# Linux on the Psion 5MX/5MX-PRO HOWTO

**Last modified 9 July 2006. Changes.**

This document shows how to get a Psion 5MX or 5MX-PRO to boot up the Linux operating system, how to subsequently configure the system, and how to obtain, install and use applications suitable for the 5MX's limited resources. This document also contains notes on the remaining problems at the moment, and instructions on such things as compiling a custom kernel. The framebuffer and X windows are supported. Compactflash, touch screen, and serial ports are supported. Now even sound and recording are supported!

This HOWTO is a new rendering of the original 5MX HOWTO (http://linux-7110.sourceforge.net/howtos/series5mx/5MXHOWTO.htm) with corrections and updates. It isn't perfect, but it is better.

Other useful hints may be found on the Linux on netBook HOWTO (http://linux-7110.sourceforge.net/howtos/netbook_new/index.htm). The netBook and 5MX systems have considerable similarity.

PDF Version of this HOWTO (5MXHOWTO.pdf) (about a MB )

Please post information, questions, FAQ (with answers?) to the OpenPsion mail list (mailto:linux-7110-psion@lists.sourceforge.net).

## Table of Contents

# 1. Disclaimer

This HOWTO describes installing and configuring Linux on the Psion 5MX/5MXPro. There are no guarantees, and the procedures described here may turn your Psion into a brick/doorstop (well, not likely, but possible - a hard reset fixes most everything on a 5MX, besides outright hardware failures). Don't blame anybody if something goes wrong. That said, if you have any problems you can always post a query on the mailing list (send e-mail to: OpenPsion mail list (mailto:linux-7110-psion@lists.sourceforge.net) (linux-7110-psion@lists.sourceforge.net) ). Somebody there will likely be able to, but nobody is obligated to, assist you.

**Make sure you backup, backup, backup!!!**

# 2. Credits

This document has been passed down through several versions of HOWTOs. It was derived from a derivative of the original booting-howto written by Stephen Harris and additions by George Wright. The bulk of the credit for OpenPsion on the 5MX goes to those dedicated hackers who got the working kernel up and running and the various bits of hardware working. The credits list for the present document is as follows:

1. Wookey
2. Brian Dushaw
3. Tony Lindgren
4. Thomas A. de Ruiter
5. Nathan Catlow
6. George Wright
7. Stephen Harris

Most of these folks have moved on to other things in life, so there is little use trying to contact them - the mail list is the place to go.

Please see the openpsion website (http://www.openpsion.org/) (or more directly linux-7110.sourceforge.net (http://linux-7110.sourceforge.net/) for the more excellent people involved in the port.

# 3. Introduction

The OpenPsion project is an effort to port the widely used Linux operating system to the various Psion hand held computers, and the Series 5mx/5mxPRO pocket computers, in particular. Some familiarity with linux in general is probably essential for getting OpenPsion going, although at least one of the prepackaged distributions (e.g., "Distorted Woody") are fairly complete and ready to go.

## 3.1. Why Linux on the 5mx?

It is sometimes asked why one would want to install and run linux on one's 5mx? Everyone has different reasons, but I think the obvious reason is that if linux is an environment that one is used to and familiar with, then it is nice to have linux on the 5mx - a portable linux box as a PDA that meshes nicely with one's linux desktop machine. Another reason is that it is a real linux installation, and one can learn an awful lot about linux by working with the installation, both by installing the system, and by having a portable linux machine handy to play around with. Linux on the 5mx is a linux PDA with a keyboard that runs about 30 hours on two AA batteries (although some of the latest linux PDA are awfully nice and clearly outstrip the 5mx in performance in so many ways...).

I suppose also it is possible that a portable linux machine such as the 5mx could be used for purposes such as data collection or control in environments where a portable machine would be useful, particularly now that the real time clock for the 5mx is working and wake-up alarms can be set. The 5mx can be used to collect GPS data when in the field, for example.

Finally, linux is currently under active development, while the original EPOC environment has stagnated. Although many linux tools just won't run on the 5mx because of resource limitations: memory, video, cpu speed, disk size, which limits what linux can do on the 5mx, often suitable linux equivalents to EPOC functions can be found. And we certainly don't mean to complain about the EPOC operating system. EPOC is magnificent for many things, but limited in others. With linux on the 5mx, one obtains a dual-boot system, hence one gets the best of both worlds.

## 3.2. What Works, Generally

The team has ported the Linux 2.4.27 kernel to the Series 5mx/5mxPRO (2.4.32 is our development version), created ARLO (the boot loader), and created several initial ramdisks (initrd) for the project. In addition, a number of larger prepackaged distributions based on Debian Woody have been made available for installation on compactflash disks. All hardware devices on the 5mx are supported by kernel 2.4.27 or later to one degree or another.

At the moment, these distributions contain all the basic elements of any linux system: **bash, vi (vim), groff, less, man, perl, nawk, bc, dc, lynx (a web browser), ppp, telnet, ftp (ncftp), rsh, ssh, minicom, emacs ("zile", a smaller emacs-type editor is probably preferred), etc** . Starting with a basic system, it would be easy, though it would take a little time, to assemble your own custom system. All known systems are based on Debian Linux since this system has the best support for the ARM cpus. New debian arm packages (*_arm.deb, or a slimmer set of packages, *_arm.ipkg) can be downloaded and installed directly using **dpkg -i *_arm.deb** (or even *apt-get*), similarly for **ipkg**. Most people have been pleasantly surprised at how far along the whole OpenPsion project has come. OpenPsion is a fully functional, basic linux system. **X11** now works "out of the box." Most of the larger distributions, or most useful systems, are fairly sizeable and so require a compactflash card of 64MB or greater (256+ MB recommended) in size (compactflash cards of several GB in size are now available). Smaller systems based on initrd's can also be found, and they can be quite functional. The system seems to be quite stable - crashes or siezures, etc. are not often reported.

Larger packages such as **yorick or octave** - numerical computation packages vaguely like matlab - can also probably be installed except that this package required the X libraries, and so it would take up a great deal of space (I think yorick would work in console mode, if one had the space for it.) The **gnu compilers** can also be installed, and although they have been shown to work fine, they work rather slowly.

## 3.3. Psion Series 5mx/5mxPRO Specifications

The Series 5mx Pro differs from the 5mx only in having the operating system loaded into RAM and hence upgradable. The Pro series also offers memory options of 24 and 32MB. Ericsson also produced a rebadged version of the Series 5mx called the MC218.

```
Processor - 32-bit ARM 710T CPU (RISC based), running at 36.864 MHz
```

```
Internal Memory - 16, 24, or 32MB RAM

Internal Memory - 10 MB ROM

Removable Disk Type - Type I Compactflash (CF) Disks

   (disks of up to 1+ GB MB have been shown to work)

Display Resolution - 640x240 (Half-VGA)

Display Type - Monochrome touch-screen (16 shades)

Default OS - EPOC (32 bit, multitasking)

Serial Ports - Standard RS232 and SIR Infrared; up to 230400 baud.

Power - 2 AA batteries, backed up by a CR2032 lithium battery.

   Approx. 30 hrs of operation.  Optional 6V external power.

Screen Backlight

Sound - 1/2 W, 8 ohm loudspeaker  OKI MSM7717 and MSC1192 chipsets.

Microphone - electret with active gain control.

Keyboard - 53 key, QWERTY layout

Size - 172X89X24 mm

Weight - 350g (with 2 AA batteries)

Operating Temperature - 0 to 40 C
```

## 3.4. An OpenPsion Screenshot!

Other screenshots can be found HERE (http://linux-7110.sourceforge.net/screenshots.shtml).

# 4. How Does It Work?

**Bootloader** - The bootloader, ARLO, does for EPOC what Loadlin does for Dos. It releases the memory from EPOC, places the kernel and initrd in the appropriate places in memory, and then boots the Linux kernel. The initrd is not necessary if you have a linux system installed on compactflash. All EPOC information is lost from memory as linux takes over. Thus, if you don't have a compactflash disk, then you have to reinstall linux (ARLO, kernel, initrd) to EPOC the next time you reboot - and you will have lost any information you created in EPOC or linux if you did not backup.

**Kernel** - The kernel has patches applied to it, which makes it appropriate for ARM based systems, and then it also has a few more patches for the Psion hardware. The kernel port to the Series 5MX is quite functional now, with ongoing development. Support for the various components of the 5MX is essentially complete.

**Initrd** - This is a virtual file system image, which contains a miniature installation of Linux. These can range in size, but the more RAM used for the ramdisk will leave less memory available for the operating system. The kernel loads the initrd image as the root partition. For fully functional installations, installing a system on compactflash disk is preferred.

**Compactflash Disk** - The linux installation can be set up on a compactflash disk, which behave exactly as IDE disks. The kernel will boot up, find the root partition on the disk and proceed as in any other linux system. The use of compactflash disk also means that all information is saved on disk, so that the system (including ARLO and kernel) is stored to disk for any reboots.

**Dual Boot** - Because the EPOC system is stored in ROM, it is always available. A 5MX with linux is a dual boot system by default, so you can go between EPOC or linux as you like.

# 5. Preparation

What you will need:

1. A backup of your Psion's disks. A backup is very important because *everything* in your C: drive will be erased. Your D: drive likely be safe, but a backup is still worthwhile.
2. The ARLO for your model of Psion.
3. A kernel image (and kernel modules if you are using those)
4. Option 1: A compressed initrd (a very limited system)
5. Option 2: A prepackaged distribution to be installed on a compactflash drive (Debian dpkg based - Sarge, most likely). Requires ca. 256 MB or larger CF card.
6. Probably not needed: A paperclip or similar, thin, blunt item (to reset your machine if it hangs)

ARLO, the kernel image, a compressed initrd and a prepackaged distribution can be obtained from the Arlo section of the www.openpsion.org (https://sourceforge.net/project/showfiles.php?group_id=8846) File releases website (and elsewhere - see various links from the OpenLinux pages). ARLO, a kernel image and the initrd can often be found bundled together in a tarball.

# 6. Copying Files to the Psion and Making Partitions on

# the Compactflash Disk

## 6.1. Serial Ports, Minicom, and Comms

I call your attention to "irpsion5" which comes standard with the Linux irutils package. This utility lets you send files from the psion EPOC to a Linux box via IRDA.

The simplest way to copy a small file from the desktop computer to the psion is to execute "cat filename > /dev/ttyS0" on the desktop and log the output in comms on the psion. Similarly, you can send a file in comms (as ascii) and log the output on the desktop by "cat /dev/ttyS0 > filename" (^C when done). These assume that your psion is plugged into the /dev/ttyS0 serial port, of course.

[As an aside, note that you can print from EPOC on the psion to the serial port, and redirect the output from the serial port to the printer on the desktop by "cat /dev/ttyS0 > /dev/lp0". Start the redirect before printing, and ^C when done.]

All this assumes that you have your serial port properly set up for communication, which is the first thing to get working. The best way to go about setting this up is to use minicom on the desktop computer to verify the communication to comms on the psion - the right serial port to use (/dev/ttyS0 or /dev/ttyS1 usually) has to be determined and the baud rate has to be set right (115200 8N1 is usually best).

Once communication is working properly, you can send files back and forth between comms and minicom using xmodem, ymodem, zmodem, or ascii.

You can set the serial port baud rate manually using *stty -F /dev/ttyS0 115200* (see the man page for stty for other options).

## 6.2. PLP Tools

A better and more sophisticated way to get the files onto your Psion is to use Plp tools plptools (http://plptools.sourceforge.net/). [Personally, I've found plptools to be a little tempermental...] Compile and install these, and then use this to transfer the necessary files onto the Psion's CF through EPOC. The information here is for the most part copied from the plptools documentation. Please read the man pages for ncpd(8), plpftp(1), and plpnfsd(8); these are the programs that make up plptools. The daemons set up an nfs-like connection to the psion, so that the psion's "D:" drive will be mounted to /mnt/psion/d:, similarly for the "C:" and "Z:" drives.

**ncpd** is the daemon which handles the serial link to your Psion. It listens at port 7501 for local connections and provides basic PLP/NCP services for the other two programs. It auto- connects to the psion, even after unplugging/switching off therefore it can run always in background (if you have a spare serial-device). If supplied, the -e option will cause ncpd to automatically exit when the connection to the Psion is lost.

**plpftp** is a FTP-like program for manipulating files on the Psion.

**plpnfsd** is a daemon, which provides NFS-like access to your Psion. It automatically makes the psion's filesystems available below an NFS-mounted directory (default /mnt/psion). By default, plpnfsd is installed suid-root, so any normal user can start it and get's the mounted directory owned by himself. As this program is usually used on single-user machines, this does not hurt security. Like the others, this program auto-reconnects after a link-failure, so you can keep the psion mounted all the time, even when it is not connected. Due to Rudol Koenig's clever error-handling, you don't need to worry about blocked io-processes if the psion isn't available. You simply will get an "device not configured" error, when accessing a file on a previously connected psion which has been disconnected. After that, the mount-point will appear empty. As soon as the psion is connected again, the subdirectories will reappear. (possibly with a few secs delay)

For example to copy the arlo.sis file onto the Psion's C: drive (ramdisk) you type the following command: *cp arlo.sis /mnt/psion/c:*

There is also a KDE-based interface to the programs above. See the plptools website or the package documentation for more information.

