which will restart the PDP-11.

> Octal numbers are important for the PDP-11; the SR switches are grouped in coloured blocks of three. Each block makes one octal number. So for booting unix v7, set the SR switches to octal 107, i.e. binary 001 000 111, as per the picture below. Octal is so much nicer than hexadecimal.

You can reconfigure all of this, see the pidp11/systems/selections file further down in this manual.

Here is the current menu:

DEC operating systems:	**Unices:**	**Demonstration programs:**
0001 rsx11mplus	0102 211bsd	1001 idled
0002 rsts7	0105 unix5	1002 blinky
0003 rt11	0106 unix6	
0004 dos11	0107 unix7	
0005 ias	0113 sysiii	**Nankervis System:**
	0115 sysv	1000 nankervis

Boot option 1000 is a special one, to some extent duplicating the systems shown above. It is a copy of the system from Paul Nankervis: a single PDP-11 configured with a huge number of drives with some 15 operating systems. See further down in this manual.

Try out the front panel

The front panel switches let you control the PDP-11 at any time. Toggle in a program or debug a running system in single-step mode. Very short introduction:

First, **halt the system** (depress HALT), and

- execute in **single-step mode** by repeatedly toggling CONT.
- move the HALT switch back up to the ENABLE position and toggle CONT to **continue normal operation**.
- when halted, **inspect a memory address (or register)**: set SR switches to form the address of a memory location, then toggle LOAD_ADRS. Hit EXAM to show its contents on the DATA leds.
- when halted, **modify the contents of a memory address (or register)**: set the SR switches to the desired address, toggle LOAD ADRS (look at the ADDRESS leds mirroring your SR switches), set the SR switches to the desired value, toggle DEP (look at the DATA leds and also notice the ADDRESS leds have auto-incremented for your next word of data).
- If you do not want to continue a running program, but instead **start executing another program** (that you might have toggled in, or which exists somewhere in memory) you enter the starting address using the SR switches, hit LOAD ADRS, then hit START (instead of CONT).
- **CPU registers** can be viewed and edited by inspecting 'special' addresses at the top of memory using the EXAM and DEP switches. R0 is at address 01777700, R1 at the next address, and so on.

For a short, practical experience in programming on the front panel, see the little step-by-step guide on www.retrocmp.com/attachments/article/146/consoleserialport.blinkenlight_instructions.txt

For a more in-depth overview of the front panel functions and indicators, see www.pdp-11.nl/pdp11-70startpage.html and for more detail, the indispensable PDP-11/70 Processor Handbook, Chapter 11 (bitsavers.trailing-edge.com/pdf/dec/pdp11/1170/PDP-11_70_Handbook_1977-78.pdf)

Shutting down the PiDP-11

Important. To shut the system down in an orderly manner, **push the HALT switch down and then depress the ADDRESS rotary knob**. Wait 15 seconds more before you cut the power.

Note that you ***also*** need to shut down many operating systems on the PDP-11 before you leave them. This is your responsibility, as it was on a real PDP-11. It's not necessary in RSX-11, before you enter the time and date. But any time *after* that step, RSX-11 requires a RUN $SHUTUP command. 2.11BSD and unix v7 requires you to enter the `sync` command three times before leaving. RT-11 does not need shutting down.

Leaving the PDP-11 to do things on the Pi (and then come back to the PDP-11)

You can slip back to the underlying Linux system's command line at any time using the `CTRL-A, d` keystrokes. Once you're done, return to the still-running PDP-11 by entering `~/pdp.sh` . To understand more about this, see the chapter on the *screen* utility in Section 3.

You can also stop the PDP-11 (CTRL-E, exit) and you'll find yourself on the Linux command prompt. In which case, you could start up the PDP-11 again by running /opt/pidp11/bin/pidp11.sh . Or, to restart it more formally inside *screen*, use `/etc/init.d/pidp11 start` and grab the screen using `screen -D -r`

Using the PiDP-11 software without the PiDP-11 hardware

Maybe you want to try before you buy, or perhaps you need a second PDP-11 to network to your 'real' PiDP. They do get lonely if not networked. The PiDP-11 software can be used without the physical PiDP-11 hardware attached. Of course, being Blinkenless means that you cannot control the system from the front panel switches.

Instead, after booting up for the first time, either:

- Press CTRL-E, type `cd ../rt11` (or any name from the boot menu), type `do boot.ini`
- Or alternatively (sometimes this is more thorough), exit and restart the PDP-11 simulator: type `CTRL-E`, `exit` to return to Linux. Type `cd /opt/pidp11/bin` , and restart the simulator using `./pidp11.sh X`. Where X stands for the operating system number that you want to boot into.

Setting up a Raspberry Pi

As there is a Raspberry Pi inside, there are endless configuration options you can tinker with. Which is fun, flexible, and highly confusing. Start with a **simple** setup, and add any desired confusion later on.
The PiDP can happily run as a PDP-11, and at the same time do anything else you'd like a Pi to do. Use it **concurrently** as a file server, media server or home automation system. Please test your Pi with the PiDP-11 software **before** you build the kit, so you know the software bit is good before you start soldering. The PiDP software runs fine without the PiDP hardware.

The basics

Download the normal Raspbian SD card image from www.raspberrypi.org/downloads/raspbian/ . Choose the full version with GUI (needs 16GB SD card), as the PiDP uses X in some situations. You can use a program called Etcher (www.balena.io/etcher/) to put the downloaded image file on your SD card. Remember: the default login is pi, with password raspberry.

A. If you use a HDMI monitor and USB keyboard:

1. Insert the SD card into the Pi and power up. Set up Wifi. You can either do that in the GUI, or using 'sudo raspi-config', whatever you prefer. Detailed instructions are on www.digikey.com/en/maker/blogs/raspberry-pi-3---how-to-connect-wi-fi-and-bluetooth.

2. Enable ssh and VNC. In the GUI: Raspberry Icon->Preferences->Raspberry Pi Configuration->Interfaces tab. In raspi-config: main menu, Interfacing Options. Don't enable Bluetooth unless you will not be using the built-in serial port of the PiDP.

B. If you do *not* have a HDMI monitor to plug in to the Pi: you can also set up wifi and ssh on the SD card before you even put it in your Pi. Then you can use ssh sessions remotely to do everything. Using VNC you can even use the GUI. There is really no need for a HDMI monitor. Three steps:

1. Insert the SD card into your PC, and on the boot partition (which a PC can see) create and empty file named 'ssh'.

2. Create a file 'wpa_supplicant.conf' and use a text editor to put the following in it (edit the ssid name and psk password to match your own wifi, and change the country code if you care):

```
country=us
update_config=1
ctrl_interface=/var/run/wpa_supplicant

network={
 scan_ssid=1
 ssid="MySSID"
 psk="MyPassword"
}
```

3. Insert the SD card into your Pi, boot, and see the following chapter to connect wirelessly.

C. If you want to do the setup using the standard TTL serial connector on the PiDP-11: you will need to enable the serial port. Edit /boot/config.txt (`sudo nano /boot/config.txt`) , and add a new line at the bottom: `enable_uart=1` . You can also just edit this file by inserting the SD card into your laptop and do it from there. You do not even need ssh & wifi or a hdmi monitor, you could do everything over the serial port. Look for PL2303HA USB-Serial connector cables on places like AliExpress, they work fine for a cost of about $1.80 including shipping.

*Important: **make sure to use a good power supply**. The newer the Pi, the more sensitive it is to even the slightest power glitch, leading to crashes and corrupted SD cards much more often than you'd think. Use the Official Raspberry Pi Power Supply if in doubt, it's only $12.50 or so. Note the Pi likes 5.1-5.35V, a 5.00V power supply is only marginally OK! Lastly, if you decide to power the PiDP through the key switch, and you use the Pi 3B+, read the Hardware Notes at the end of this manual.*

Wireless operation

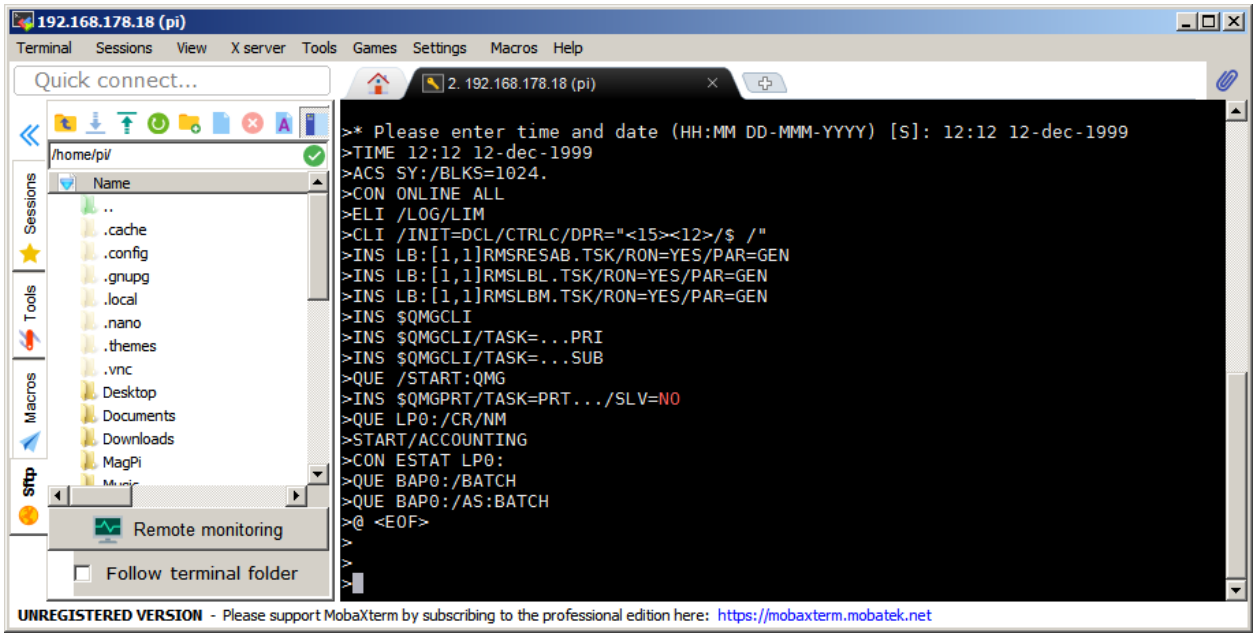
This section is not about the PiDP-11, just about the Pi itself. Many people do not realise how tightly the Pi can be integrated with your laptop. It makes using the Pi (and PiDP) much more pleasant. Place the PiDP anywhere and use it wirelessly from the sofa.