You may have to manually set the baud rate of the serial port before starting up ncpd and plpnfsd to get it to work. Try:

```
stty -F /dev/ttyS0 1:0:800018b2:0:3:1c:7f:15:4:5:1:0:11:13:1a:0:12:f:17:16:0:0:2f:0:0:0:0:0:0:0:0:0:
```

to set the serial port speed, etc.

## 6.3. Partitioning the Compactflash using initrd

If you want to boot your system off a CF disk, you need to repartition it. These disks arrive from the store with a single Fat16 partition. You will need a reasonably-sized CF - 64MB or larger is preferred. A more effective way to repartition the compactflash disk is to plug it directly into a desktop or notebook computer, as described below. If you don't have this capability, then you can follow the procedure given next.

Begin by installing ARLO, a kernel and an initrd.gz onto the "C:" disk in EPOC (see the section Booting using ARLO). Then, boot into initrd.gz, using a kernel that supports CF disks. When this minimal linux system has booted up, execute "fdisk /dev/hda" to partition the drive and set the partition types. Partition your system so that you have a 4-6Mb Fat16 partition (or larger if you want a larger disk to use in EPOC) as /dev/hda1. Then make the rest of the disk (/dev/hda2) an ext2 partition for linux. For the Fat16 partition (/dev/hda1), use type 4, (FAT16 <32M) (assuming it's less than 32MB). Quit fdisk and format the ext2 filesystem ("mke2fs /dev/hda2"). You will probably have to boot into EPOC to format the "D:" disk (/dev/hda1) as an msdos filesytem (depending on how good your initrd system is...does it have mkdosfs?). When you have rebooted back into EPOC and formated the new, smaller "D:" drive, install ARLO, a kernel and an initrd on this Fat16 partition (the "D:" disk). Get ARLO properly set up, and now you don't have to re-install ARLO every time you reboot. You also have a larger disk to work with in linux.

To boot a CF based system, be sure to pass the appropriate option into the kernel (see Passing parameters into the kernel); this can now be set up using the ARLO GUI. Alternatively you can use the rdev command to alter the flags in your kernel (i.e., set the root device) before you "glue" it.

Additional pointers on installing a linux system without a desktop system can be found HERE (http://linux-7110.sourceforge.net/howtos/installation/poorandnocompactflashreaderHOWTO.html).

## 6.4. Partitioning, Formatting, and Copying to Compactflash using a Desktop Computer

The quickest and simplest way to install OpenPsion, or to copy large files to the system, is to use a (US$10-20) pcmcia or USB adapter to plug your compactflash disk directly into a notebook or desktop computer. The basic installation is in two steps (1) format the compactflash disk to have a 3.5 MB (or larger) dos filesystem and an ext2 filesystem of the remainder of the disk and (2) to unpack tarballs such as "disk1.tar.gz" (for EPOC, ARLO, initrd, and kernel) and "disk2.tar.gz" (for the linux system) into those respective filesystems. If you don't have such disks, you will have to bring over the various bits over the serial line...try to get the tarballs.

**STEP 1.** With the compactflash inserted in your desktop or notebook computer, access it with fdisk; BE VERY SURE YOU ARE USING THE RIGHT DEVICE - YOU CAN TAKE OUT YOUR DESKTOP LINUX SYSTEM!. I used "fdisk /dev/hde", but you may need to use "/dev/sda" etc. - whatever device your system recognizes as your compactflash. Delete the dos partition on the compactflash. Make the first primary partition to be 3.5 MB or larger in size (cylinder 1 to +3500K), and set its type to dos fat16 (I used type 4). Make the second partition to be the remainder of the compactflash and leave it as an ext2/linux partition (unless you have a large disk and want swap space or a second partition for some reason). Make the dos filesystem on /dev/hde1 ("mkdosfs /dev/hde1" ***see large capitalized letters above***) and the ext2 filesystem on /dev/hde2 ("mke2fs /dev/hde2" ***ditto***). Done with step 1!

**STEP 2.** Mount the dos filesystem - "mount -t vfat /dev/hde1 /flash" (I used /flash but you can use any convenient mount point). Unpack the first disk tar package into /flash: "tar xfz disk1.tar.gz -C /flash". Unmount the dos filesystem "umount /flash". Mount the ext2 filesystem - "mount -t ext2 /dev/hde2 /flash". Unpack the second linux system disk tar package into /flash - "tar xfz disk2.tar.gz -C /flash". Note you are better off using tar this way, rather than untarring the package to a filesystem and copying the filesystem over, because tar preserves the links, while "cp" will make duplicate files of the links and so use up much more space (but perhaps there is a smarter way of using "cp"). [You can also untar this filesystem in a convenient isolated directory, and browse through it and/or modify it to your own purposes before putting it on the compactflash disk]. Unmount the ext2 filesystem "umount /flash". Done with step 2!

If you are using modules with your kernel, unpack the modules files to, e.g., */lib/modules/2.4.27-vrs1-5mx2* if your kernel is version 2.4.27-vrs1-5mx2 (found by "uname -a"). You may have to execute "depmod -a" after boot up before you can get the modules properly working.

After waiting for the compactflash disk to properly unmount from the desktop (or execute "sync" before unmounting), take the disk out of the computer and put it in the psion. To boot up linux, start up ARLO (this used to be: highlight "arloapp.exe" on the "D:" disk and hit return, but now there is a nice ARLO GUI - see the section Booting using ARLO) and boot linux. You will likely get a LILO-like menu of flavors of systems to boot up. For example, Hit 1 for the normal multiuser system. Hit 2 for the normal single user system. Hit 3 for a system using the initrd.gz file, e.g. for system maintenance. Other options may be to boot with the console on the serial port, so that the psion can be accessed from e.g., minicom from the desktop. Normal use will probably be use compactflash disk and LCD. Without fuss or muss, when linux boots up you should get a login prompt - enter "root". In all likelihood no passwords required.

If you have large files or sets of files that you need to get onto the compactflash, you are probably better off plugging it into the notebook or desktop computer, mounting the disks there, and copying things directly. The compactflash disks are very fast; far faster than 115200 baud!

## 6.5. Using RSH/RCP and PPP Between the Desktop and OpenPsion

Once you have OpenPsion installed, your serial ports set up, and PPP working between the psion and the desktop computer, you can use rsh/rcp to copy things back and forth. Be sure rsh-server, rsh-clients and inetd or xinetd are installed. If you have set your system up to avoid using passwords, be sure the */etc/pam.d* directory is deleted after installing rsh. If "openpsion" is the hostname of your psion, then from the desktop *rcp filename openpsion:* will copy *filename* over to the psion. Similarly, *ssh* can be used, although it will be a little slower because of the encryption overhead.

See the documentation for rsh or ssh on configuration, and see the additional notes in the section on Serial Ports and PPP.

# 7. Compactflash Comments

Once formatted with a Fat16 partition for EPOC and an ext2 partition for linux, EPOC will see only the Fat16 partition (the D: drive). Linux will see and access both partitions, just as the case for Windoze; the /etc/fstab file gives the name of the Fat16 partition, of course. With such partitions, the psion effectively becomes a dual boot system. If you want to run EPOC applications as well, then be sure to format enough space (32MB?) to allow for saving those applications on the D: disk.

Disk sizes of over 1 GB have been used successfully in a 5MX. There is no known case of a compactflash disk failing to work because of its design, actually. There are occasional outright hardware failures, of course.

At one point I was told that I might save space using a JFFS2 filesystem, rather than ext2. However, I ran across an e-mail archive that had a message from the author of JFFS2 to the effect that compactflash cards are for all intents and purposes IDE devices, and ext2 was the way to go. Treat them like an ordinary IDE disk. Compactflash disks can be written on about 100,000 times before they give out, but these cards are also smart enough to distribute/rotate writing on them so as not to "wear out" particular spots. [If you do a little math you will find that it will take a long, long time to make enough writes to wear out a compactflash card.]

Compactflash and swap space. Because compactflash disks can be written on about 100,000 times before they give out, one generally does not want to use swap space, and I add the "noatime" option when mounting the disk to ensure that the access times are not constantly written to the disk. It would seem to be worthwhile making some swap space, if you had a large disk, that you could enable from time to time should you need it. On the other hand, if you really needed swap space you could just make a file for it.

Most compactflash disks should work in the psion, but drives of size greater than 1-2 GB are as yet untested. Also be aware that some disks work better than others, others are flakey, and some of them are faster than others; "your mileage will vary". Some cards won't work at all. Check out the DiskBench page (http://www.fortunecity.com/skyscraper/babbage/399/) (there are others around, e.g. for cameras and write time for photos) for a list of results and a utility to test the speed of your disk. It is not always true that you get what you pay for in this case (IMHO) - price seems to bear little resemblance to performance. It is worth searching the web for a few minutes for reviews of compactflash cards before purchasing. If you want to have a fairly complete system with X-windows and compilers, and still have disk space to work in, a 256 MB disk or greater is recommended. As of this writing (April 2006) disk of 1 GB size could be bought for around US$50.

It is worth getting a pcmcia compactflash reader (US$10) or other USB reader (US$20) for formatting and copying information directly to the compactflash card - these cards are as fast as UDMA33 IDE drives, so writing directly to them is WAY faster than trickling over the information on the 115200 baud serial line. Disks in pcmcia are recognized as IDE drives (/dev/hde), while disks in USB are recognized as SCSI disks (see your /var/log/messages file after you plug it into your notebook/desktop to see which disk it is).

Currently the compactflash card must be in the socket when you boot, and must not be removed while running. The current driver does not support hot-plugging the card.

I've been informed that the IBM microdrives do not fit into the Psion 5MX, nor do they work. You can read about the sad, if courageous, tale HERE (http://www.portal-pda.com/guides/microdrive/microdrive.html) [broken link...]. One hypothesis is that the microdrives do not work (after prying open your psion to cram it in) because the psions may not supply enough power to run them. Some lunatics have suggested powering them externally, but we'll leave that as a rainy day project... :)

You can execute "hdparm -u1 /dev/hda" which keeps the clock from slowing down with writes to the CompactFlash. The -u1 enables the unmaskirq option for the IDE driver, so the timer does not miss interrupts during the writes.

The Psion 5MX disk benchmarks vary considerably between compactflash brands, but are generally a little under 1 MB/s write speed. Testing with file sizes over 1 MB currently causes roughly one lost interrupt for each MB written on Psion on a Sandisk 96MB card, so that's why writes can be much lower. This may not happen on faster cards, and they have values for write speeds that are closer to laptop values. The cause of the lost interrupts is unknown, there are no available documents for ETNA.

Write times determined using the "time" command can be flawed because it severely underestimates the actual time of execution for technical reasons.

It develops that not all compactflash disks keep strict adherence to the compactflash standard. There have been reports that some compactflash cards are not recognized by one linux kernel version, but recognized by other linux kernel version. In particular, the 2.4.27 kernel seems not to recognize a few compactflash card types at boot up, while the 2.4.19 kernel recognizes those cards. The issue seems to lie in the IDE code, rather than the Psion 5MX kernel patch. The issue is presently a research topic, but if you find your new linux installation fails to boot because the compactflash card is not recognized, you might try a different type of compactflash card.

# 8. Booting using ARLO

## 8.1. Installing ARLO

You need to install ARLO (the boot loader - equivalent to LILO) onto your psion. To do this, visit the README file for ARLO. (http://www.yipton.demon.co.uk/arlo/latest/readme.html) The procedure has become fairly straightforward, so only a brief description of ARLO installation will be given here. The latest version of ARLO can be found http://www.yipton.net (http://www.yipton.net/), or more locally (and preferably) at Sourceforge Files (https://sourceforge.net/project/showfiles.php?group_id=8846). ARLO has a GUI that one starts from the applications panel of EPOC. It allows the configuration to be adjusted in a user-friendly manner.

ARLO can be installed to either the C: or D: drives as you select - it will work fine from either. However, if you install it on the C: drive, you will need to reinstall it each time you boot linux, because linux will use the C: drive as system memory.

You may need to set up a configuration file for ARLO if you use an older version of ARLO - see the documentation.

## 8.2. Getting a kernel

You will need to get a binary linux kernel; making sure you have the latest kernel is probably a good idea anyways. Obtaining an precompiled kernel is easiest. I would recommend obtaining one from the Files at Sourceforge (https://sourceforge.net/project/showfiles.php?group_id=8846). The kernels for the 5MX must be uncompressed. You will need the kernel type that goes with your model of 5MX (DE, UK, or US). Copy the *uncompressed kernel image* and *initrd.gz* files to the drive on the Psion that you installed ARLO as, e.g., *linux.image* and *initrd.gz*.