I greatly recommend an all-in-one Windows program called **MobaXterm** (mobaxterm.mobatek.net). Linux has its own tools, and other Windows programs are available (puTTY, Tera Term). But if you are new to the Pi, start with Moba.

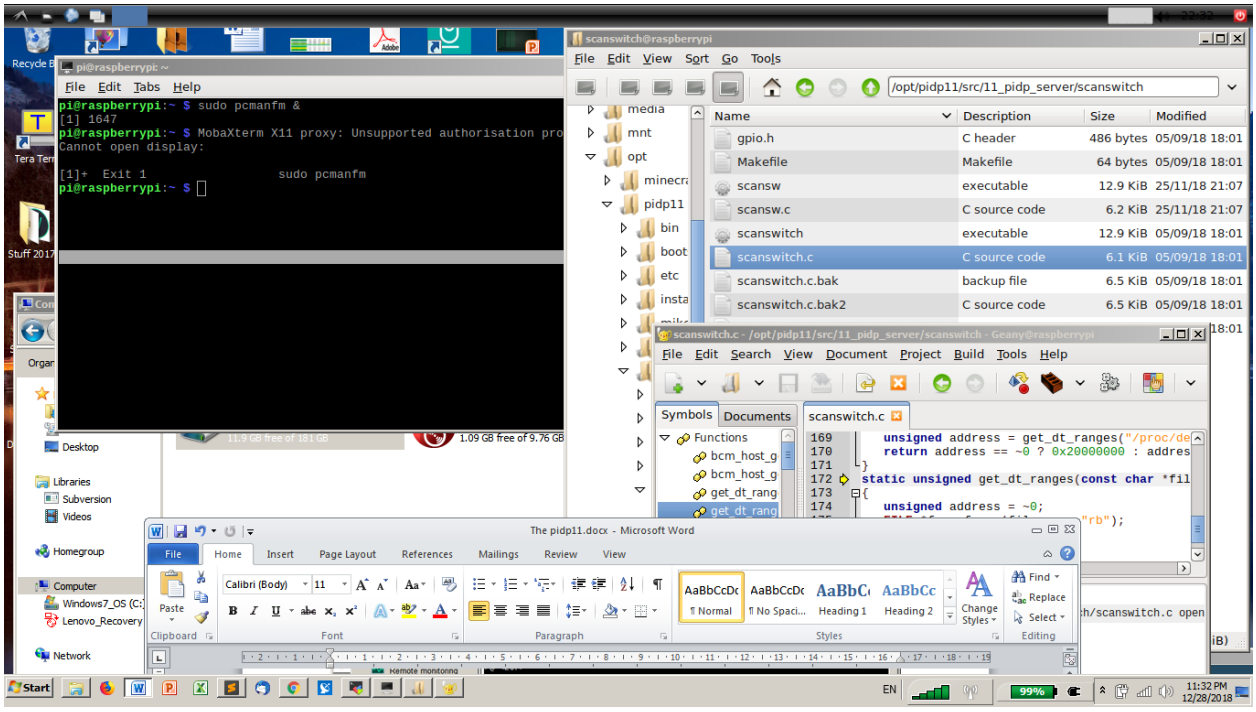
First, you will need to **find out the IP address of the Pi**. Use the Tools->Network Scanner in Moba to find that out, you'll see the Pi appear in the scanner's list (sometimes you have to repeat the scan a few times). It's the entry with ssh enabled, if you do not see its name. Click on the Deep Scan icon on that line and note the **port number**.

Now you can use Sessions->New Session to **set up an SSH terminal into your Pi**. Click on the SSH icon and enter the Pi's IP number. Check if the port number is right. You can also enter your user ID ('pi', normally) for added comfort.

To connect, double-click the newly created User Session (top right of the Moba window). You should be in. Also, you will see a file browser to the left of the terminal. You can **drag and drop files between your laptop and your Pi**, but of course you cannot save files in every directory of the Pi without proper permissions set up. Right-click text files in the file browser to edit them comfortably on your PC.



Playing with Moba some more will make you see that a GUI program on the Pi can easily run as a window on your laptop. Enter `lxsession&` in your ssh command line, and you'll get **Raspberry Pi GUI programs on your laptop mixed among your Windows applications**. See below:



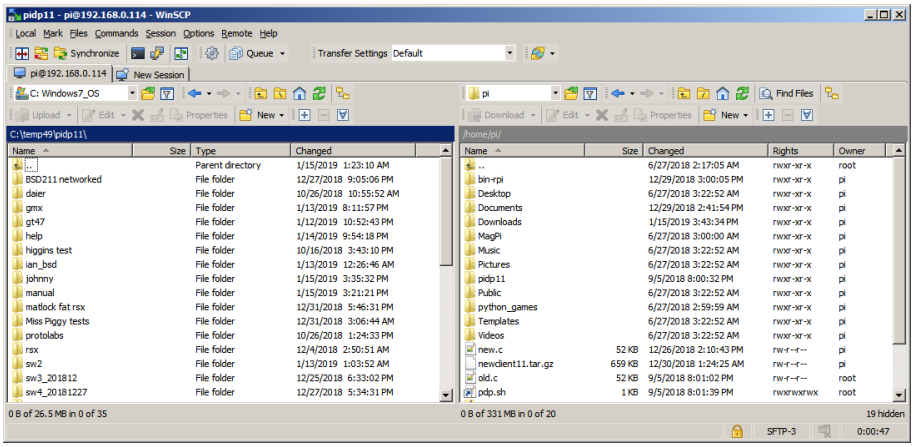
In the above picture, the Pi menu bar is set at the top, so you start Pi applications from above and Windows applications from below. Right-click that black menu bar to change its location. Perhaps a 30% sized Pi menu bar at the center bottom of the screen is better. Also, note that the window with the Pi's desktop can just be dragged off-screen. Pi programs move independent of it on your Windows desktop anyway.

For PiDP-11 purposes, if you don't care too much about security etiquette, the command **`gksudo pcmanfm&`** is useful. It allows you to edit and move files as superuser in the pidp11 directory, making it easy to tweak the PiDP-11 setup (with the acceptable risk of breaking something).

See www.raspberrypi-spy.co.uk/2018/12/remote-access-pi-using-mobaxterm/ for more detailed instructions on Moba.

You can also use **VNC** to get the standard Pi desktop onto your laptop. This is also the way to get any graphical output from the PDP-11 over a wireless connection, see the RT-11 section. You should use Real VNC Viewer (www.realvnc.com/en/connect/download/viewer/) rather than Moba. Set the resolution/display size to something acceptable from the Pi desktop (Raspberry icon ->Preferences-> Raspberry Pi Configuration->Set Resolution button).

Another very useful, Norton Commander-style tool to move files between your laptop and your Pi is **WinSCP**, see the picture to the right.



It can not just transfer files in and out of the Pi, but also in and out of the PDP-11 running under RSX-11M+ or 2.11BSD, once you have set up networking on the PDP-11. You can download it at winscp.net .

From this point on, you will find that using the Pi remotely is simple and painless. Not strictly necessary for use as purely a PDP-11, but you'll find it useful.

Understanding the PiDP-11 software setup

If you've installed the pidp11 software, the following is what happens. During boot-up, the pidp11 software is started up inside a virtual *screen* terminal and the front panel begins to light up. When you log in, you're given that virtual terminal screen, irrespective of how you logged in: HDMI monitor, ssh terminal , or serial port.

It's useful to know what is inside /opt/pidp11. What follows is a description of 'what is & does what'.

The pidp11/bin directory

You can manually start the PDP-11 with /opt/pidp11/bin/pidp11.sh, this script is also used when the Pi boots up.

> But starting the PDP-11 this way – as opposed to what is done at boot time – means the PDP-11 console is not wrapped in a virtual *screen* terminal, so you can't pop in and out or transfer the console to somewhere else! For that, you'd first start a new *screen* with pidp11.sh.

You can edit the list of bootable operating systems by editing /opt/pidp11/system/selections. It's just a text file that determines what SR switch numbers (in octal!) at boot time will start which operating system. Have a look at it.

- **pidp11.sh**: starts the PDP-11 simulation. To explain:
    - it exports a DISPLAY variable so SDL knows where to paint any graphics screens (VT11)
    - it makes a directory in the RAM disk, for temporary control files when starting simh
    - it makes sure an RPC portmapper is running so simh can talk with the front panel
    - it kills any front panel driver that might still be running and be in the way
    - it uses the scansw program to read the front panel switches
    - it uses getsel.sh translate the octal SR switch number into a subdirectory name
    - it creates a temporary bootscript in the RAM disk for simh to boot off
    - it starts the client11 (simh) and server11 (front panel driver) programs
    - it loops back to restart simh and the front panel driver whenever the front panel driver is used to exit simh. This happens when the user sets some SR switches and depresses the Address rotary knob to reboot.
    - If the user exits simh manually (CTRL-E, exit), the script ends with bringing down the front panel server too.

- **client11**: the simulator
  a symlink to the simh binary in /opt/pidp11/src/02.3_simh/4.x+realcons/bin-rpi/pdp11_realcons
  If you changed the source code, you can run *pidp11/src/makeclient.sh* to update simh (takes

a while) or *pidp11/src/makeserver.sh* to update the front panel driver. After that, just run pidp11.sh again to see your code improvements in action.

- **server11**: the BinkenBone front panel driver
  This is a symlink to the actual binary in /opt/pidp11/src/11_pidp_server/pidp11/bin-rpi/pidp1170_blinkenlightd.

- **scansw**: quickly reads the SR switches, to determine what system to boot.
  This is a symlink to the actual binary /opt/pidp11/src/11_pidp_server/scanswitch/scansw

- **getsel.sh**: a script that translates an octal number (from the SR switches) into a directory name.
  If the lookup table in the selections file does not contain your SR switch number, 'default' is used. Which is systems/idled.

- */run/pidp11/tmpsimhcommand.txt* is a boot script created on the fly and fed to simh when simh is started up. This script just changes the default directory to your chosen operating system (i.e., pidp11/systems/unix7), and then calls the boot.ini boot script in the subdirectory of the chosen operating system.

You might wonder why the above is all done in a pidp11.sh script. Why not just program it into the simh binary? The answer is that this approach maintains compatibility with normal (non-PiDP) BlinkenBone simh clients. Only a few lines need to be inserted to a stock BlinkenBone simh sources to enable the SR switch front panel rebooting, and even that is optional. You may not care too much, but it allows you to run the simh client side on your super-fast desktop PC if you want. And it'll add flexibility in the future.

The pidp11/install directory

Install.sh sets up the PDP-11 to start at boot time, inside a Linux *screen* virtual terminal. If you don't want to autostart the PDP-11 at boot time, there is no need to do the install. You then just run `/opt/pidp11/bin/pidp11.sh` whenever you want to start the PDP-11.

- install.sh will:
  - install the Linux 'screen' utility if not done already,
  - insert pidp11/etc/rc.pidp11 into the Raspberry Pi's boot process so it is started for you automatically at boot time (and checking the file actually exists),
  - insert pidp11/etc/pdp.sh into your home directory and add it to your .profile . pdp.sh is used to return to the PDP-11 terminal inside a *screen* session, when you have left the *screen* session with the PDP-11 to do something on the Linux command line.
    So you leave the PDP-11 terminal with the 'CTRL-A, d' keystrokes, and you return to it using ~/pdp.sh . In special cases, a more forceful 'screen –D –r' is necessary though.

Install.sh will also install the pcap and SDL2 packages, which are required to add networking and graphics when compiling the simh client. **If** you do not use install.sh because you prefer the PDP-11 to only start up if and when you want it, you need to install these packages manually:

```
apt-get update
apt-get install libsdl2-dev
apt-get install libpcap-dev
```

The pidp11/systems directory

This is where your operating systems live. Each one is stored in its own subdirectory (for instance, pidp11/systems/unix7/) which contains:

- **boot.ini**: a simh command script telling simh what to do to boot up. That means things like attaching disk and tape images, and configuring the PDP-11 hardware to match what is needed for the operating system. Such as terminals, network interfaces etc.

- the actual **disk and/or tape images** containing the system, which are used in your boot.ini. Things like RP06 or RK05 image files.

Adding a new operating system is as simple as storing its files in a new subdirectory in pidp11/systems/, and adding the name of that new subdirectory into the pidp11/systems/selections text file with an octal reference number in front of it. You can pick that number yourself to boot it from the front panel. The updated front panel boot menu is shown when you restart the PDP-11.

The pidp11/src directory

- **src/02.3_simh** contains the BlinkenBoned version of simh, and you will recognize this from the normal simh source code if you are familiar with that. Recompile using `/opt/pidp11/src/makeclient.sh` . There are some improvements over regular simh in the graphics area, thanks to Ian Schofield.

- The **src/11_pidp_server/pidp11** subdirectory contains the server (front panel driver) for the PiDP-11. Recompile using `/opt/pidp11/src/makeserver.sh` .

- The little scansw helper program is in **src/11_pidp_server/scansw** . Recompile simply using `sudo make` in that directory.

After running one of the above compiles, you can immediately try your code changes by running `pidp11.sh` again. It might be offensive to some that there are shell scripts involved in the compilation process, instead of just a pure makefile. Sorry.

Installing your own PDP-11 software

The PiDP-11 comes with lots of historical software ready to boot. But you might want to add your own systems. The previous chapter is useful to read through, it will demystify what is going on. The recommended way of adding your own PDP-11 software is as follows:

- make a new subdirectory inside `/opt/pidp11/systems/`
- `sudo nano /opt/pidp11/systems/selections` will open that text file containing the list of bootable software the PiDP knows of. Add a line with (1) the octal number that you want to use to boot into your new software, a tab, and then (2) the name of your directory. Now, any time you want to start your software, just (re)boot with that octal number on the front panel.

Inside your new subdirectory will at least be a `boot.ini` text file which sets up simh for you. The following lines are required (although not the 4M, you might sometimes want to install less memory).

```
set cpu 11/70, 4M
set realcons host=localhost
set realcons panel=11/70
set realcons interval=20
set realcons connected
```

Simh is highly configurable, and though it might be daunting at first, some of its special features are worth knowing. You will likely use `boot.ini` to enable storage devices and attach disk/tape image files to them, and end with a boot command to start the freshly configured PDP-11. But you can also enter a program through the ini file, even using PDP-11 mnemonics. That is a great way to write a bare-metal program in fact; see `systems/idled/boot.ini` as an example. In short, read the two simh manuals: the general simh manual and the simh-PDP11 manual. It's not complicated but quite a bit to understand. Just start with hitting CTRL-E and experiment a bit with the simh command line.

Section 2: A Quick Tour through PDP-11 Operating Systems

Historical Unix

The PDP-11 was the first computer for which you could get Unix.

To celebrate the 50th anniversary (yes, 2019) of Unix, the PiDP-11 has Unix v5, v6 & v7 installed as well as System 3, System 5, 2.11BSD (the most usable PDP-11 Unix) and Ultrix-11. The last two actually come with TCP/IP networking.

A huge effort has been made through the TUHS (www.tuhs.org/) to reconstruct the earliest Unix systems, sometimes literally from old tapes left under cabinets for decades. A good starting point for exploring the end result is a.papnet.eu/UNIX/ , where the PiDP-11 Unix images came from. Googling will reveal many, many more interesting materials.

Study Unix v6

The famous Lions Commentary on v6 was a highly prized possession, photocopied from owner to owner due to copyright issues at Bell Labs. It's now online at lemis.com/grog/Documentation/Lions. Still studied today in Computer Science courses, a guided tour of the Lions commentary is available as an audio podcast: mytuner-radio.com/podcasts/cs-450-unix-v6-audio-commentary-michael-saelee-504050393.
If you want a rigourous entry into programming the PDP-11 and Unix v6, this course will get you quite far: http://doc.cat-v.org/unix/v6/operating-systems-lecture-notes/script/OS.pdf

Play with Unix v7

V7 will make you feel much more at home than v6. And the original documentation is excellent. First read a.papnet.eu/UNIX/v7/files/doc/03_beginners.pdf for a quick overview. The rest of the documentation is in the same doc/ directory, and might be one of the best introductions to unix ever: http://a.papnet.eu/UNIX/v7/files/doc/

If you are the type of person who likes a good bone to chew on, why not figure out this C program:

```
short main[] = {
        277, 04735, -4129, 25, 0, 477, 1019, 0xbef, 0, 12800,
        -113, 21119, 0x52d7, -1006, -7151, 0, 0x4bc, 020004,
        14880, 10541, 2056, 04010, 4548, 3044, -6716, 0x9,
        4407, 6, 5568, 1, -30460, 0, 0x9, 5570, 512, -30419,
        0x7e82, 0760, 6, 0, 4, 02400, 15, 0, 4, 1280, 4, 0,
        4, 0, 0, 0, 0x8, 0, 4, 0, ',', 0, 12, 0, 4, 0, '#',
        0, 020, 0, 4, 0, 30, 0, 026, 0, 0x6176, 120, 25712,
        'p', 072163, 'r', 29303, 29801, 'e'
};
```

It was the first winner of the IOCCC (see ioccc.org/1984/mullender/ for more information) and thus, will break your brain. See how deep you want to go, but it gives you reason to delve into v7.

To get it going, first read about the basics of ed, the v7 editor, in the documentation. You can simply paste the source code through your terminal into ed. As you will find out, you have to edit line 7 because '#' does not paste very well. Amend the line by using \#. Before you know it, ed is no longer quite as alien as you thought. The program then compiles with `cc mullender.c` and this being normal unix, you run `./a.out`.

> Before you do, make sure that your Backspace key generates a CTRL-? and not a CTRL-H. The Delete key (generally assigned to Backspace on terminal emulators) is for v7 what CTRL-C is today. You won't have problems with lxterminal but puTTY or Moba need that set up.

At this point, to figure out what mullender.c does, you can hurt yourself with adb, the v7 debugger, or look around with the help of the simh command line. It contains everything from disassembly to a fancy breakpoint system to explore a running system. People back in the day would have given an arm and a leg for such a tool, and today it is ideal as a friendly way of learning what is going on inside. Start with `HELP STEP` on the simh command line if you don't want to read the simh manuals straight away.

<work in progress>

A Quick Tour of 2.11BSD

2.11BSD is the nicest Unix on the PDP-11, virtually identical to 4.3BSD on the VAX. They even shared the same 4.3 manuals. This is a pretty spiffy Unix. It's recommended to use the PiDP-11 from ssh, rather than from a terminal window in the GUI if that is what you normally do. Because you'll use vi as the editor, you need VT-100 terminal emulation. And things like Moba or puTTY have that, but Raspbian's lxterminal does not. If you want to use the GUI, xterm is also an option – see elsewhere in the manual for using that.

Boot up the PiDP-11 with the SR switches set to 102 (Switch SR6 =1, then 000 010).

You see the system boot up, followed by a ':'. Hit enter to boot 2.11BSD,which then ends with a "#" prompt. You are now in single user mode. You can look around, edit files, and so on.

The section below is an abridged version of Bob Meyer's PDF guide:
groups.google.com/d/msg/pidp-11/_gfc-_4YC24/FiYaZaOQDgAJ

Setting up networking

This chapter assumes you have plugged an ethernet cable in your Pi. If you prefer to set up PDP-11 networking over wifi instead of ethernet, see the 'networking' section in a later chapter. For now, the recommendation is that you first try things over Ethernet and then explore wifi as an alternative.

To start multi-user mode, press CTRL-D. After a few lines are displayed you will see "Assuming NETWORKING system ...". Since the config is not complete for ethernet, there are errors, and this message will hang for a few minutes. There are a couple more sections that hang. Just be patient. In the end you will see a login prompt. Log in as root (no password) and look around, type `uname -a` and so on.

Since we are getting ready to open the PiDP-11 to your LAN and possibly the world beyond your firewall if you so choose, now is a good time to give root a password with the `passwd` command.