## 8.3. Getting the kernel booted from EPOC

At this stage you are ready to startup ARLO, set a few configurations, and boot into Linux. Double check you have good backups of you Psion and then start up ARLO. From ARLO you should be able to use various ARLO configurations to set the *linux.image*, *initrd.gz* (if you use an initrd.gz), and any options you want to pass to the kernel (e.g., *root=/dev/hda2*)- follow your nose or see the ARLO documentation. ARLO can be set to boot from any number of *initrd.gz* files or linux systems on disk.

**Loading the initial ramdisk (initrd).** You can easily load a ramdisk (the initrd.gz file) by setting this image in the ARLO configuration.

**Passing additional parameters into the kernel.** You may need to pass any other parameters into the kernel you need to type: *root=/dev/hda2* to set */dev/hda2* as the root filesystem. If you are using an initrd, you should not use this option.

A full description of kernel parameters can be found in /usr/src/linux/Documentation/kernel-parameters.txt of the kernel source tree.

**Running the kernel.** In ARLO, highlight the system type you want to boot and hit return; your psion should now boot into Linux! You should see a penguin that goes with the framebuffer. If the system appears faded, set the framebuffer depth to be 2bpp, rather than 4bpp; you can also try "Fn ." to increase the contrast some.

If it does a double bleep and then puts up the EPOC logo, something has gone wrong. Try double checking the *linux.image* file or your linux configuration. In desparation, try reinstalling ARLO.

## 8.4. Autobooting ARLO from reset

If you want to set up your psion to automatically boot into ARLO after a reset/reboot you must create a 'D:\system\data\wsini.ini' file on your CF Fat partition. The wsini.ini file must be a text file, and the last line should be changed to the following line:

```
STARTUP d:\arlo\arlosh.exe
```

This presumes you have the ARLO executable on the d: drive (compactflash) and the associated ARLO files. You can see the Psion's default \system\data\wsini.ini file by going to that directory on the Z: disk (ROM). To see the Z: disk, in EPOC hit "Ctrl, Shift, Tab". You will have to copy the wsini.ini file to either C: or D: disks, and then, e.g., import the text file into Word to edit it.

## 8.5. How do I get back to EPOC?

These days you should be able to just execute "shutdown -r now" to get the psion to reboot. If you did not implement a "wsini.ini" file to start up ARLO by default [see above], the machine will boot into EPOC. Alternatively, you can hit the equivalent of "control, alt, delete" by hitting "function, menu, delete" to cause a reboot.

If you are worried, type: *sync* a few times before starting the reboot, to be sure that stuff in memory is written to disk.

If your machine is hung for some reason and you need to reset it manually, then: Open the backup battery door and locate the small copper coloured circle near the battery, using a partly unfolded paperclip or similar, gently press in the copper coloured circle.

```
 ----------+
           |
   +-------+
   |o      |   <----- The small reset switch is silver/gold and resessed.
   |---    |            [about the size of a pin head]
   |    \  |
   |    |  |           Use the tip of a paperclip, or something like that
   |    /  |           to press it gently and when you hit the "On" button you
   |----   |           should here two happy beeps :) as EPOC starts up.
   | +----------------+
   | |                |
   | +----------------+
   +-------+
           |
           |
           |
 ----------+

(ASCII art courtesy of Chris Ross!)
```

Now close the backup battery door, and hit the Esc/on key. The machine should beep twice and then display the Psion 5MX splash screen. After a delay while EPOC reads its system out of ROM and reloads it into RAM, you will be back in the EPOC system. [While pressing the Esc/on key you may need to holding down both shift keys press to encourage EPOC to clear all memory, but this is probably not needed any more.]

More hints on reseting the machine can be found on the FAQ/reset (/faqs/resetting.htm) page.

Sometimes the reboot will fail because of an erroneous configuration on the compactflash disk - just remove the disk to get it to start EPOC, and then fix what's wrong on the disk (of course).

# 9. Using Your New System

Once your psion boots into linux, you have a basic, functional linux system. This section discusses some of the more useful things you can do, or get set up, as they might pertain to OpenPsion, but this document is not meant to be a tutorial on linux - in other words, if you have a specific problem, or task you want to do, you might consult the linux literature (man pages, HOWTO's, etc.) on the issue.

## 9.1. Mounting /proc and Compactflash Partitions

If your compactflash partitions are not mounted automatically, you can mount them:

```
mount -t msdos /dev/hda1 /mnt/disk1
mount -t ext2 /dev/hda2 /mnt/disk2
```

where /mnt/disk1,2 are used as examples of places where you might mount these partitions. If the above actions do not happen automatically for your system, you could set up those actions to happen at bootup using an /etc/rc.d file of some sort.

You can also do *dmesg* to see the system messages, or perhaps more primitively: *dmesg < /mnt/dmesg.txt*. *cat /var/log/messages* can also be useful and fun to take a look at, assuming system logging is set up.

## 9.2. The Keyboard and Setting Keys

The keyboard is same as for Psion 5.

**Keyboard features.** The Psion keyboard is very different from the standard PC keyboard. At the moment, the flavor of linux kernel you download sets the type of keyboard (UK, US, or DE). We use the following special key assignments (UK keyboard):

**Table 1. Psion and PC Keys**

| Psion: | Fn | Menu | Fn-T | Fn-Del | Menu-1 | Ctrl-Menu- | Ctrl-Menu-Del | Fn-Space | Fn-Esc |
|---|---|---|---|---|---|---|---|---|---|
| **PC** | AltGr | Alt | \| | ‘ (backtick) | Alt-F1 (VC1, etc.) | SysRq | Reboot | Backlight on/off | Power on/off |

Changing these default keys using the "loadkeys" and a keymap file might make these special keys inoperable (the newest kernel has supposedly fixed that). "dumpkeys > filename" will dump out a set of key mappings that you can take a look at. "loadkeys filename" will then load in that keymap (with whatever modifications you have made to it). You have trouble saving your special keys functionality, you can comment out or delete the lines for the special keys (e.g., the space key) to preserve their special functions (e.g., the backlight).

The Shift, Ctrl, Alt, and AltGr keys are "sticky" by default. They behave as illustrated in the following example:

**Table 2. Key Behavior**

| Sequence | Normal | Sticky | CONFIG_SMART_SLOCK |
|---|---|---|---|
| | | no | yes |
| Shift down, up, A | a | A | A |
| Shift down, up, down, up, A | a | a | a |
| Shift down, A, A Shift up | A A | A a | A A |

"Fn M" or "Fn ." will lighten or darken the screen as for EPOC (although there seem to be lingering problems with the screen colors (or video modes).

The psion will likely start several terminals that can be selected using "Menu 1", "Menu 2", etc. You can go back and forth between the various screens at will, and so multiprocess (unlike Windoze).

## 9.3. Setting the Font

You can change the font and font size using the "consolechars" program (assuming your system has this installed). (I find the default fontsize to be too small.) The fonts are in the "/usr/share/consolefonts" directory. For example,

```
consolechars -f default8x9
consolechars -f default8x16
consolechars -f lat4u-19
```

will change the font to increasingly larger sizes, and

```
consolechars -d
```

will change to the default font. Each console has to be set independently. You might set your favorite font in your *~/.profile* file.

## 9.4. Setting Framebuffer Depth

Use *fbset -a 2bpp*, *fbset -a 4bpp*, or *fbset -a mono* to set the framebuffer depth. In earlier kernel versions, the colormaps of the framebuffer still needed fixing, so that the system default is 4bpp, appeared to be faded. A work around was to change to 2bpp, but you'd be better off upgrading your kernel. (*lynx* seems to work better in mono.) To change the framebuffer depth, you must have the file */etc/fb.modes* configured. This file might consist of:

```
  #Video modes for Psion 5mx

   mode "default"
  # D: 3472.222 MHz, H: 5425.347 kHz, V: 22605.613 Hz
   geometry 640 240 640 240 4
   timings 288 0 0 0 0 0 0
   grayscale true
   endmode

   mode "4bpp"
 # D: 3472.222 MHz, H: 5425.347 kHz, V: 22605.613 Hz
   geometry 640 240 640 240 4
   timings 288 0 0 0 0 0 0
   grayscale true
   endmode

   mode "2bpp"
 # D: 3472.222 MHz, H: 5425.47 kHz, V: 22605.613 Hz
   geometry 640 240 640 240 2
   timings 288 0 0 0 0 0 0
   accel true
   grayscale false
   endmode

   mode "mono"
 # D: 3472.222 MHz, H: 5425.347 kHz, V: 22605.613 Hz
   geometry 640 240
   640 240 1
   timings 288 0 0 0 0 0 0
   accel true
   grayscale false
```

```
endmode
```

## 9.5. Pointers and Things that Work

**RESET.** The screen can be refreshed by typing "reset". A "reset" can sometimes be very effective at curing what ails your psion.

**BC/DC** *bc* is a simple line calculator type thing - try *man bc*. *dc* is similar, but more reverse polish like. Calculate pi to arbitrary many decimals; if that isn't useful, what is?

**EDITORS.** In the editor department, OpenPsion is still rather limited. *vi* (*vim*) certainly works, as does *zile*, a slimmed-down emacs clone (emacs itself could be installed, if you wanted to use the diskspace for it). However, the touchscreen is does not work for any of the console editors, so you can't point and click your way along with an editor. X is now supported supported, however, together with the touchscreen and all of the X applications, including more advanced editors.

**COMPILERS.** The *.deb files for the compilers gcc, g77 can be installed and they work o.k., although a little slowly. You need available disk space to install these.

**REBOOTING.** One can now easily reboot the psion without having to use the paper clip to press the secret button for a hard reset. "Ctrl Menu Del" or *shutdown -r now* will both probably work, and properly (maybe... at least it can be set up to work properly) shutdown the linux system. NOTE that you will lose everything from memory on the EPOC side of things - all of your set up parameters, serial port settings, the date and time, etc.

**USING LYNX.** *lynx*, a simple terminal-based web browser, seems to be set up to use colors which are not available for the psion. I tried to set things up in */etc/lynx.conf* so that the colors are black and white. You can also change the frame buffer color depth to "mono" using *fbset -a mono*. *lynx* may work better with mono. You can browse www.nytimes.com and download a package or two from ftp.debian.org. Also check out: The BBC News PDA Page (http://news.bbc.co.uk/low/english/pda/). Also check out the browsers that work in X.

**X.** The X window system can now be installed on the 5MX - see the section on X11 in this HOWTO. Installation of X brings with it all of the X applications.

**PRINTING** There is nothing to stop you from installing the lpr package, setting up the PPP network connection to the desktop computer (see below), and then setting up network printing through the desktop. This will require running the lpd printer daemon, and setting up the */etc/printcap* file. With this you can print from your psion to the printers that are connected (local or network) to your desktop computer.

However, a simpler way to print a file is: *dd if=filename | rsh desktop 'lpr -'*. Perhaps better for a 115200 baud connection: *gzip -c filename | rsh desktop 'gunzip -c | lpr -'* Or perhaps *gzip -c filename | rsh desktop 'gunzip -c | enscript --filter-stdin=-'* (and so forth...).

## 9.6. /etc Adjustments

Most of the configuration of linux is set up in the /etc directory. If you have disabled passwords, be sure the */etc/pam.d* directory does not exist, or you will have to use a password. Check out your */etc/rc.d* and */etc/inittab* files. Also *cron* files are there; *cron* works fine, assuming your clock is set correctly. You may also have a */etc/halt* file that gives to procedure for shutting down the system.

## 9.7. /proc Adjustments

There are a number of useful setting in the /proc directory tree. These allow you to set various screen or power configurations. The */proc/cpuinfo* file shows the cpu type. The directory */proc/psionw* has files particular to the 5MX:

```
backlight case contrast cpu lcd mains powerhook sleep state uart1 uart2
```

which can be used to list or set various states of the 5MX.

Some of these files can be used to set the degree or amount of things: The sleep file can be used to set the number of seconds of idle before the 5MX turns itself off: *echo 600 > /proc/psionw/sleep*. While *echo 60 > /proc/psionw/contrast* sets the contrast.

Some files can be used to turn things on or off: backlight or lcd can be set to 0 or 1 for "off" or "on", respectively.

While other files just report the state of things: *mains* is 0 or 1 depending on if the external power is connected or not. I suppose *uart1* (ttyAM1) and *uart2* (ttyAM0 - IR) are the same way - 0 or 1 if they are off or on; *case* is 1 if open, perhaps.