There are three files that need to be edited: `/etc/hosts`, `/etc/netstart` and `/etc/resolv.conf`. You need to have an Ethernet cable plugged in (and working, obviously); and you need to know your gateway IP address. Presumably something like 192.168.1.1 or 192.168.0.1.

> A two-line intro to the vi editor:
> ESC :q! is abort, ESC :wq is save & exit. Move your cursor on top of a character and press 'x' to delete it or 'r' to replace it. 'i' lets you insert text and if you have inserted enough, press ESC.
> But make sure you use a terminal with VT-100 (puTTY, Moba) or you get a garbled screen.

Now, choose an IP address for the PiDP-11 under 211BSD. For example, used below, 192.168.1.44. Just make sure 44 or whatever you pick is not used by anything else in your network (like the Raspberry Pi underneath!). Now edit the three files to set up the right IP address of your router and

the chosen IP address of the PiDP-11. You just need to edit these two IP addresses and the broadcast IP range, nothing more.

```
# vi /etc/hosts
      127.0.0.1 localhost
      192.168.1.1 router.home.lan router
      192.168.1.44 pdp11.home.lan pdp11

# vi /etc/resolv.conf
      domain home.lan
      nameserver 192.168.1.1
      nameserver 8.8.8.8
      nameserver 8.8.4.4

# vi /etc/netstart
      […check just these 4 lines to see if they are what you want…]
      hostname=pdp11
      netmask=255.255.255.0
      broadcast=192.168.1.255
      default=192.168.1.1
      […]
```

Type sync three times to clear out the disk buffers and restart the PiDP-11 (if the SR switches are still set to 102, just press on the Address rotary knob. Alternatively, press CTRL-E and type 'do boot.ini').

You will notice that this time, after you type ^D to start multiuser mode, the process doesn't hang, but proceeds smoothly. The only error should be relative to the line printer, which has not been set up.

Log in as root with the password you assigned earlier. Now for the test: enter `ping google.com`

If you don't get a series of responses from google, then compare all the files described above with what you have in the host and in the PiDP-11. Comments & questions are best placed on the forum in the thread about ethernet on BSD2.11 (groups.google.com/forum/#!topic/pidp-11/_gfc-_4YC24).

Looking around

You are now able to telnet and ftp in to the PiDP-11. For Windows users: WinSCP provides a very nice file exchange mechanism. You can transfer and compile C programs to your heart's content; this is a very useable system! If you want to study the BSD source code, there is a phenomenal online guide to understand the inner workings: wfjm.github.io/home/ouxr/

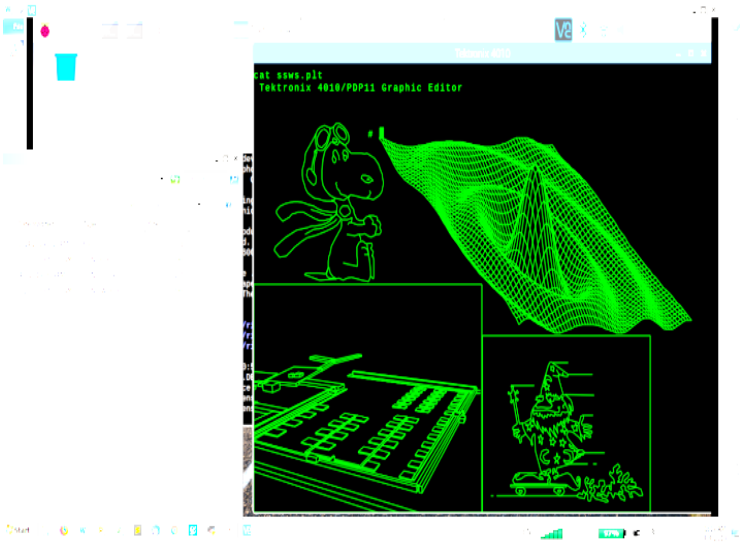
It probably makes sense to install an ftp server on the Raspberry Pi side, so you can bridge between the PDP-11 with 2.11BSD and the Raspberry Pi. Do so with

```
sudo nano /etc/proftpd/proftpd.conf
```

The Pi's ftp server is ready for use straight after installing.

Continue the tour

Rene Richarz set up a web page with lots of how-tos, from programming in C and Pascal, to making a networked weather station on BSD and using (simulated or real) Tektronix graphics terminals. This picture shows what the PiDP-11 can do using his Tektronix 4010 simulator.



His **highly recommended how-tos** are on github.com/rricharz/pidp11-2.11bsd .

The Tek simulator with instructions on how to run it on the Pi: https://github.com/rricharz/Tek4010

(Tip: the .plt files can be ftp'd over to 2.11BSD, and you display simply with `cat plotfile.plt`)

This code snippet shows how easy it is to do vector graphics in C:

```
/*  tek.c 2.11 BSD tektroxix graphics test    */
/*  rricharz 2019 for 2.11BSD Unix on PiDP-11        */
/*  Uses Berkley C syntax!                    */
/*  If you want to use this, you need to      */
/*  login from a Tektronix terminal or emulator      */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "tekio.h"

double pi = 3.14159265;

int main(argc, argv)
int argc;
char *argv[];
{
        double step;
        double mx = MAXX/2;
        double my = MAXY/2;
        double r = MAXY / 2.2;
        double deg, x1, y1, x2, y2;
        printf("This program requires a Tektronix terminal!\n");
        clearScreen();
        step = pi / 100.0;

        x1 = 0; y1 = 0;
        for (deg = step; deg < 2*pi+step; deg += step) {
                x2 = r * sin(2*deg);
                y2 = r * sin(3*deg);
                drawVector((int)(x1+mx), (int)(y1+my), (int)(x2+mx), (int)(y2+my));
                x1 = x2;
                y1 = y2;
        }

        moveTo(10,10);
        exit(0);
```

```
}
```

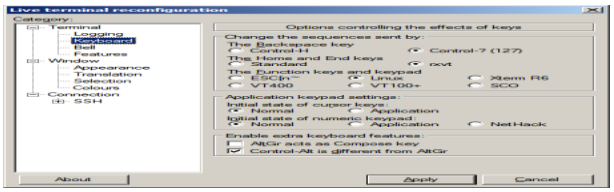
A Quick Tour of RSX-11M Plus

This section owes everything to Johnny Billquist & Mark Matlock

RSX was probably what DEC wanted you to run on a big PDP-11, and for good reason. The 'M+ 4.6' version installed on the PiDP-11 is very up to date, including a range of programming languages and Johnny Billquist's TCP/IP package. It can easily be networked and be used as a web server, for instance.

Also, you should explore HECnet, a worldwide network of real and simulated DECnet systems: www.update.uu.se/~bqt/hecnet.html

Terminal Setup

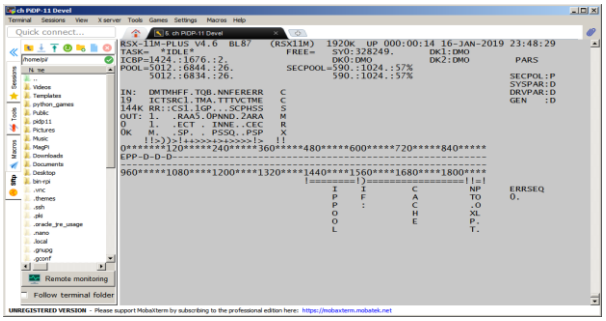


- To make the Backspace key act rather than the Delete key: select top row, 2nd radio button in putty and Moba (right-click in terminal, select Change terminal settings…) and putty: Select Terminal, Keyboard, select Backspace as CTRL-? (127)

- To set lowercase input, enter `SET /LOWER=TI:` on the command line.

Learning your way around, first steps

This assumes you are entirely new to RSX and just want to look around. RSX is different. As in, not like Unix. Some quick tips: CTRL-Z is what RSX programs expect if you want to leave them. Run `ADVENTURE`, leave with `quit`. You ought to do a RUN $SHUTUP to shut down the system. A quicker way to shut down is `STOP/ACC REBOOT` , and then `SCP STOP`. Don't be too afraid of disk image corruption if you do not shut down properly by mistake, all that happens is that a log file gets corrupted and that is fixable (ask on the forum). `RMDEMO` (pictured) gives you an interactive overview of the running system that will become more meaningful once you get to know your way around. And `SWR` shows you the value of the SR switches on the front panel.



I found this to be the best introduction to RSX: http://skn.noip.me/pdp11/resources/rsx/DEC-11-OMBGA-A_D_RSX-11M_Beginners_Guide_Jun77.pdf

Others swear by Pieper's *A guide for Users*: www.bitsavers.org/pdf/dec/pdp11/rsx11/Pieper_RSX_A_Guide_For_Users_1987.pdf

Make sure you at least read enough to move around the file system (there is no CD to change directories, and no (sub)directory structure in the Unix sense. But `SET /DEF=[IP]` will bring you into the directory of the TCP/IP package, and things like `SET /DEF=[0,0]` brings you into unnamed UFD's. Do a `DIR`. The RSX editor is `EDT`. Once you're in, type `change` to move to the comfortable full-screen editor. Emacs is also present: `EMA`.

Setting up Networking (Ethernet)

This chapter assumes you have plugged an ethernet cable into your Pi. If you prefer to set up PDP-11 networking over wifi instead of ethernet, see the 'networking' section in a later chapter. For now, the recommendation is that you first try things over Ethernet and then explore wifi as an alternative.

Thanks to Johnny Billquist's work, RSX is fully online and can do some wonderful things that you should explore. During boot time, say No when the system asks you for LAT (no need now) and yes to TCP/IP. Once booted up, try a `ping google.com`. It should just work. Do `IFC SHO ALL/NUMERIC` to show network status, or:

```
>ifc sho if0:
Intf   State Address                      Broadcast       MTU   ACP    Line
IF0:   DHCP  Frodo/24                     192.168.0.255   1500  ETHACP UNA-0

>ifc sho rou
Internet routing table:
Dst                   Gwy                  Use        Ref        Flg IF
Broadcast             192.168.2.9          0          0          001 IF0:
192.168.2.0/24        192.168.2.9          0          0          001 IF0:
127.0.0.0/8           Localhost            0          0          001 IF1:
Default/0             192.168.2.1          0          0          000 IF0:
```

Try telnetting in to the system. Now is a good time to read up on the networking package with all its goodies: mim.update.uu.se/tcpipdoc . That file is actually served up from RSX!

Setting up email with BQTMAIL

Enter `MAIL`, and look around. To configure it for your mail provider: <work in progress>

Web & ftp servers

They are up and running. Have a look from your PC at http://<IP address of RSX system> . Of course, you can also set your browser to ftp://<IP address> and check in on the ftp server.

In fact, as of January 2019, the first PiDP-11 is serving a web page on the internet: www.rpg.fi/pdp11/ . Not bad for what really is a machine designed in the very early 70s, right?

Networking fun with XLisp

<work in progress>

Programming Languages installed on the system

Installed on the system are F77, BP2, DTR, C, Cobol, APL and XLisp.

If you like exotic, you can actually program in APL using the custom APL character set, thanks to a Tektronix 4013 display. Here is the Tektronix terminal simulator: groups.google.com/d/topic/pidp-11/dlJcVUFBfxs/discussion ; and here is a quick how-to from Mark Matlock: groups.google.com/d/msg/pidp-11/9cMQQWGwg40/xCmH01_vAgAJ

The DECUS library of RSX software

A second 1.6GB disk image can be added to the `systems/rsx11mplus/` directory, containing the contents of the RSX freeware CDROM produced by Tim Shoppa some years ago. This has almost every RSX SIG tape produced between 1979 and 1990 available as a virtual .DSK file that can be mounted from inside RSX copied to a scratch area and explored. You can download this archive using `wget http://rsx11m.com/PiDP11_DU1.zip`. Unzip and save in the `systems/rsx11mplus` directory. The `boot.ini` file already expects it. See the special section later in the manual for a detailed walkthrough.

Using your network printer from RSX

Really: just `PIP TC:"192.168.0.yourprinter";9100=file.txt` . Change the IP address in the last line to your printer's IP address and you can pipe stuff directly to it. If you want to alter font size etc. take a look at [IPF77]NETPRINT.FTN to see how easy things are to do in F77.

Vintage RSX-11 Gaming

See Section 3 for a how-to on setting up the multi-user MTREK game, and an assortment of VT-100 games. Log in with multiple terminal(emulators) over telnet and you'll have a party going.

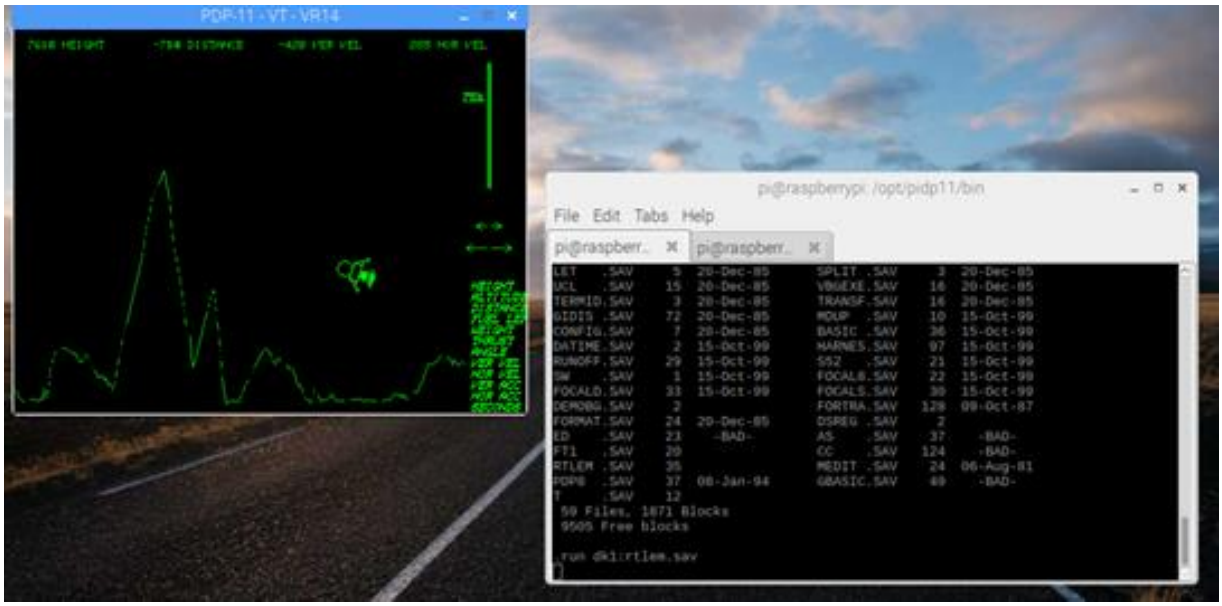
A Quick Tour of RT-11

With thanks to Ian Schofield, who brought graphics to life

> *20190204: the current setup of rt11/boot.ini enables the vt11 device by default. For that to work out, your Raspbian must have the GUI up and running. If you installed on a command-line-only Raspbian, it will fail. In which case, edit the boot.ini by commenting out the* `set vt enable` *line. But no graphics then!*

DEC's basic-but-nice real time operating system will be familiar to users of OS/8, CP/M, or even MS-DOS, and is easier to befriend than RSX or RSTS for newcomers. `DIR DK1:*.SAV` will show you the programs on the second disk pack. `HELP` or `HELP *` will show you the commands available.

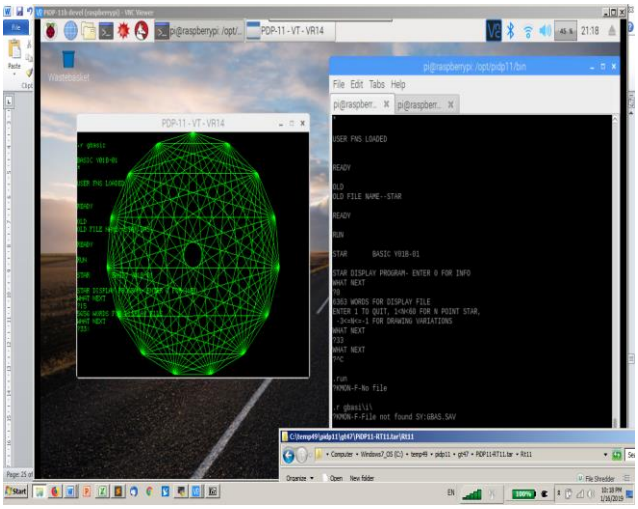
First, entertainment. The system is equipped with a VT11 vector graphics display system and VR14 CRT. The emulated display is shown on the HDMI monitor (use VNC if you don't have a monitor attached, works fine). Read `systems/rt11/graphics.txt` for configuration options (for instance, setting it to full-screen mode).



To have a quick experience of early video gaming: enter `R RTLEM.SAV` to start Lunar Lander, which is what the VT11 is often remembered for. Your mouse replaces the original light pen, and you click on the bar and arrows on the right of the display. But there's rather more to it. Read `systems/rt11/lunar.txt` for instructions. When you've crashed your lunar lander, just reboot the system (depress the Address rotary knob with SR switches still set to 03).

After you've rebooted, explore the VT11 as a video terminal. Enter `GT ON` and you'll see your cursor move over to the CRT, away from your regular terminal. Imagine you only had a Teletype rather than a video terminal; then this was it was obviously a Big Thing. Please note that you still have to use the keyboard of your regular terminal though!

Now that you're working on the CRT display, enter `DIR *.BAS` and `DIR *.GT`. Both file types are basic programs. Enter `R GBASIC`, then hit return when you see the * prompt. Enter `OLD`, `STAR.BAS`, `RUN`, then `15`.



Try some of the other Basic programs, some are quite nice. For the seventies this was a pretty fancy and expensive bit of hardware.

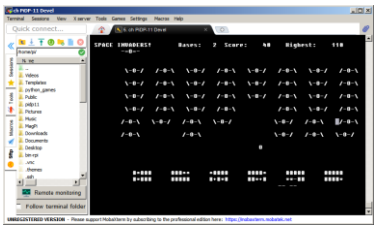
Basic, however, is basic. Disk DK1: contains a FORTRAN with VT11 graphics libraries for proper scientific purposes. The manual, as ever, is on Bitsavers.org.

This RT-11 system is equipped with RL02 removable disk packs of 10MB each. Enter `GT OFF`. Let's remove the DK1: pack and replace it with another one that contains VT-100 video games, for play on normal terminals. Press `CTRL-E` to enter the simh command line, and enter `detach rl1`. Now attach the new disk with `attach rl1 rt11games.rl2`, followed by `continue`. You are back in the PDP-11 after a virtual disk change.

The following assumes that you are using a VT-100 compatible terminal emulator. Moba or puTTY are recommended. The lxterminal window of the Pi's GUI will not do.

> So if you are working from the plain terminal window in the GUI, this is a good time to ssh in to the PiDP-11 using Moba or puTTY. You can then 'grab' the PDP-11 terminal with either `~/pdp.sh` or (if that won't work) the more forceful `screen -D -r`. Now that you are on a VT-100 terminal, continue the tour. (If you insist on GUIs, install xterm, that's proper VT-100)

Do a `DIR DK1:*.SAV` to see some of the games (there is more, however). A decent Pac Man is there, but I believe it requires a VT-52 terminal? `RUN DK1:SPCINV` will give you Space Invaders at any rate. Dungeon, Adventure, Star Trek and Super Star Trek are also present.



Lastly, see the .VT files on the newly mounted disk pack. Enter `TYPE CASTLE.VT` and enjoy the animation.

Of course, there is more to RT-11. It is a nice environment for programming. On the disks, you will find MACRO-11, FORTRAN and C, plus the KED full-screen editor (first read avitech.com.au/?page_id=959 though), even a PDP-8 emulator… for a quick introduction of RT-11, see the PDP-11 How-To book, section 2.0: uknc.narod.ru/Doc/pdpbook.txt. The RT-11 user manuals on Bitsavers.org will get you compiling assembly and Fortran afterwards. There is loads of RT-11 software on Bitsavers.

Section 3: Things you should know

Various hard & software topics

This manual has only described the basic operation of the PiDP-11.

Much more is possible: connecting multiple serial terminals, networking, emulating more advanced graphics displays.

The PiDP-11 can also be used as a (biggish) Arduino, because the I2C interface of the Raspberry Pi is brought into the PDP-11 as a Unibus device for instance. Program modern electronics from a proper CPU architecture.