The *powerhook* file is a new addition to the kernel. A path to an executable can be entered into this file which will then be executed when the 5MX goes to or from sleep. The executable obtains a single flag, either 0-sleep, or 1-wakeup. This executable can be used for, e.g., checking the battery status or alarm settings. As an example,

```
echo "/etc/powerhook.sh" > /proc/psionw/powerhook
```

will enter the script powerhook.sh to be executed at sleep and wake up. This script might look like:

```
#!/bin/sh
# /etc/powerhook.sh
# $1=0 Power off
# $1=1 Power on
BATT_LOW_TRESHOLD=50

if [ $1 -eq 0 ]; then #Power off
  #Here is place to set /dev/rtc to wakeup

  #Real poweroff
  echo 0 > /proc/psionw/state
else #Power on
  #Battery check
  if [ -f /proc/apm ]; then
    P=`cut -d" " -f7 /proc/apm`
    P=${P%\%}
    if [ $P -le $BATT_LOW_TRESHOLD ]; then
      echo "Battery low $P%" | wall
    fi
  fi

  #Here is place to wakeup your remainder/calendar prog
fi
```

Which will check the battery status at wakeup and give a warning if the battery is low, and also run the script of your choice at wake up as well (such as play a sound to wake you up, or start the coffee pot (http://www.tldp.org/HOWTO/Coffee.html)).

The *cpu* file reports cpu frequency - I gather this was meant to set the cpu frequency (for over or underclocking...), but it doesn't seem to work.

Is there a setting for the time idle seconds before auto-off of lcd or backlight? I don't know...

The 2.4.27 and later kernels include a way to read battery status. Just *cat /proc/battery* to see the status (this has recently been moved to /proc/psionw). One could set up a script that can runs regularly to check the battery and give a console warning about an impending low battery.

Then there is /proc/sys/vm/laptop_mode which is usually set to 0. There are a number of webpages concerned with setting this to 1, and other options for power savings on laptops - is this of any use on a 5MX? Dunno.

The */proc/apm* file was added to the 5MX kernel to provide compatibility with existing tools. The plan may have been to provide the same output as i386 */proc/apm* to enable regular APM monitors to work on the Psion. However, since */proc/apm* really only shows voltage-level (used above for powerhook), some "smarter" energy handling code would be nice. I think this was discussed on the mailing list a few years ago. As shown above, the small script

```
#!/bin/sh
  P=`cut -d" " -f7 /proc/apm`
  P=${P%\%}
  echo "Battery level is:  $P%"
```

will display the battery level using the readings in /proc/apm.

## 9.8. Keeping Time Between Reboots

The resetting of the clock with every reboot is particularly annoying. With every reboot the psion's time get sets back to 1 January 1999 at 5 pm. There might be a way of writing the time to a file that would be written at shutdown and read at boot up by both the psion and linux; you might lose a few seconds with every reboot, but that would not be too bad. Work is in progress on this issue. You can read some notes on this subject, including a crude system for keeping track of the time when rebooting between EPOC and linux at this timekeeping page (http://staff.washington.edu/dushaw/psion/timekeeping). The page includes a couple of OPL programs for writing system time to a file in EPOC, as well as setting the system time from a file. There is also an awk script that will let you set the linux system time interactively, rather than try to remember the esoteric *date* format.

## 9.9. The Real Time Clock, /dev/rtc

With a kernel that supports the 5MX's real time clock, you can access the hardware clock and its alarm. The alarm is useful because it lets you set a time when the 5MX will wake up from sleep and do something like sound an alarm (see the section on sounds). The traditional way to access the hardware clock's time is using the utility *hwclock*. **N.B.:** the Debian Sarge hwclock utility doesn't seem work, while the old Woody version does. The download "alarms.tgz" contains this Woody binary.

One simple C program for setting the alarm is called *rtc.c* (a quick and dirty, but functional, cut and paste job), which you can download from: alarms.tgz (includes sound samples, source code, ARM binary for rtc.c, and alarm scripts). The program sets the alarm and then waits for the alarm before completing. So you can have a script:

```
  #!/bin/sh
  rtc hour:minute  [or "rtc +hour:minute" to have an alarm hour:minute later than the present time.]
  play a sound
```

And so get an alarm to go off at hour:minute. And yes, it wakes up the 5MX if you turn it off! It is best to synchronize hardware and system clocks to the same time, and to the same UTC or local time (the real time clock is set to UTC by default).

```
  hwclock --localtime --systohc
```

will synchronize the two clocks. To see the settings of the real time clock:

```
  cat /proc/driver/rtc
```

A similar utility to rtc.c is available in the Debian apmd package ("apmsleep"), but that comes with a load of apm baggage. I couldn't find a simple utility for setting an alarm.

One issue with the rtc is that it does not account for the change in date when midnight is crossed. So the alarm may not sound if midnight occurs between the time the alarm is set and the expected wakeup time. A workaround is to first set a wakeup just before midnight, and then reset the alarm to the correct wakeup time after midnight is passed. The *alarms.tgz* tarball has a script "setalarm" that performs these steps.

## 9.10. Installing New Packages

New *arm.deb or *.ipk packages can be installed on your psion, depending on if your system is based on *dpkg* or *ipkg*. So if you want to install the package *mdate_1.0.1-3.deb* [a utility for reporting Mayan dates] first get the package from debian.org, making sure it is an ARM binary, copy it to the psion, and execute *dpkg -i mdate_1.0.1-3.deb*. Remove it by *dpkg -r mdate*. In some cases, the distribution builder has hacked out large elements of the dpkg system to better compact the distribution, and *dpkg* will complain, sometimes failing to install things altogether, or installing the binaries but failing to configure things. In the former case, you can manually make whatever file or directories are missing and try to procede that way (i.e., put on your hacking cap.) In the latter case, you have probably successfully installed the binaries, so what are you complaining about? *dpkg* will also install a great deal of metadata - you may want to find this stuff and delete it to save on disk space.

Distributions based on "Debian Woody" are also "testing" distributions, and so some installations will fail because elements of the distribution (e.g., the libc library) are old. In that case, you have to go about installing or updating the missing elements. Updating can be like a plate of spaghetti. But it also may be a piece of cake.

In most cases, however, new packages will install with little trouble.

Installation of packages with apt-get will work, if a swap space of size ca. 64MB is temporarily set up for additional memory. It is very slow, but it works o.k.

One problem is that dpkg tries to read the file */var/lib/dpkg/available* completely into memory. Because the 5MX has only 16 MB memory and /var/lib/dpkg/available is about 11MB, dpkg doesn't have enough memory to do this job. Swap space can resolve this issue somewhat, but it is very slow - sometimes resulting in mere disk thrashing. One solution to the problem is:

```
mv /var/lib/dpkg/available /var/lib/dpkg/save.available
touch /var/lib/dpkg/available
```

The available file contains the packages available from Debian, so removing it disables apt-get. The status file is more important, and contains the status of your system, and cannot be rebuilt. The available file can be cleared and rebuilt with:

```
dpkg --clear-avail
apt-get update
```

So, putting it altogether, perhaps the optimal procedure for installing a package is to use the "-d" option for apt-get, which is to download the package and its dependencies only. Then, after downloading is complete, rename the */var/lib/dpkg/available* file to something else, and create an empty replacement file as above. Then, cd to */var/cache/apt/archives/*, and execute *dpkg -i *.deb* to install the new downloaded *.debs. Then, clean up the downloaded files with *apt-get clean* and restore the *available* file. A simple bash script, called, e.g., *dpkginstall*, to perform these steps (copy /var/lib/dpkg/available to /var/lib/dpkg/save.available before trying!):

```
#!/bin/sh
echo "Turning on swap"
swapon -a

echo "Getting the packages"
apt-get -d -y install $1
sync

echo "Shuffling available"
rm /var/lib/dpkg/available
touch /var/lib/dpkg/available
sync

cd /var/cache/apt/archives/
echo "Installing packages"
dpkg -i *.deb
sync

echo "Turning off swap"
swapoff -a

echo "Cleaning packages"
apt-get clean
sync

echo "Copying back available"
cp /var/lib/dpkg/save.available /var/lib/dpkg/available
sync
```

Execute the script with *dpkginstall foo* to install package foo. Worked well for me.

## 10. Serial Connections and PPP

Both hardwire and infrared serial ports work fine on the 5MX in linux. This section describes how to use them, and how to set up PPP connections with them.

Some distributions come with gsmlib, that allows you to send SMS and manipulate your cell phone phone book. For more information about gsmlib see: http://www.pxh.de/fs/gsmlib/.

The 5MX serial ports are /dev/ttyAM0 (IRDA) and /dev/ttyAM1 (hard wire). If these special files do not exist on your system then:

```
mknod /dev/ttyAM0 c 204 16    [ IRDA serial port ]
mknod /dev/ttyAM1 c 204 17    [ regular serial port ]
```

will make them.

## 10.1. Serial Connection: Wire

Connections to both a modem using PPP, or to the desktop computer using minicom, seem to work fine (115200 baud). The serial port is */dev/ttyAM1*; it works just like any other serial port. Use minicom to debug serial connections.

In order to make it easy to connect the 5MX to the desktop, the Psion serial cable is a null-modem cable. This means that to connect other serial devices (GPS, modems), a null-modem adapter is required. See the netBook HOWTO for a discussion.

Although the 5MX is specified to support speeds to 115200 baud, in reality speeds of 230400 baud can be obtained. However, the serial ports on most computer motherboards only support 115200 baud. Generally, an external serial port such as USB or PCMCIA must be used to obtain a 230400 baud connection to the 5MX. Fast Connection to Windows (http://linux-7110.sourceforge.net/oldwiki/5mx/FastSerialConnectionToWindows.shtml) describes how to do it with windoze.

## 10.2. Serial Connection: Infrared/IRDA

The IR serial port also works; one can communicate back and forth via minicom, and set up a PPP connection over this port to e.g., a notebook computer. You can then rcp files back and forth out of thin air! Woohoo!

With a Debian distribution, you can just install the irda-utils package to get all you need for irda. Edit the /etc/default/irda-utils configuration file (enable irattach, enable discovery, use device /dev/ttyAM0), and then execute */etc/init.d/irda-utils start* and */etc/init.d/irda-utils stop* to start and stop irda.

More primitively, to enable the infrared connection, one executes *irattach /dev/ttyAM0 -s 1* on the psion, and *irattach /dev/ttyS1 -s 1* on the other machine (if ttyS1 was recognized at boot up of the notebook computer as the IR device). There is also likely to be a */etc/init.d/irda* startup script on the notebook computer that you could use. The *-s 1* flag enables system logging.

Even though ttyAM0 or ttyS1 are used to make the IR attachment, */dev/ircomm0* is still the serial port to use on either notebook computer or psion. The command *irdadump*, which is a standard tool, can help in debugging the connection - it will show, at the lowest level, when or if the remote IR signal is picked up or not.

NOTE: the use of IRDA is meant to consume a considerable amount of power; turn it off by killing the irattach process when it is not in use.

For real amusement, it may be possible that the IR and normal serial connections can be bound together as one in order to double the bandwidth of the serial connection. See the IR HOWTO (and let us know if you got this to work!).

## 10.3. Running Getty's

The serial connections, PPP, and rsh involve running a getty on the serial ports, either on the notebook or on the psion. *getty 115200 /dev/ttyAM1* or *getty 115200 /dev/ircomm0* on the psion, or *agetty 115200 /dev/ttyS0* or *agetty 115200 /dev/ircomm0* on the remote computer. These getty's are probably best started up in the */etc/inittab* file, because they need to be "respawned". You can restart init so that it rereads this file without rebooting by *init q*.

## 10.4. Setting up PPP

This subsection contains two sub-subsections on

**1. Connecting the Psion to the Desktop.**

**2. Connecting the Psion to the Internet Using an External Modem.**

Refer to the Linux PPP howto: http://www.linux.org/docs/ldp/howto/PPP-HOWTO/index.html (http://www.linux.org/docs/ldp/howto/PPP-HOWTO/index.html).

### 10.4.1. Connecting the Psion to the Desktop

As an example, PPP can be configured with telnetd. However, *rsh (ssh)* may be preferable to telnet, because then you can efficiently copy things between the psion and desktop using *rcp (ssh)*. Install the rsh (ssh) client and servers if you want *rsh (ssh)*; *rsh* is somewhat preferable to *ssh* for psion-computer connections because it does not have the overhead of ssh encryption. [NOTE: installing packages with shells, daemons and so on with dpkg, will sometimes make a directory /etc/pam.d. If you have disabled passwords on your system, you must delete this directory; if it exists you will not be able to login!! If you forget and find yourself unable to login, boot to single user mode and delete the pam.d directory.]