This section should be read *in addition to* the information on the web site. Specifically, theTechnical Details and Hacks pages contain information not repeated here. For many PiDP-11 special topics, the web site at obsolescence.wixsite.com/obsolescence/pidp-11-connectivity-options is a good starting point – as is the user forum at groups.google.com/forum/#!forum/pidp-11.

Online PDP-11 Documentation

One of the best things about Digital's computers is the documentation that comes with them. Tens of thousands of pages in hundreds of manuals are available as downloadable PDFs from www.bitsavers.org/pdf/dec/pdp11/ . Start with the www.bitsavers.org/pdf/dec/pdp11/1170/PDP-11_70_Handbook_1977-78.pdf to get an overview of the hardware; then move on to whatever operating system you find most interesting.

Available PDP-11 Software

Although many of the available operating systems are included in the PiDP-11 distribution files, a tremendous amount of additional software is available on http://www.bitsavers.org/bits/DEC/ (DECUS and PDP-11 subdirectories) and spread across the internet. The DECUS PDP-11 collection for RSX-11M is also included as a mountable hard disk with floppy images as a downloadable add-on to the PiDP-11 software bundle.

File exchange between PDP-11 and the Pi

ftp is very effective if you have set up networking, for those operating systems that support it. See other chapters in this manual.

John Forecast's **fsio** allows you to read/write DOS/BATCH-11 and RT-11 disk images from the Pi. RSX and RSTS can also access RT-11 disks. Download from groups.google.com/d/msg/pidp-11/idH1DztUnDs/w8xitrxYCgAJ . See fsioSimh.txt, included in the archive, for examples.

Graeme Wightman's **pdp** (Pseudo Disk Processor) is a similar utility, with its roots dating back to Ultrix(!). The current download location is groups.google.com/d/msg/pidp-11/DXUADP7eeiU/kljrjp1qCgAJ

Jon Brase wrote a tool to convert raw binaries into absolute loader format **paper tape images**: github.com/jwbrase/pdp11-tools. There's also a tool to convert octal data into PDP-11 binary format.

Terminal Confusion

If you are new to the PDP-11 you may find the whirl of terminal options confusing. The problem: different types of emulated PDP-11 terminal connectors can be 'plugged in' to different types of (emulated or real) terminals on the Pi. And you can mix & match to your liking. The following should clear up the confusion.

First, in the default setup, what you see as the PDP-11 terminal is a combination of the PDP-11's console terminal (on the real hardware, the single DL11 serial port into which you would plug the terminal for the system administrator) and the simh command console. The simh presence is noticeable only when you hit CTRL-E (which freezes the PDP-11 and brings you into the simh command line until you enter 'continue'). But you can separate thePDP-11 and simh consoles if you wish. See one of the sections below.

Secondly, the simulated PDP-11 thinks you are connected through its console terminal but in reality, you have logged in either via telnet, ssh, the Pi's one serial port, using the lxterminal on the Pi's GUI or who knows what. The PDP-11 does not know any of this. They're all the same one-and-only DL11 console terminal from its perspective.

Thirdly, PDP-11s of course can have more than just the one terminal attached. Additional hardware (DC11, KL11, DL11, DZ11) provided the extra connections in different ways. All of these can be enabled in simh, but of course something must be hooked up to them. And there you again have the same range of options on the Linux side: ssh, telnet, USB-serial, terminal window on the GUI - you can set up what you want as a substitute for the serial terminals on the real PDP-11. Even the real terminals.

Lastly, real terminals typically meant VT-100 or VT-220 terminals. Unfortunately, lxterminal, the standard Linux GUI command line window, is not VT-100 compatible. If you use a full-screen editor or play a game like Space Invaders, that becomes a problem. Easily solved: log in over ssh for instance. Or install xterm if you want to keep using a window on the GUI, xterm is VT-100 compatible.

> How to move to Xterm when the PDP-11 is running: `CTRL-A, d`; then:
> (1) 1st time setup: `sudo apt-get update`; `sudo apt-get install xterm`;
> (2) moving over: `xterm&`; click on the new xterm window and `type ~/pdp.sh`.

Using screen, a quick walkthrough

The idea: screen is a program that acts like a virtual terminal. You can log in to it, detach from it, run programs in it. Why is that useful for the PiDP?

- You can pop in and out of the PDP-11 terminal to do stuff on the Pi, the PDP-11 just keeps going in the background.

- The PDP-11 terminal can be started on the Pi's HDMI display. Then, later on, grabbed on your laptop over ssh. Or on an xterm display. All the time, the PDP-11 just keeps running and does not care.

Ctrl-a d detaches you from screen's virtual terminal and returns you to the Linux command line. Now, re-connect to your screen by a screen –r . screen –D –r reconnects you even if screen itself objects to it. For instance, because another terminal is still connected. Now you're back in screen, do the following:

Ctrl-a ESC: brings you into the scrollback buffer. Use the cursor keys to scroll.
You can copy and paste in this mode: <space> to start marking text, <return> to copy the marked area. ESC to leave this mode. Then, CTRL-a] will paste into your prompt. Or, CTRL-a > will save the clipping to a file. Handy too: Ctrl-a < lets you paste text from some file into the terminal.

Ctrl-a | splits your screen vertically. And Ctrl-a S splits horizontally. Ctrl-a TAB jumps across splits, and in the new 'empty' slot, Ctrl-a c will create a new session. Now, you have your PDP-8 on the left, and your Pi on the right! Ctrl-a X closes the *split*, but that *session* still lives on in the background. Do another Ctrl-a c, just to create another new session. Now, to see a listing of all three screen sessions, do a Ctrl-a " , and you can select which session you want to go into. Ctrl-a <number> is a quick way of popping in and out of sessions. To end a session, type exit if the session happens to be a Linux terminal session. Or Ctrl-a k to kill and close the session, which is a bit rude. Even ruder is Ctrl-a \ , which kills everything running inside screen, and screen itself.

Screen also has a command line of its own. Ctrl-a : to enter it. Type dinfo<return> after that and screen will tell you what type of terminal you are using. But you can do all sorts of things in the screen command line.

Need to send Ctrl-a to the PDP-8? Screen is intercepting it. But Ctrl-a a will send the PDP-8 a regular Ctrl-a. Lastly, the man page on screen is very useful – even for people who do not like man pages normally!

Separating the simh command console from the PDP-11 console

One useful thing you can configure in your boot.ini is a separation of the simh console (it talks to you when you press the HALT, DEP, EXAM etc. switches or enter its simulator command line using CTRL-E) and the PDP-11 console. In the default setup for the PiDP they are one and the same display. But for proper authenticity (and because it gives you VT-100 compatibility), it makes sense to have the PDP-11 terminal in a separate telnet session away from the simh console. The simple way is to add a line

```
set cons telnet=23
```

to boot.ini. Now, log in to the PDP-11 using telnet, in this case localhost, port 23.

With multi-user operating systems, you may also leave user 1 on the combined simh/PDP-11 console and give user 2 his 'pure' PDP-11 terminal. You can also use a dedicated serial port instead of a

telnet session. See the simh manuals: the general simh manual and the simh-PDP11 manual. Also, see the *Enabling up to 4 serial ports* section.

Enabling up to 4 serial ports

You'd use 4 USB-serial cables for them, plugging their USB ends into the Pi. The web site contains the hardware details. But adding this to your boot.ini file will make the PDP-11 use them (use either the PDP-11 dz or vh device, depending on the situation):

Here is a setup for the boot.ini of RSTS/E 10.1. This is for the DHU, but the incantation works by changing vh to dz for other operating systems too:

```
set vh lines=16
set vh dhu
set vh nomodem
set vh enable
; Note that these are "shuffled" so the cabling matches the panel
attach vh line=0,connect=/dev/ttyUSB3
attach vh line=1,connect=/dev/ttyUSB1
attach vh line=2,connect=/dev/ttyUSB0
attach vh line=3,connect=/dev/ttyUSB2
```

The 4 serial ports as Linux terminals

Normally, you should not do this. You should keep the 4 USB-Serial ports free for use by simh as a VH or DZ device, controlling the terminals all by itself. But maybe you want your glorious VT-220 terminals as regular Linux terminals. Even then, you could still telnet in to the PDP-11 with them that way (it's the endless range of setup options that make it all a bit confusing). To set terminals up for use by Linux, and thus making them unavailable as PDP-11 VH or DZ terminals):

1. Copy the file `serial-getty@.service` from `/lib/systemd/system/` to `/etc/systemd/system/`

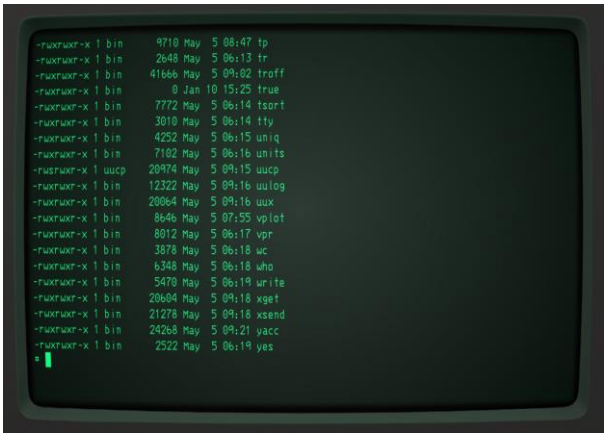
2. The file must be renamed as `serial-getty@ttyUSB0.service`, so that it points to your serial terminal which will be ttyUSB0 (or 1, or 2, or 3 depending on how many terminals you have. You need one file for each USB-serial adapter).

3. The file may need to be edited with the parameters of your serial terminal, or the default setting may work. In my case I changed the first line in the [Service] section to read:
`ExecStart=-/sbin/agetty ttyUSB0 9600 vt100`

4. Now, enable the service for each adapter with: `sudo systemctl enable serial-getty@ttyUSB0.service`. You should get a 'Creating sim link ....' message. The service will now start up at every boot. Reboot.

But: if you want to use your terminals as simh devices and not as Linux devices after all, remove the /etc/systemd/system/serial-getty@ttyUSB0.service file again.

A very nice CRT emulator



See the screen shot. There is a simulator for all sorts of CRT displays: cool-retro-term. It might improve life with your flat-panelled laptop.

Linux users can download it here: github.com/Swordfish90/cool-retro-term .

You might run it on your Linux laptop. You can, however, also run it on the Pi itself if you use a HDMI monitor or VNC. For that:

```
# Install the prerequisites:

sudo apt install build-essential qmlscene qt5-qmake qt5-default
qtdeclarative5-dev qml-module-qtquick-controls qml-module-
qtgraphicaleffects qml-module-qtquick-dialogs qml-module-qtquick-
localstorage qml-module-qtquick-window2 qml-module-qt-labs-settings qml-
module-qt-labs-folderlistmodel

# Get it from GitHub
git clone --recursive https://github.com/Swordfish90/cool-retro-term.git

# Build it
cd cool-retro-term

# Compile
qmake && make
```

Before starting the program (`./cool-retro-term`) you will need to enable the Pi's GPU using `sudo raspi-config` → `advanced options` → `GL Driver` → `GL Fake KMS`, and reboot. When in Cool itself, switch off a lot of the CPU intensive graphics options (noise, jitter) and performance becomes acceptable.

Note that the author of Cool asks for a small contribution if you like his work, see the github page.

Boot configuration 1000: The Nankervis System

Boot configuration 1000 is a special one, to some extent duplicating other systems that are available through rebooting with the front panel SR switches.

This is a copy of the system from Paul Nankervis: a single PDP-11 configured with a huge number of drives with some 15 operating systems.

All can be booted into by setting the SR register to 1000, after which you use 'boot xxn' to select the drive/OS to boot.

Paul's site skn.noip.me/pdp11/pdp11.html is well worth a visit for the annotated walkthroughs that will give you a quick hands-on introduction to the operating systems.

Paul Nankervis Collection - List of guest OS's

Start with 'boot *disk*' and follow the instructions

Disk	OS	Instructions
RK0	Unix V5	Boot using: **unix** then login as **root**
RK1	RT11 v4.0	The lightest/fastest OS here
RK2	RSTS V06C-03	Boot and login as **1,2** with password **SYSTEM** or as **11,70** using **PDP**
RK3	XXDP	Diagnostic OS and utilities
RK4	RT-11 3B	Distribution for RT-11 Version 3B
RK5	RT-11 V5.4F	Distribution for RT-11 Version 5.4F
TM0	RSTS 4B-17	Boot ROLLIN from TM0 and restore DK0 with "DK:<MT:VIXEN/REW". Reboot from DK0 with "/BO:DK" and login as **1,2** with password **SYSTEM**or **11,70** using **PDP** . All case sensitive.
RL0	BSD 2.9	Boot using: **rl(0,0)rlunix** CTRL/D to get to multiuser
RL1	RSX 11M v3.2	Login as **1,2** with password **SYSTEM**
RL2	RSTS/E v7.0	Option: <LF> Suboption: <LF> ... Login as **1,2** using **SYSTEM** or **11,70** using **PDP**
RL3	XXDP	Larger version of diagnostics - including PDP 11/70 utilities
RP0	ULTRIX-11 V3.1	CTRL/D to enter multiuser mode. Login as **root** with no password
RP1	BSD 2.11	Will autoboot and enter multiuser mode. Login as **root** with no password
RP2	RSTS/E v9.6	Answer boot questions and login as **1,2** (password **SYSTEM**) or **11,70** (no password)
RP3	RSX 11M v4.6	Starts logged in as **1,2** (password **SYSTEM**) - user accounts **200,1** (no password) or **11,70** (password **PDP**)

If your PiDP-11 lives on a bookshelf

Mine does. Then you have to get up and toggle the front panel switches every time you want to boot into a new OS. The alternative is simple: when the PiDP runs idled (and presents its boot menu on the terminal) just press `CTRL-E`, do a `cd ../system_name` and then `do boot.ini` . It prevents having to get up from the sofa.

The RSX SIG software collection – and a little demonstration

As mentioned earlier in this manual, a 1.6GB disk can be added to the systems/rsx11mplus directory. The disk contains a collection of mountable disk images, and the following how-to of accessing their contents is from Mark Matlock.

I thought it would be helpful to give an example of how to use the RSX SIG collections on DU1:[1,1] on the DECUS disk. To get the DECUS disk on your PiDP-11/70, do the following on the Pi command line (to explain, the second line, rm, is to remove the dummy placeholder generated when the RSX boot.ini was run before and did not find a DU1 image then):

```
cd /opt/pidp11/systems/rsx11mplus
sudo rm PiDP11_DU1.dsk
sudo wget http://rsx11m.com/PiDP11_DU1.zip
sudo unzip PiDP11_DU1.zip
```

Reboot the PDP-11 into RSX (set SR switches to 1, depress Address knob). Now, the DU1 disk is mounted and the RSX SIG files are in DU1:[1,1]. So, back in RSX with the freshly added DU1 disk now present:

```
SET /DEF=DU1:[1,1]
```

How to find your way around: one way to search the archive is to use GREP. The installed RSX version of it has a slightly different syntax:

```
GREP DU1:[1,1]*.TXT /SE:"BASIC"
```

You will see all references to "BASIC" in a long list. Note what file the matches come from, i.e. RSX86B.TXT. You can also view the RSXnnn.TXT files with EDT, or if you have a 48 line terminal window try VTL. If you only have 24 lines, MOR is the same file viewer with 24 lines. It is similar to EDT but read-only and can handle wild-card file names. NEXT is the command to look at the next wild-card file.

From this list, I saw that the version of Michael Reese / Frank Borger Basic-11 that I wanted was on a number of tapes/disks but the most recent version appeared to be on RSX86B.TXT This was a collection of RSX shareware distributed at the 1986 Fall DECUS meeting. There was a version for IAS / RSX11D in [330,21] and the M/M+ version is in [300,21]. Each .TXT file has a corresponding .DSK container file that Tim Shoppa has created so it is much easier to work with than virtual tapes. To make it even easier he created a MOUNT.CMD command file so…

```
@MOUNT RSX86B
; RSX86B
vcp conn [1,1]RSX86B.dsk/mou:pub:ovr
VCP - Device VF0: has been assigned.
asn vf0:=RSX86B:/gbl
@ <EOF>
```

Now VF0: is actually a disk drive with the 1986 RSX SIG tape contents. To make it easier to remember which VFn: has what contents, a logical name was also created so that it can be used thus:

```
DIR RSX86B:[300,21]
```

You should see the files for BASIC-11. There is not much extra space on these .DSK files (check for yourself with `PIP RSX86B:/FR`). So let's copy it to DU1: but first we need to create the matching directory:

```
UFD DU1:[300,21]
PIP DU1:[300,21]=RSX86B:[300,21]*.*
```

Now let's go to DU1:[300,21] and try to build it.

```
SET /DEF=DU1:[300,21]
```

Usually, there will be a README.1ST description of what's there and what to do with it. Do a `TYP README.1ST`, and you'll see there are three steps to get this particular bit of software going on RSX.

1. Check BASPRE.MAC to see if I want to make any changes to the operation of the Basic (it is fine as is)
2. Assemble the Macro-11 files using 11ASM.CMD
3. Task build using BIGBASIC.CMD

Take a quick look at 11MASM.CMD to see if we see anything that would create a problem: `TYP 11MASM.CMD`. If you feel that looks all good, let's do it! Also, it is more fun to watch this step from RMD on another logged in terminal to RSX. RMD will show you the programs running and if you ran the MAC below from TT0: then RMD T,T=MACT0 will show you haw MAC is opening the various files on different LUNS and snap shots of the registers and task states. RMD I let's you see the disk activity.

```
MAC @11MASM
MAC -- Open failure on input file
SPAWN,SPAWN/-SP=BASPRE,SPAWN
```

Well, we got an error because MAC couldn't find SPAWN.MAC. Why?

```
DIR RSX86B:[*,*]SPAWN.*
Directory VF0:[330,21]
20-JAN-2019 11:00
SPAWN.MAC;1          14.         12-JUL-1985 18:23
SPAWN.OBJ;1          2.          12-JUL-1985 18:24
Total of 16./16. blocks in 2. files
```

There it is. The [330,21] directory was the IAS / 11D version of BASIC-11 that this RSX version was branched from. And someone forgot to put SPAWN.MAC in the [300,21] directory for RSX.

```
PIP DU1:[300,21]=RSX86B:[330,21]SPAWN.MAC
MAC SPAWN,SPAWN/-SP=BASPRE,SPAWN
```

Now we are ready to task build it. Again, let's take a quick look at the TKB command file. Do a `TYPE BIGBASIC.CMD` to look what'll be done. The installed name will be …BAS so if we run it from TT0: it will run as BAST0.

```
TKB @BIGBASIC.CMD
```

No errors! Check to see the .TSK that was made:

```
DIR *.TSK
Directory DU1:[300,21]
20-JAN-2019 11:05
RSXBASIC.TSK;1       114.    C  20-JAN-2019 11:04
Total of 114./114. blocks in 1. file
```

Now install it:

```
INS RSXBASIC
```

If there is any question about what name the task will have, try it out:

```
BASIC
RSX BASIC   //  VERSION 29AU86   //   READY
EXIT
```

Now for a good BASIC game to try things out. One I like for the pre-installed BasicPlus2 (BP2) compiler/interpreter is LUNAR. I have a copy of it in DU:[6,1]LUNAR.B2S

```
PIP LUNAR.BAS=DU:[6,1]LUNAR.B2S
BAS
RSX BASIC   //   VERSION 29AU86   //   READY
OLD "LUNAR"
READY
RUN
```

Good Luck! P.S. -- If you just wanted to play LUNAR and not do care about all the rest of this, I have a compiled BP2 copy of it in `DU:[6,1]LUNAR.TSK`. So you don't have to install this second version of Basic from the DECUS disk collection at all, really. Just do:

```
SET /DEF=DU:[6,1]
RUN LUNAR
```

Serious gaming on RSX-11

(thanks to Mark Matlock)

There has been an interesting discussion of various games that can be played under various PDP-11 operating systems. One of my favorites has been MTREK or Multi-User StarTrek. Up to 8 players can play against (or together against) other players or robot ships.