To connect your Psion to your computer:

Have something like this for the */etc/ppp/options* file on the Psion:

```
-detach
defaultroute
noauth
nocrtscts
lock
lcp-echo-interval 5
lcp-echo-failure 3
/dev/ttyAM1
115200
```

(or use */dev/ircomm0* for IRDA)

Then add the ppp user to your Psion's */etc/passwd* file:

```
echo "ppp:*:101:101:PPP User:/etc/ppp:/usr/sbin/pppd" >> /etc/passwd
```

And make sure pppd is executable for user ppp:

```
chmod a+x /etc/sbin/pppd
```

To get telnetd running, add the required entry to the */etc/passwd* file:

```
echo "telnetd:*:101:101::/usr/lib/telnetd:/bin/false" >> /etc/passwd
```

Then add the telnetd entry to the */etc/inetd.conf* file:

```
echo "telnet stream tcp nowait telnetd.telnetd  /usr/sbin/tcpd  \
               /usr/sbin/in.telnetd" >> /etc/inetd.conf
```

You will need to run a getty on e.g., */dev/ttyAM1* to enable logins to the Psion by telnet or ppp. The getty gives you the login prompt when you connect over the serial port. The command *getty 115200 /dev/ttyAM1* - or */dev/ircomm0* - will start a getty on the serial port. It may be preferrable to start the getty on the serial port in the */etc/inittab* file.

Finally add your computer's PPP IP address to */etc/hosts* on the Psion to make telnet login happen faster:

```
echo "# This makes the telnet login to psion faster
 192.168.1.100   gateway" >> /etc/hosts
```

On your desktop computer have something like this for a */etc/ppp/peers/psion* file:

```
-detach
noauth
nocrtscts
lock
local
connect '/usr/sbin/chat -v -t3 ogin--ogin: ppp'
/dev/ttyS0
115200
192.168.1.100:192.168.1.101
```

Then just type *pppd call psion* on your computer to connect to Psion. [You may prefer to start up the ppp connection from the psion, rather than from the desktop computer - but that might not be a security risk you want to take.]

To access the internet from your Psion through the desktop computer, you need to add routing, masquerading, and NAT [I don't know what these are either...but it works, so who am I to complain?] to your desktop:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j MASQUERADE
```

then you can browse the internet from your psion using *dillo* or *lynx*. Be aware that your desktop kernel must be compiled to support these things; default RedHat kernels support this. You may also have to disable, or otherwise configure, your firewall for this to work properly.

### 10.4.2. Connecting the Psion to the Internet Using an External Modem

Either a standard external modem (with null-modem adapter) or an external infrared modem (e.g., Psion's nifty external travel modem - the Diamond Mako travel modems are apparently identical to the Psion travel modems) will work for using your psion to connect to the internet over the telephone lines. You might be able to pick up an infrared

Psion (or Diamond Mako) travel modem from ebay.com for cheap; it works just fine in OpenPsion with very little fuss once IRDA is working - relink */dev/modem* to */dev/ircomm0* and use baud rate 115200.

With its installation via dpkg (or ipkg?), PPP is pretty much all set up and ready to go with an external telephone modem. Edit the files */etc/chatscripts/provider*, */etc/ppp/peers/provider*, and */etc/ppp/pap-secrets* to include your own ppp information, or just run the script *pppconfig*, which should come with PPP. PPP is started and stopped using *pon* and *poff*. (I made a script "ppp-on" that starts *pon*, dumps its messages to */dev/null*, and puts it in the background.) You may also need to modify the */etc/resolv.conf* file, to give your nameserver IP numbers (but this is unlikely).

# 11. Using Sound

The newer linux kernels for the 5MX now support sound quite well, including recording and minimal mixer support. It is unknown if the sound drivers will work as a module.

## 11.1. Working Features

- Plays recordings made in EPOC without any distortion: *cat soundfile* > */dev/audio*.
- Plays 8 kHz 16bit PCM files: *cat soundfile.wav* > */dev/dsp*.
- Microphone works with /dev/dsp and /dev/audio.
- Utilities play and rec from sox package work.

Installation of the sox package will give you the two utilities "play" and "rec". However, play produces nasty clicks occasionally with large sound files, which is likely an issue with play. The more primitive *cat foo* > */dev/dsp* plays sound without any clicks. Different buffer sizes inside the sound code have been tried to alleviate the clicks. Currently the buffer size is 4kB, dynamically allocated only when needed, which seems to work best.

Sound files native to the psion can be played - they are "*.al" files, so that "play -t al psionsoundfile" will play them. So, you can record sound under EPOC, and then play them back under OpenPsion without trouble. You can now also record sound under linux with "rec foo.al", which will make a recording in file "foo.al" that will play back without distortion.

## 11.2. Devices

/dev/dsp: set to 8000Hz 16bit signed linear PCM, mono.

/dev/audio: set to 8000Hz A-Law, mono, which is a format used by EPOC.

/dev/mixer: two channels are supported - master volume and microphone volume. Microphone can be a recording source. Either of the mixer utilities aumix, cpm or gom will work (cpm only on virtual terminal - it needs 80x25).

PCM and A-law are the only supported formats. 5mx hardware is not capable of changing sample rate, doing stereo, or doing sensurround. Nor can you plug in a sub-woofer.

The external buttons for sound on the 5MX are defined as keys F13, F14, and F15. These keys can therefore be tied to the appropriate sound functions using, e.g., keylaunch. In fluxbox, you can define these keys in the .fluxbox/keys file to be:

```
None F13 :ExecCommand rec /tmp/recording.al
None F14 :ExecCommand killall -2 sox
None F15 :ExecCommand play /tmp/recording.al
```

So that you can hit the record button to record, the stop button to stop recording, the play button to play the recording, and then again the stop button to stop playing.

## 11.3. Setting an Alarm

The openpsion kernel can be put in sleep mode and having an, e.g., daemon running (e.g., an ipaqalarm/uschedule (http://linux-7110.sourceforge.net/files/Software/Utility/uschedule_ipaqalarm/) (taken from ipaq, where it is no longer available) system) that will cause the psion to wake up and play a sound, e.g., for a morning alarm. A daemon is not actually needed to wake up the 5MX, the alarm for the real time clock has to be set. A utility for setting the real time clock alarm (with source code), and some example sounds in Psion format, can be downloaded here: alarms.tgz. See the section on the real time clock in this HOWTO.

A script, called e.g.,*setalarm*, that can be used to sound an alarm and then keep sounding the alarm until a key is pressed is setalarm. Execute with *setalarm 8:30* to set an alarm for 8:30. As you can see after 20 plays of the sound file, the script gives up and assumes that either nobody is around (so don't keep playing sound until the batteries are dead), or you REALLY want to sleep in. One of the difficulties with setting an alarm like this is the so-called "midnight barrier", where if the time of the alarm is earlier than the current time the alarm will not sound - the rtc does not account for the change in day. The work around in the "setalarm" script is to wake up just before midnight, wait for midnight to pass, and then set the alarm for the correct wakeup time. Surprisingly difficult to debug.

## 11.4. Sound Details

The sound code does not use floating point, which was one suggestion as to why the sound driver could not handle *.wav files. Rather, the sound chip on 5mx expects A-law encoding, and the play utility (from sox) for some reason cannot recode linear PCM to A-law on the fly. So, to make the sound driver half-usable, 16-bit 8-kHz PCM mode and recoding procedures have been added, which seem to work fine.

The 5mx hardware (OKI MSM7717 and MSC1192) is not capable of adjusting the sound level, so to mimic EPOC functionality the samples are recalculated. Now, the default volume level of 50 means no adjustment, and samples are passed to codec as they were received.

# 12. Installing PicoGUI Windowing

picoGUI (http://www.picogui.org/) is a space-efficient windowing system developed for PDA's. It is rather limited in its applications, however, and its development seems to have ended.

You can find out about installing picoGUI on the Psion 5MX from Tader's Psion 5mx Linux Pages (http://thomas.de-ruiter.cx/projects/psion/). You can find some screen shots there as well.

Grab the binaries (and an actual distribution tarball) from: http://thomas.de-ruiter.cx/projects/psion/files/PicoGUI/ There are two subdirs:

```
/etc
/usr
```

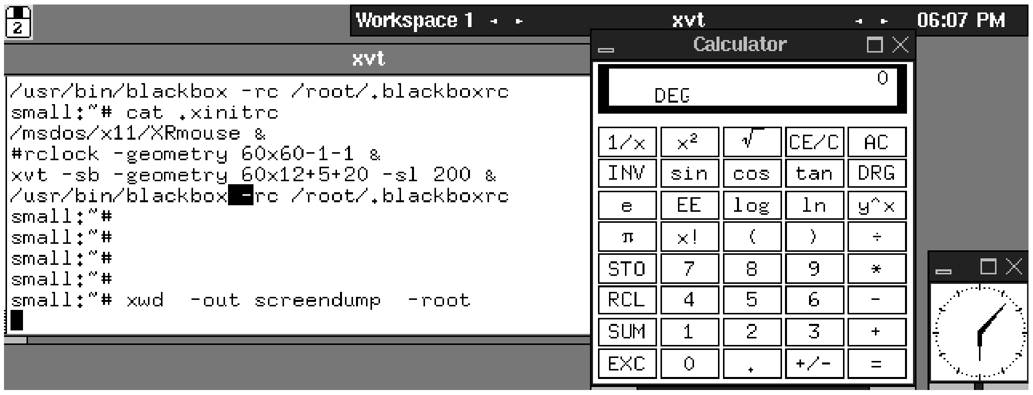
copy both to the ext2 partition. Then type *pgserver*. (Note: You get a crappy error message if you've changed the color depth to 2bpp with *fbset -a 2bpp*.)

There is a binary package for the Zaurus available on: http://pgui.sf.net which is for the ARM as well. Maybe /etc/pgserver.conf needs a little tweeking. But that should work as well.

# 13. Installing X Windows

## 13.1. An X on OpenPsion Screenshot!

X windows can be installed on the 5MX. This involves three basic steps: (1) install the relevant *.deb files (*.ipk files are as yet untested), (2) get the Xipaq X server that has been specially configured for the 5MX, and (3) configure a window manager. X windows is a fairly large overgrown package, but the number of applications designed to run under X windows is staggering. Note: The 5MX Xipaq server seems to be available only as a binary; the original patches to the X source code for the 5MX seem to have been lost. If you have information as to the wherabouts of that patch, please post to the OpenPsion mail list.



```
/usr/bin/blackbox -rc /root/.blackboxrc
small:~# cat .xinitrc
/msdos/x11/XRmouse &
#rclock -geometry 60x60-1-1 &
xvt -sb -geometry 60x12+5+20 -sl 200 &
/usr/bin/blackbox -rc /root/.blackboxrc
small:~#
small:~#
small:~#
small:~#
small:~# xwd  -out screendump  -root
```

## 13.2. Prerequisites

You will need to have a compactflash with something like 80 MB of free space available for the linux distribution with X windows. After installation, a great deal of disk space can be freed up by the deletion of extraneous fonts, binaries, and other meta data. Without doing any of this clean up, my 96MB linux partition was left with 12MB of free space.

Until the ipkg system gets sorted out, you will also need a distribution based on dpkg.

## 13.3. Packages to Install

The following packages and binaries, or their more modern versions, need to be installed: (The *.deb packages can be obtained from the distribution packages (http://www.debian.org/distrib/packages) site at debian.org. Be sure to arm binaries, not i386 binaries.)