Part one of this post is about how to find and install the programs necessary to play MTREK.
Part two gets you playing the Classics (Pac Man, Centipede, Missile Command) on a VT-100 terminal.

First, all this software and over a decade of RSX SIG tapes (approximately 500 MBytes) are available as described in the previous section. You may have already downloaded this but if not, do it now:

```
cd /opt/pidp11/systems/rsx11mplus
sudo rm PiDP11_DU1.dsk
sudo wget http://rsx11m.com/PiDP11_DU1.zip
sudo unzip PiDP11_DU1.zip
```

After that, reboot the PiDP11 into RSX (SR switch to 1, depress Address knob), and a new DU1 drive should automatically be mounted.

```
SET /DEF=DU1:[1,1]
```

The latest version of MTREK is on RSX83B.DSK which can be found by searching the *.TXT files with GREP or MOR as described in manual

```
@MOUNT RSX83B
DIR RSX83B:[*,*]MTREK.*

Directory VF0:[307,110]
15-APR-2019 21:52

MTREK.BLD;1          6.          19-OCT-1983 12:25
[…]
MTREK.TKB;1          1.          19-OCT-1983 13:00

Total of 157./157. blocks in 7. files
```

So the MTREK files are in [307,110] and we need to copy them to DU1: to make sure we have plenty of free disk space.

```
UFD DU1:{307,110]
PIP DU1:[307,110]=RSX83B:[307,110]*.*;*
SET /DEF=DU1:[307,110]
DIR *.TSK

PIP -- No such file(s)
```

Hmmm.. No prebuilt tasks apparently, so we will have to compile from the sources.

```
DIR *.CMD

Directory DU1:[307,110]
15-APR-2019 21:58

MTREK.CMD;1          7.          19-OCT-1983 12:53
MTREKUP.CMD;1        1.          19-OCT-1983 13:11

Total of 8./8. blocks in 2. files
```

Ok, let's look at the MTREK.CMD

```
TYPE MTREK.CMD
```

A long listing follows but it is clear that this is the build command file. It is also apparent that MTREK is written in RATFOR which is Rational Fortran (an oxymoron?). Anyway, when I did a directory of all the files in [307,100] the .FTN files are there so we don't need to process the .RAT files into .FTN files. Let's build MTREK from what we have in Fortran.

```
@MTREK
>;
>; MTREK.CMD
>;
>; This command file will preprocess, compile & build Multi-trek for
>; your system. The following system tasks must be installed (or
>; available thru flying installs) with the expected names:

[..]

>* Do you want to preprocess the RATFIV source? [Y/N]:N

[..]

>; Taskbuild the global common & tasks
>TKB @TRKCOM.TKB                !global common
>TKB @MTREKD.TKB                !universe driver
>TKB @MTREK.TKB          !ship
>TKB @MTREKINI.TKB       !universe initialization
>TKB @ROBOT.TKB          !robot ship
>TKB @MAP.TKB            !universe map maker
>;
>; All done!!
>; Do @MTREKUP to install Multi-trek
>;
>@ <EOF>
```

No errors! Now, read MTREK.DOC to learn how it’s played, then

```
@MTREKUP
```

You will be asked for a random number to build your unique Universe, install the memory common that holds the universe, install the MTREKD daemon that moves stuff around, and install the MTREK ships and Robot ships. Also, MAP.TSK will create a large text file UNIVERSE.MAP to guide you to "Boldly Go Boldly Where No Man has Gone Before".

Part two: Classic video games on the VT-100 terminal

This is a collection of arcade games (PACMAN, CENTIPEDE, MISSLE COMMAND, that use VT100 graphics and some incredible escape sequence coding again in that RATFOR language.

```
SET /DEF=DU1:[1,1]
@MOUNT RSX83A
UFD DU1:[312,354]
PIP DU1:[312,354]=RSX83A:[312,354]*.*;*
SET /DEF=DU1:[312,354]
DIR *.TSK

Directory DU1:[312,354]
15-APR-2019 22:31

BKO.TSK;1           58.     C  18-MAY-1983 10:50
BKOFSL.TSK;1        42.     C  18-MAY-1983 10:50
CEN.TSK;1           67.     C  18-MAY-1983 10:50
CENFSL.TSK;1        51.     C  18-MAY-1983 10:50
MSL.TSK;1           70.     C  18-MAY-1983 10:51
MSLFSL.TSK;1        54.     C  18-MAY-1983 10:51
PAC.TSK;1           68.     C  18-MAY-1983 10:51
PACFSL.TSK;1        52.     C  18-MAY-1983 10:51
WONDER.TSK;1        30.     C  18-MAY-1983 10:51

Total of 492./492. blocks in 9. files
```

The .TSK files that DO NOT use the FSL (FCS Supervisor Library) will run fine. They have .DOC files to explain them and are best played on a terminal with the baud rate set to 9600 to 19200 baud.

The last game in this directory WONDER.TSK is one I am not familiar with. It won't run because it links a FSL library from an earlier version of RSX. There is a .TKB file which can be used to task build it but it is also broken. Fixing it is not hard if you are familiar with RSX but if you are not edit the file to the following:

```
TYPE WONDER.TKB

WONDER,WONDER/MA=WONDER,UVT100OLD,LB:[1,1]F77FCS/LB
/
SUPLIB=FCSFSL:SV
ACTFIL=1
```

```
UNITS=5
//
```

Then, you can play:

```
TKB @WONDER.TKB
```

Johnny Billquist has many additional games running on his PDP-11 'MIM', accessible online over HECnet, and that is a great place to try them. If you set up networking, you might copy them over as well. One of my favorites is the famous "Leather Godesses of Phobos" Infocom adventure. For details: mim.update.uu.se

PDP-11 networking over wifi instead of ethernet

Work in progress: consider this section a stub; check the PiDP-11 forum for the latest developments.

The default setups (the boot.ini files) assume PDP-11 networking is done over eth0 (Ethernet), not wifi. There are, nevertheless, ways to use wifi, and given the idea of a PDP-11-in-the-living-room, that is desirable. It does require some Linux-foo. If in doubt, try Ethernet first. But if you want to do PDP-11 networking over wifi:

1. Upi Tamminen posted a how-to making use of Proxy ARP: groups.google.com/d/msg/pidp-11/aIFONtn2z4A/CMM7puvdCAAJ
2. John Forecast posted a how-to using vde tunneling. That requires a second Pi at the end of the tunnel: groups.google.com/d/msg/pidp-11/EfOin6Njln0/_a4Vqeu-BwAJ

Proxy ARP, a verified step-by-step following Upi Tamminen's recipe. Works for 211BSD, not RSX-11M+.

- `sudo nano /opt/pidp11/systems/211bsd/boot.ini`
  now change `attach xu eth0` into `attach xu tap:tap-simh0`
- `sudo apt-get install parprouted uml-utilities`
- `sudo nano /etc/sysctl.conf` , and paste the following at the bottom of the file:
  ```
  net.ipv4.conf.all.proxy_arp=1
  net.ipv4.ip_forward=1
  ```
- `sudo nano /etc/network/interfaces.d/tap` , and copy/paste the following in it:
  ```
  auto tap-simh0
  iface tap-simh0 inet manual
      pre-up tunctl -t tap-simh0 -u root
      up ip link set tap-simh0 up
      up /usr/sbin/parprouted wlan0 tap-simh0
      up /sbin/ip link set wlan0 promisc on
      post-down ip link set tap-simh0 down
      post-down tunctl -d tap-simh0

  auto tap-simh1
  iface tap-simh1 inet manual
  ```

```
pre-up tunctl -t tap-simh1 -u root
up ip link set tap-simh1 up
up /usr/sbin/parprouted wlan0 tap-simh1
up /sbin/ip link set wlan0 promisc on
post-down ip link set tap-simh1 down
post-down tunctl -d tap-simh1
```

- Pick an IP address in your network that is not used by anything else, something like 192.168.0.17. Use that for your configuration of 2.11BSD, just follow the setup as per the 211BSD Quick Tour that assumed you were using ethernet.

Powering a Pi 3 Model B Plus via the key switch

The Raspberry Pi 3 Model B Plus came out after the PiDP-11. It's great, and you should use it. But if you don't power the Pi through its on-board microUSB connector, but instead use the PiDP serial/power connector and the key switch as a hard on/off switch: the traces for 5V and GND on the board are just a bit too thin for the power-hungry Plus. If you use a 5.0V power supply and not the Official Pi Power Supply (which really is 5.35V…), then I recommend you solder power wires (5V, GND) from the pin header of the key switch straight to the GPIO connector. Apologies, I should have known to add more beef to the power trace from key switch to GPIO.

See this link for the importance of a good PS, also in terms of performance(!) for the Pi 3B+:
www.jeffgeerling.com/blog/2018/raspberry-pi-3-b-review-and-performance-comparison

Porting a new version of BlinkenBone to the PiDP

This section is just in case a new BlinkenBone release comes out and you find you cannot wait for a PiDP-11 version of it to be delivered... **There are only two minor code insertions**. And their only purpose is to reboot the system into another operating system/configuration using the SR switches or do a power-down from the front panel.

For the client (simh): Insert these two code snippets into src/02.3_simh/4.x+realcons/src/REALCONSrealcons_console_pdp11_70.c. See the current version to pinpoint the logical insertion point in that file.

(1) An additional include at the top of the file:

```
#include <string.h>
```

(2) In function realcons_console_pdp11_70_service, insert the grey lines:

```
if (_this->keyswitch_power->value_previous == 1) {
    //------------- PiDP inserted code
    FILE *bootfil;
    bootfil=fopen("/run/pidp11/tmpsimhcommand.txt", "r");
    fscanf(bootfil, "%[^\n]s", _this->realcons->simh_cmd_buffer);
    strcat(_this->realcons->simh_cmd_buffer, "\n");
    fclose(bootfil);
    //-------------
}
return SCPE_OK;
```

For the server (front panel driver): Just use the /src/11_pidp_server code from the current version, it'll work fine with any new BlinkenBone client.

Porting a new version of simh to BlinkenBone

Remember: simh->BlinkenBone->PiDP-11. If you want to port a new simh into BlinkenBone, that is non-trivial. Jörg Hoppe has kept up with changes and additions into simh, but doing it yourself will require a good mastery of the diffutils.

-- End of v0.2 draft version --