```
cpp_2.95.4-9_arm.deb
libncurses5_5.2.20020112a-5_arm.deb
libdps1_4.1.0-14_arm.deb
libfreetype6_2.0.9-1_arm.deb
libglib1.2_1.2.10-4_arm.deb
libutahglx1_0.0-cvs-20010702-3_arm.deb
libxaw7_4.1.0-14_arm.deb
xfree86-common_4.1.0-14_all.deb
xbase-clients_4.1.0-14_arm.deb
xlibs_4.1.0-14_arm.deb
zlib1g_1.1.4-1_arm.deb
xfs_4.1.0-14_arm.deb  (optional)
xserver-common_4.1.0-14_arm.deb (optional)
```

The X fonts are slightly optional:

```
xutils_4.1.0-14_arm.deb
xfonts-base_4.1.0-14_all.deb
xfonts-scalable_4.1.0-14_all.deb
xfonts-75dpi_4.1.0-14_all.deb
```

Bearing in mind the severe memory constraints, X terminal emulators to consider are:

```
xvt_2.1-14_arm.deb
rxvt_2.6.4-3_arm.deb
xterm_4.1.0-14_arm.deb (a memory hog)
```

Of these xvt is preferred, because it uses much less memory.

In addition, you will need to download the Xipaq X server from the "Sourceforge downloads" (http://prdownloads.sourceforge.net/linux-7110/Xipaq.tar.gz?download) download site from OpenPsion. This tarball contains:

```
Xipaq
Xmodmap
```

There are several approaches to using the mouse functions. See the section below on Mouse Issues.

You will also need a Window manager for X. Here we will discuss the Blackbox window manager, *blackbox_0.62.1-1_arm.deb*. The Windowmaker manager might be a little better (see screenshot below), and there are other managers as well.

Install the above *.deb packages, and copy the Xipaq server to */usr/X11R6/bin/X*.

Before starting X, first make sure you are in 4bpp by *fbset -a 4bpp*, and be sure the touch panel device "/dev/tpanel" exists (*mknod /dev/tpanel c 10 11*). Then execute *startx*! If you are not in 4bpp, you will get a segmentation fault. You can start X more simply by just *exec xinit* which will save a little memory (*startx* is a script, so it keeps a shell open while X is running.)

You can start X up with the screen rotated. Start the xserver with:

```
Xipaq -screen 640x240@$ROTATION
```

where for values of $ROTATION:

```
($ROTATION <45)||( $ROTATION>=315) screenrotation = 0°
($ROTATION <135)&&( $ROTATION>=45) screenrotation = 90°
($ROTATION <=225)&&( $ROTATION>=135) screenrotation = 180°
($ROTATION <=315)&&( $ROTATION>=225) screenrotation = 270°
```

A rotation of 90° can improve the useability of some applications as long the required width is less than 240 pixels. [No, setting the resolution to something bigger (like 640x480 or 1280x1024) doesn't do virtual resolution (nor does it increases your screensize)]. You can set up a small script called /usr/bin/X11/Xrot containing the lines:

```
  #!/bin/sh
  /usr/bin/X11/Xipaq -screen 640x240@90
```

(and *chmod 755 Xrot*). Then, the rotated X can be started with *xinit -- /usr/bin/X11/Xrot*.

## 13.4. Configuring the Blackbox Window Manager

There are three main configuration files that need to be set up to run the Blackbox Window Manager: (1) .xinitrc, (2) .blackboxrc, and (3) ~/.blackbox/menu. These are discussed in turn.

**~/.xinitrc**

```
/msdos/x11/XRmouse &
xrdb ~/.Xdefaults
#rclock -geometry 60x60-1-1 &
xvt -sb -geometry 60x12+5+20 -sl 200 &
/usr/bin/blackbox -rc /root/.blackboxrc
```

*startx*, in the end, looks to this file to start up the X applications. The above file ends with blackbox until that process terminates (i.e. when you are done with your X session.) The rclock application is commented out as an example of things you can do to conserve memory.

**~/.blackboxrc**

```
session.screen0.slit.placement: CenterRight
session.screen0.slit.direction: Vertical
session.screen0.slit.onTop: False
session.screen0.slit.autoHide: False
session.screen0.toolbar.onTop: False
session.screen0.toolbar.autoHide: False
session.screen0.toolbar.placement: TopRight
session.screen0.toolbar.widthPercent: 66
session.screen0.workspaces: 2
session.screen0.focusLastWindow: False
session.screen0.edgeSnapThreshold: 0
session.screen0.rowPlacementDirection: LeftToRight
session.screen0.focusNewWindows: False
session.screen0.windowPlacement: RowSmartPlacement
session.screen0.focusModel: SloppyFocus
session.screen0.workspaceNames: Workspace 1,Workspace 2
session.screen0.strftimeFormat: %I:%M %p
session.screen0.colPlacementDirection: TopToBottom
session.screen0.fullMaximization: False
session.cacheLife: 5
session.colorsPerChannel: 4
session.opaqueMove: False
session.imageDither: True
session.menuFile: /root/.blackbox/menu
session.styleFile: /usr/share/blackbox/styles/Minimal
session.autoRaiseDelay: 400
session.cacheMax: 200
session.doubleClickInterval: 250
```

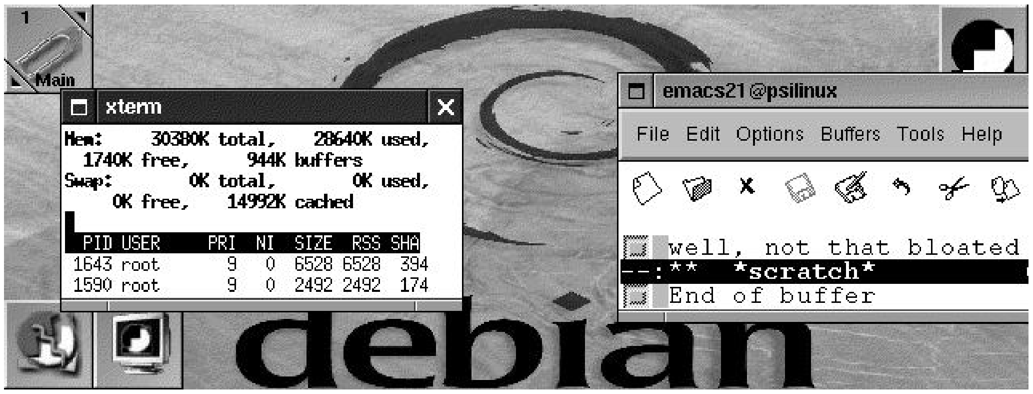
The main elements here are the menu file (discussed next) and the styleFile. The styleFile sets the colors and contrast - the "Mimimal" style seemed to work well for the psion, but you can try other styles.

**~/.blackbox/menu**

```
#Blackbox menu file
[begin] (Blackbox 0.62.1pre0)
[exec] (xvt) {/usr/bin/xvt -sb -geometry 60x12-10-10 -sl 200}
[exec] (rxvt) {/usr/bin/rxvt -sb -geometry 60x14-10-10 -sr -sl 200}
[exec] (xedit) {xedit -geometry 500x200-5-5 }
[exec] (xcalc) {xcalc -geometry 200x200-15-15 }
[exec] (rclock) {rclock -geometry 60x60-1-1 }
[submenu] (Window Manager)
   [exec] (Edit Menus) {xedit -geometry 500x200-5-5 ~/.blackbox/menu}
   [config]   (Config Options)
   [reconfig] (Reconfigure)
   [restart] (Restart)
   [submenu] (Styles) {Which Style?}
 [styles menu] (Blackbox Styles) {/usr/share/blackbox/styles}
   [end]
[end]
[exit] (Exit)
[end]
[end]
#end of menu file
```

It is obvious how one would go about adding additional applications to the menu. The "style"'s file gives a list of all of the styles that come with the blackbox package (e.g., "Minimal").

Here is a screenshot of the Windowmaker Manager, an alternative manager. Emacs21 is also shown running. This manager has a number of advantages over Blackbox.

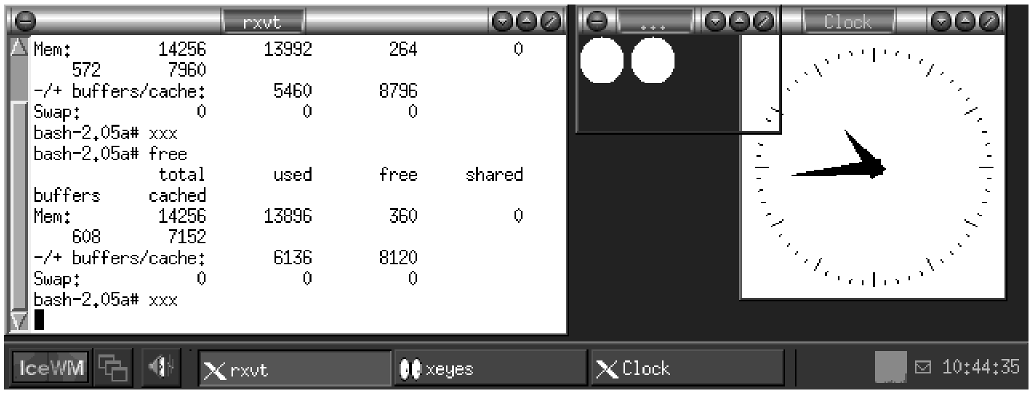


```
Mem:     30380K total,    28640K used,
  1740K free,      944K buffers
Swap:        0K total,        0K used,
     0K free,    14992K cached

  PID USER     PRI  NI  SIZE  RSS SHA
 1643 root       9   0  6528 6528  394
 1590 root       9   0  2492 2492  174
```

```
well, not that bloated
--:**  *scratch*
End of buffer
```

Here is a screenshot of the IceWM window manager, another lightweight manager.



```
Mem:         14256      13992        264          0
    572       7960
-/+ buffers/cache:       5460       8796
Swap:            0          0          0
bash-2.05a# xxx
bash-2.05a# free
             total       used       free     shared
buffers     cached
Mem:         14256      13896        360          0
    608       7152
-/+ buffers/cache:       6136       8120
Swap:            0          0          0
bash-2.05a# xxx
```

## 13.5. Using the extra-screen icons

It is possible to use the toolbar of the 5MX screen in X windows using a utility call xpsitouch (http://www.muru.com/linux/psion/xpsitouch/), an LCD toolbar handler. The README (http://www.muru.com/linux/psion/xpsitouch/README) file tells the details on implementation. Xpsitouch basically ties clicks on the icons to specific functions that can be defined. These functions can be used to do things such as rotate the pointer click (mouse 1, 2, or 3), or launch specific applications or scripts.

## 13.6. Mouse Issues

The psion's touch screen has mouse button 1 pressed by default when you touch the screen. If you hold down the Menu key you can move a window by moving the mouse in the window. If mouse button 3 is the default and you hold down the Menu key, the "mouse" can be used to resize a window by moving the mouse anywhere in the window.

You can change the default mouse button manually using xmodmap:

```
xmodmap -e "pointer = 2 1 3"
```

will change the default button to 2, etc.

Most desktop systems like fluxbox, fvwm, WindowMaker, blackbox include a system for setting up custom keybindings for executing commands. fluxbox includes such a system natively, blackbox requires the bbkeys package. These systems allow one to set up keybindings to do things such as change the default pointer button (what button is executed when the screen is tapped), or adjust volume, or execute a play or record of sound.

The *keylaunch* package from the X windows section at debian.org can be used to tie custom key strokes to executing arbitrary commands such as those using xmodmap to change the mouse buttons. Start *keylaunch &* in your .xinitrc file. Set up a file *.keylaunchrc* that contains:

```
# Format:
# key=...KeyName: Command
#
# ... No modifier
# *.. Shift
# .*. Ctrl
# ..* Alt

key=..*Return: xvt
key=.*.1:xmodmap -e "pointer = 1 2 3"
key=.*.2:xmodmap -e "pointer = 2 3 1"
key=.*.3:xmodmap -e "pointer = 3 1 2"
```

So that

- <Ctrl><1> will set the default mouse button to 1,
- <Ctrl><2> will set the default mouse button to 2,
- <Ctrl><3> will set the default mouse button to 3, and
- <Menu><Enter> will launch an xvt terminal.

It might be possible to set up special key strokes to behave as the mouse buttons - the accessX extensions to X apparently allow you to do that. This is the MouseKeys feature. No luck as yet getting this to work. (If you type "Cntrl""Shift""NumLock" on your desktop, you can then use the number keypad to move the mouse around and execute mouse button clicks. The Psion does not have such a number keypad...)

**Defunct Tools: xbut, XRmouse**

One tool that binds the mouse buttons to key strokes is *xbut*. It allows you to map key stroke combinations to mouse buttons, using a *.xbutrc* like:

```
# Xbut generates mouse clicks from the keyboard
#
# To see the keycodes, run xev
#
66,Shift,1 # Shift+Left
49,Shift,2 # Shift+Down
65,Shift,3 # Shift+Right
```

You can download a binary for *xbut* HERE (http://www.muru.com/armlinux/programs/).

The mouse button functions can also be emulated with XRmouse. This nifty tool uses up considerable RAM at the moment, alas. The XRmouse application can be seen in the top left hand corner of the screenshot at the top of this page. XRmouse takes a bit of getting used to, but it works quite well after a time. The binary and source can be downloaded from http://projects.gnome.hu/xrmouse/index.en.html.

To use XRmouse you need the gtk libraries:

```
libgtk1.2-common_1.2.10-9_all.deb
libgtk1.2_1.2.10-9_arm.deb
```

## 13.7. Keyboard Issues

The .Xmodmap file that works for me is:

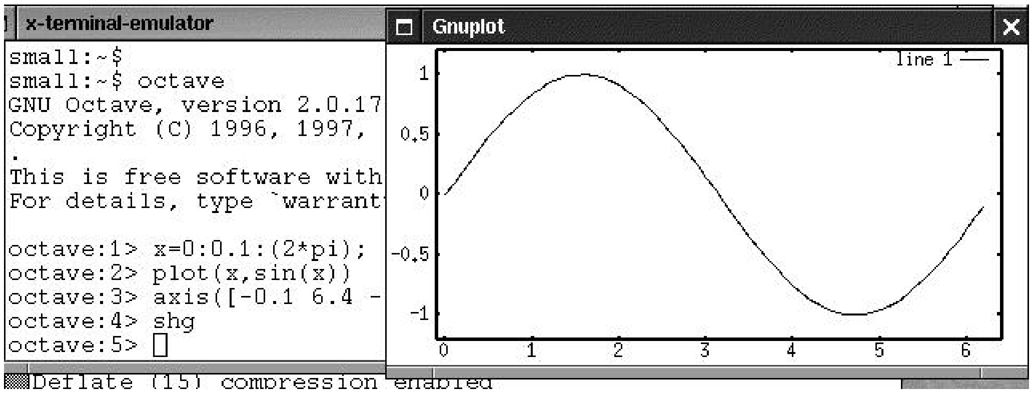
```
keycode 71 = Shift_L
keycode 63 = Shift_R
keycode 47 = Control_L
keycode 55 = Mode_switch
keycode 39 = Alt_L
clear Lock
add Shift = Shift_L
add Shift = Shift_R
add Control = Control_L
add Mod1 = Alt_L
add Mod5 = Mode_switch
```

## 13.8. Useful X applications

While all X applications are technically supposed to work on the psion running X, there are three limitations (1) screen size, (2) memory, and (3) disk space. You must choose your applications carefully, given these constraints.

- **Window Managers** As a window manager, Blackbox works well, as does IceWM and WindowMaker.
- **Terminal Emulators** Given the memory limitation, xvt is the probably best terminal emulator, rxvt is a second.
- **Editors** While xedit will work, axe may be a better editor that uses about the same resources, but it is a little buggy. gvim also works, which may be one's best bet. Or a slimmed down version of vi such as nvi. Slim versions of emacs, or emacs clones, are also available.
- **Web Browser** As a browser, try dillo - set the geometry (try 630x220) and other features in the ~/.dillo/dillorc file (a template for which can be copied from /usr/share/doc/dillo).
- **Calculators** xcalc is a fine calculator that comes by default. Calcoo is another calculator that might work. Other calculators seemed problematic.

  You can also install octave and gnuplot. Octave is a matlab-compatable (mostly) package. (Install the *.deb file for octave/arm from Debian packages for an uptodate version. An older version octave-2.0.17.tar.gz (5MXHOWTO/octave-2.0.17.tar.gz), available here (1.5 MB tarball), is more lightweight, hence more suitable for the 5MX. Install with "tar xfz octave-2.0.17.tar.gz -C /usr/local"). Install gnuplot to get graphics; put "gnuplot*geometry: 400x200+0+0" in your .Xdefaults file to have the figure size work for the Psion. Here is a screenshot for amusement:



```
small:~$
small:~$ octave
GNU Octave, version 2.0.17
Copyright (C) 1996, 1997,
.
This is free software with
For details, type `warrant

octave:1> x=0:0.1:(2*pi);
octave:2> plot(x,sin(x))
octave:3> axis([-0.1 6.4 -
octave:4> shg
octave:5> 
Deflate (15) compression enabled
```

- **Clocks** rclock is a slightly better clock than xclock. It has a interesting options in its .rclock file - see the man pages.
- **Games** xbill is the best game, but the Debian package won't fit entirely on the screen - you can download a binary xbill-2.1-ARM.tgz (5MXHOWTO/xbill-2.1-ARM.tgz) 50 KB package slightly modified for a smaller screen - start xbill with *xbill --size 240*. to change the size (240 is the smallest size). Install with *tar xfz xbill-2.1-ARM.tgz -C /usr/local/*; requires libtif1 and xaw3dg Debian packages. ace_penguin has a nice set of card games of the solitaire variety that work well.

- **Agenda** The "plan" package, under Miscelleneous at the Debian Packages site, is a nice replacement for Psion's Agenda that works well. You can use it to set alarms.
- **PDF viewers** Alas, there are no pdf viewers that work effectively on the 5MX in linux. xpdf would be your best bet. However it is a little too demanding of the 5MX resources. It requires a few MB of space to install the various packages and their dependencies (e.g., the ghostscript fonts).
- **E-Book readers** There seems to be a dearth of e-book readers for linux, and an even greater dearth of e-book readers for openpsion. Possible options are gutenbrowser available from the Software/X11 download at openpsion.org. Also take a look at OpieReader (http://www.timwentford.uklinux.net/), which might also work. There's a fltk version which should work fine with X installed (plus the fltk library which is nice and small) and is reasonably light-weight. There is also a heavier weight version for QT which works with QPE/Qtopia but the fltk version is probably a better bet for the Psion. The source code for the fltk version is at http://www.timwentford.uklinux.net along with makefiles which will require a bit of work as they currently create a flpda (the pda software on the agenda) as well as a fltk version, plus they use the Agenda cross compiler. The target for the pure fltk version is (IIRC) fltkreader. Someday someone might even compile one of these for openpsion...
- **Screen shots** "xwd > screenshot" will make a file "screenshot" of the window or screen (depending on the options). Xwd is an X Window System window dumping utility. Xwd allows X users to store window images in a specially formatted dump file. This file can then be read by various other X utilities for redisplay, printing, editing, formatting, archiving, image processing, etc. The utility "xwdtopnm" can convert the screenshot to a *.pnm file for editing or viewing "xwdtopnm screenshot > test.pnm".

## 13.9. Using a serial mouse!

It is possible to use a serial mouse on the 5MX. But first: it was once rumoured that using such a mouse would burn out the serial chip. However, the serial chip is meant to be the same as on the netBook, a Maxim 78266, which is identical to the Maxim MAX3243-RS chip. And this chip was explicitly designed to support a mouse. But be aware that if you want to use a external mouse, there is a slight danger to your serial chip. I have used a Logitech serial mouse without a problem for a few years now on my netBook, and the same mouse on my 5MX for a short while now without a problem. And no one has ever reported a burnt out serial chip (the rumour originated with the XTM people, the 186 emulator for EPOC).

You will first need a null-modem adapter for the serial port. The 5MX cable is designed to be null-modem by default, so that it can plug directly into the PC. So to use other external devices, such as mouse or GPS receiver, you need a null-modem serial adapter. The netBook HOWTO has more information about this issue. Your local electronics store should carry such null-modem adapters, although these are becoming harder to find as serial ports disappear from the computer landscape.

Then you will need an honest old serial mouse, which are also becoming harder to find. Check your local PC recycling store for a box of used serial mice, or try e-bay.com. A PS/2 mouse with an adapter will probably not work, unless the mouse is explicitly designed to support serial. I have had good luck with Logitech serial mice.

The 5MX Xipaq server does not support a serial mouse, alas. However, the Xfbdev from familiar linux at handhelds.org (http://ipkgfind.handhelds.org/) supports the 5MX framebuffer and a serial mouse. The 5MX touch screen will not be supported with this X server, however. The version to get is xserver-kdrive-fbdev version 6.6.1-11, and you'll also likely need the libxfont0 package. Unpack these with *ar -x foo.ipk* and copy the two binaries over to the equivalent place on 5MX. Place the Xfbdev binary in */usr/X11R6/bin* and make a link from it to X there.

Then you'll need to make a link from /dev/ttyAM1 to /dev/mouse: *ln -s /dev/ttyAM1 /dev/mouse*; the Xfbdev server looks to /dev/mouse for the mouse by default. Disable any getty you may be running on /dev/ttyAM1. Then, you need to setup the serial port for 1200 baud. For this try, this script:

```
#!/bin/sh
stty -F /dev/ttyAM1 4:0:8e9:0:3:1c:7f:15:4:0:1:0:11:13:1a:0:12:f:17:16:0:0:0:0:0:0:0:0:0:0:0:0:0
```

Curiously, you may have to run this twice - it might complain about not being able to carry out the instructions the first time. Then, if you plug in the mouse, and execute *cat /dev/ttyAM1*, you should see a stream of binary characters when the mouse is moved. Alternatively, try using minicom with 1200 baud set.

At this point, with the binary mouse characters verified, you can start X with xinit to see if the mouse works.

You can reset the serial port for 115200 baud with

```
#!/bin/sh
stty -F /dev/ttyAM1 1:0:800018b2:0:3:1c:7f:15:4:5:1:0:11:13:1a:0:12:f:17:16:0:0:2f:0:0:0:0:0:0:0:0
```

## 14. Memory Issues

There are a number of approaches to saving memory on the 5MX. With 16 MB of memory (PRO users have the luxury of 32 MB), one must be careful not to start too many things running at once. Minimize, minimize, minimize, and conserve your memory!

One approach is to use tools that are designed to use less memory. Use ash as your default shell, though bash is generally far more useful. You might even be able to run xinit directly when you login, and so start up no initial shell at all - in /etc/passwd change your shell from /bin/sh to /usr/X11R6/bin/xinit...this might even work.... Use xvt or rxvt as your terminal, rather than xterm. Use a simpler, bare editor, such as nvi, rather than vim.

There are also ways of starting programs that save memory. For example, starting X using *exec xinit* , rather than *startx* or *xinit* saves the memory of a shell - startx is a shell script that stays running while X is running, and with *exec* X is started after exiting the shell and without restarting the getty on the terminal.

Don't start up applications, getty's, daemons, or shells that are not needed. Note that the installation of some packages such as *at* or *cron* will set up starting those daemons by default at boot time. These are not generally needed all the time, so you keep them from starting by deleting their startup scripts in the appropriate rcX.d directory. Or just stop them when you need more memory.

Another memory savings is to use kernel modules, rather than have all the drivers included in the kernel. This should save substantial memory. Load modules for such things as PPP, serial, IRDA, or sound only when they are needed. Kernel modules on the 5MX do work, although traditionally these have not been set up. The Sarge distribution contains all the utilities for handling kernel modules. The use of modules should save about 0.5 MB or more of memory; without modules the kernel size is about 1.4 MB.

My psion 5MX has the following memory usage for a minimal X session:

**Table 3. TOTAL AVAILABLE MEMORY: 14500KB**

| PROCESS | MEMORY | SHARED |
|---|---|---|

| PROCESS | MEMORY | SHARED |
|---|---|---|
| init | 252KB | 188KB |
| syslog | 288 | 180 |
| inetd | 232 | 160 |
| getty (ttyS1) | 192 | 128 |
| getty (ttyAM1) | 192 | 128 |
| sh | 492 | 412 |
| xinit | 652 | 556 |
| X | 2084 | 988 |
| WindowMaker | 2192 | 1505 |
| keylaunch | 684 | 596 |
| rclock | 712 | 628 |
| xvt | 1196 | 988 |
| sh | 392 | 316 |

The memory reported here is the value for RSS from top - much of this memory is shared among the other applications, particularly in X. This shared memory is an important feature of linux - starting N shells does not increase the usage of memory by a factor of N! What is the free memory available for other applications with X running? Probably around 6-8 MB.

Not all these processes are required. Of the processes listed here, syslog is probably optional, and the getty on ttyS1 is probably as well (it offers an escape from X). inetd and the getty on ttyAM1 might be started up only when one wanted to make a connection. All of which would save a little memory.

So, to be sure, one can't go beserk with the applications, while running Netscape, but many things will work fine.

As has been discussed, if you do run out of memory for a particular process, you can try to set up a swap space on your CF disk using a swap file or partition. However, this will be very slow. But it might just let you do what you want to do, with patience. This is a solution of last resort.

## 15. Freeing Up Disk Space

So you've installed your system, and you want to install X, but you don't have enough room on your disks to install it? What can you do to free up some disk space? There is a huge amount of space that can be freed up by deleting the unused material in a dpkg-based system.

1. **Strip libraries and binaries.** Many libraries and binaries are distributed with debugging information. Running *file* on these files will tell you "not stripped" if so. When copying binaries to your root filesystem, it is good practice to use:

   ```
   objcopy --strip-all FROM TO
   ```
   Important: When copying libraries, be sure to use strip-debug instead of strip-all. objcopy is a binary found in the binutils package.

2. **Delete metadata in /usr/src/doc/* directories.** These include changelog's, todo list's, readme's, etc., but be careful what you delete.
3. **Delete material in the /usr/share/locale directory.**
4. **Delete material in the /usr/share/keymaps directory.** You are stuck with the psion keyboard, and don't mess with it.
5. **Delete unused fonts from the /usr/share/consolefonts directory.**

# 16. Cross Compiling a Kernel for the 5MX

It is usually not necessary to compile your own kernel. Binaries are available and so far they have been regularly updated as the kernel development proceeds. Should you want to dabble in, e.g., kernel development, however, here is the basic information on how to compile a kernel.

## 16.1. Prerequisites

There are four things you will need to gather together to compile a kernel for the 5MX:

1. the ARM cross compiler (x MB)
2. the kernel sources (about 30 MB)
3. Russel King's generic ARM kernel patch (a few hundred KB)
4. Tony L.'s Psion 5MX kernel patch (100 KB)

The sections below discuss how to go about obtaining these, and what to do with then once you've got them. The ARM cross compiler can also be used to compile other programs that will run on the 5MX.

## 16.2. The Cross Compilation Environment

Your own custom kernel can be created using the cross compilation environment. "Cross compilation" means using an i386 desktop computer to compile code that will run on the ARM computer. The cross compiler will also allow you to compile your own applications for the ARM system. To set it up, go to http://www.aleph1.co.uk/armlinux/book/x1768.html for some (now slightly out of date but still functional) info on available toolchains. I like the emdebian one best as the install is painless and operation is seamless.

These days, however, the scratchbox environment (http://www.scratchbox.org/) is probably the best way to go.

You can also download an emulated Debian ARM system (http://909ers.apl.washington.edu/~dushaw/ARM/). This is a complete Debian Sarge system for ARM processors emulated using qemu. This is not technically a cross-compiler, but a native compiler. This system is easy to install and easy to use, but has the one downside that it runs a slower than, e.g., scratchbox. It has the advantage that since it really is Debian Sarge, compiled binaries can be tested, and are guaranteed to work, on a Debian Sarge system used on the Psion - there will be no library issues.

## 16.3. Compiling a Kernel

Using the cross compilation environment described above, you can create your own kernel. To compile a custom kernel, you will need to download a kernel tarball - see http://www.aleph1.co.uk/armlinux/book/kernelcompile.html ( http://www.aleph1.co.uk/armlinux/book/kernelcompile.html) for some generic info on kernel building. You will need the base source code for the kernel, the ARM patch (rmk's), e.g., patch-2.4.18-rmk6.gz, and a 5MX patch.

I believe that as of 5/06 OpenPsion has adapted kernel 2.4.27 as their standard kernel, with 2.4.32 being the development kernel. Kernel 2.4.19 works fairly well as well.

The 2.4.27 kernel, prepatched for the 5MX, can be downloaded from http://svn.exactcode.de/linux24-psionw/ using subversion.

```
svn co http://svn.exactcode.de/linux24-psionw/ linux24-psionw
```

Subversion is a Debian package. I think this site tries to keep the 5MX kernel patchs uptodate. You can list the patches that have been applied by

```
svn log http://svn.exactcode.de/linux24-psionw/
```

or you can see the changelog on line at http://svn.exactcode.de/ChangeLog-linux24-psionw.

The 2.4.32 development kernel can be downloaded from http://www.woodall.me.uk/psion/psion.html where you can also get kernel 2.4.19.

If you want to start from scratch and apply all the patches yourself, the kernel source code can be obtained HERE (ftp://ftp.kernel.org/pub/linux/kernel/v2.4/), and the ARM patch can be obtained HERE (ftp://ftp.arm.linux.org.uk/pub/linux/arm/source/kernel-patches/v2.4/). Then, get the 5MX patch for the kernel from http://linux-7110.sourceforge.net/files/Kernels/5mx_and_Revo/ (files/Kernels/5mx_and_Revo/) - look for a file like: "linux-2.4.18-rmk6-5mx4.patch.gz" (this one would be for the 2.4.18 kernel, of course). Be sure that the kernel source, the ARM patch, and the 5MX patch all have the same version number - version 2.4.18 is used as an example here.

Assuming that you have downloaded these and copied them to a directory like $HOME/armkernel, execute the following commands:

```
cd $HOME/armkernel
gunzip linux-2.4.18.tar.gz
tar xvf linux-2.4.18.tar
cd linux
patch -p1 < patch-2.4.18-rmk6
patch -p1 < linux-2.4.18-rmk6-5mx4.patch
```

These commands will patch the Linux kernel for use with the Psion Series 5MX.

You must now configure and compile the kernel. To configure it, execute:

```
make psion_5mx_ericsson_mc218_config
```

[uk, us, or de keyboard types are now set by kernel configuration]

```
make oldconfig
```

then one of:

```
make xconfig
```

or

```
make menuconfig
```

or

```
make config
```

(in this context the latter is preferred as it only asks you about the new options and you just take the defaults for them all - the menu stuff just makes it more conplicated). You'll need to set your keyboard type - uk, us, or de (and a patch exists for the nordic keyboard).

Once you've prepared your source, make sure your linux/Makefile has

```
CROSS_COMPILE   =   arm-linux-
```

at about line 24; this should occur by default.

When you have done this, execute these commands to make the kernel:

```
make dep
make Image
```

[NOT bzImage or zImage - kernels must be uncompressed!] This command will create a kernel image named "Image" in arch/arm/boot/ . This kernel image should then work as the kernel image to be loaded by ARLO.

## 16.4. Kernel Modules

The 5MX also supports modules, and the Sarge system includes all the utilities to handle modules. It may be that some of the more specific 5MX drivers are not supported as modules, however. But if you want to try modules, you can have a smaller kernel size and save a bit of memory.

After *make modules*, install the modules into /usr/src/arm/ [or some other directory] as follows:

```
bash$ make modules_install INSTALL_MOD_PATH=/usr/src/arm/
```

If your kernel version is x.y.z, this command will place the modules into the /usr/src/arm/lib/modules/x.y.z directory on the host, which can then be placed into an suitable filesystem, or transferred to the target machine. (note that /usr/src/arm/lib/modules/x.y.z should become /lib/modules/x.y.z on the target machine). Please also note that you should not install these kernel modules into the hosts root filesystem, since they are incompatible with your host kernel.

However, within the scratchbox virtual environment, a "make modules_install" will just safely put the modules in the virtual /lib/modules directory of the virtual environment.

You will also need the module utilities programs (modutils) in your filesystem to manage the modules. Initially, you may have to execute "depmod -a" on your new system to get modules to work properly, and be sure the modules files are owned by user root, group root ("chown -R root.root /lib/modules/*" will fix that, if not).

# 17. FAQ

## 17.1. Please see the FAQ on the main OpenPsion page (FAQ (http://www.openpsion.org/faqs.shtml)) for other possible FAQ.

## 17.2. Q: Which Psion 5MX model do I need (16, 24, or 32Mb)?

Any of these models will work fine, although there seem to be lingering problems with getting all of the memory recognized on the 24MB machine. It is best if you have the 32MB model, of course. A compactflash card is essential if you want a full system.

## 17.3. Q: How come after OpenPsion boots up the text is faded so as to be illegible?

This problem arose in the 4 bpp mode of the framebuffer depth for older kernels - some of the color mappings were not yet quite correct, but this is fixed with more recent kernels. A work around, short of updating your kernel which is recommended, is to change the color depth by *fbset -a 2bpp*, and perhaps modify your startup scripts to do this automatically. See the "Setting the Framebuffer Depth" subsection.

## 17.4. Q: My disk gives errors when my psion wakes up from sleep mode, or otherwise?

At present, the ETNA code (which governs compactflash) in OpenPsion is evolving. ETNA documentation has been lacking. Mainly, however, the various brands of compactflash cards behave differently in OpenPsion - some cards work very well, others work poorly or not at all. This may have to do with the type of controller the manufacturer put in the card (e.g., Hitatchi, Toshiba, or ??).

You can try backing up your system on a notebook computer, and then reformating the card, which has been known to be helpful.

You can also search for bad blocks on the disk. Some bad CF cards, have been "fixed" by running *badblocks* on it with destructive write test for each partition. Then use the *.badblocks file to mkfs. The command you might use is:

```
badblocks -o 96mb_sdb1_12mb.badblocks -w -p 2 /dev/sdb1
```

In one case, badblocks failed on a USB CF adapter, but then worked with a CF to PCMCIA adapter on a laptop.

## 17.5. Q: Can I have a dual boot system (EPOC and linux)?

Sure - format your compactflash card to have whatever space you need for EPOC and the remainder to be the ext2 filesystem. On the EPOC filesystem (fat16) install ARLO and the kernel, and whatever EPOC programs/files etc. you want to use (in EPOC I use the addressbook and the alarm clock mainly, as well as other things like Solun, a planetarium type thing). I have a 128MB compactflash disk with 32MB reserved for EPOC and 96MB for linux (of with about 40MB is the operating system.)

You will have to reset the clock everytime you boot between linux and EPOC, but a system is being developed that will do this automatically.

Note that you will be able to access the fat16 partition from linux, but not the ext2 partition from EPOC.

## 17.6. Q: How come when I turn my psion off for the night, it turns itself back on again?

There are two possible reasons for this. The first is that if you touch the 5MX's screen slightly it will turn on again - so you may be inadvertently touching the screen as you close the psion. The second reason is that you may have the infrared port running - did you kill the irattach process after you last used it? The IR is always searching for something to connect to, and a false hiccup will cause the psion to wake up again. You should turn off IR when you are not using it, because it uses considerable power.

## 17.7. Q: How do I get the bar "|" or grave "‘" keys?

The bar or pipe key, "|" is <Fn>-t, while the grave key, "‘" is <Fn>-<Del>.

# 18. Resources

## 18.1. The OpenPsion Website

The main website for the OpenPsion project is: OpenPsion (http://www.openpsion.org). The Files section at Sourceforge, OpenPsion Files (https://sourceforge.net/project/showfiles.php?group_id=8846 ) has Arlo, kernels, initrd's, the X-server available for downloading, together with a few applications.

## 18.2. The 5MX Pages

The 5MX Pages (http://linux-7110.sourceforge.net/5MXpages.shtml) of OpenPsion has several links with useful information on linux on the 5MX.

## 18.3. Plp Tools

This is the easiest program to transfer files from you linux desktop to the psion, pretty much plug-n-play. PLP Tools at Source Forge (http://plptools.sourceforge.net/)

## 18.4. ARLO

Zipped source and binary files can now be downloaded from the OpenPsion web page Arlo Files (https://sourceforge.net/project/showfiles.php?group_id=8846)

The official ARLO page is located at Peter van Sebille's webpage (http://www.yipton.net/)

The ARLO HOWTO is HERE (http://www.yipton.demon.co.uk/arlo/latest/readme.html)

## 18.5. Proboot for 5MX PRO

The proboot (http://www.muru.com/linux/psion/proboot/) utility allows the 5MX PRO to be set up to boot linux directly from the compactflash card. The 5MX PRO machines have a 128 byte EEPROM that loads the sys$rom.bin from the flash card; see the README (http://www.muru.com/linux/psion/proboot/README). Proboot does NOT work with the standard Psion 5mx, or Revo, please use ArLo bootloader for those instead.

## 18.6. Kernel Sources

I believe that as of 5/06 OpenPsion has adapted kernel 2.4.27 as their standard kernel, with 2.4.32 being the development kernel. Kernel 2.4.19 works fairly well as well.

The 2.4.27 kernel, patched for the 5MX, can be downloaded from http://svn.exactcode.de/linux24-psionw/ using subversion.

```
svn co http://svn.exactcode.de/linux24-psionw/ linux24-psionw
```

Subversion is a Debian package. I think this site tries to keep the 5MX kernel patchs uptodate.

The 2.4.32 development kernel can be downloaded from http://www.woodall.me.uk/psion/psion.html where you can also get kernel 2.4.19.

The source for the 5MX kernels for the longest time was www.muru.com (http://www.muru.com/linux/psion/kernel/) by Tony L., but development there seems to have ceased.

## 18.7. Precompiled Kernels

Kernels for the 5MX have been made available by Tony Lindgren. Try The Files section for OpenPsion at Source Forge (https://sourceforge.net/project/showfiles.php?group_id=8846). Note that the kernels for the 5MX must be uncompressed.

## 18.8. Initial Ramdisks (initrd's)

I don't actually know of a place to get a good solid initrd for the 5MX at the moment; this seems to be a perenial problem. One of these initrd's can be found at The Distributions for the 5MX Directory at Source Forge (/Distributions/5mx/ ) (the file 5mx-linux-boot.tar.gz, which is not a very good one, but perhaps adequate.) The dpkg distribution tarballs "disk1.tar.gz", obtainable from the the Arlo section (https://sourceforge.net/project/showfiles.php?group_id=8846) at Sourceforge Files, has an initrd in the package already.

## 18.9. The Mail List

You can post queries at the OpenPsion mail list by sending an e-mail to linux-7110-psion@lists.sourceforge.net (mailto:linux-7110-psion@lists.sourceforge.net). You might even get a reply either directly, or appearing on the mail list.

You can browse the mail list archives, including the e-mail you just sent, by going to https://sourceforge.net/mailarchive/forum.php?forum_id=7163.

## 18.10. The Psion PDA Graveyard (sob.)

Psion 5MX's apparently have a common failure with the flexible cable connecting the unit to the LCD. Typical symptoms are a screenful of horizontal lines. If this cable fails there is often nothing to do but get a new psion (try ebay.com?); replacement cables seem to be impossible to find. Some hints on how to repair your broken psion, and perhaps even some hints at preventive maintenance can be found at The PDA Graveyard (http://www.portal-pda.com/graveyard/pdagrave.html) [Broken link...]. WARNING: The photos there are not for the faint of heart; rated R for gore and violence.

# 19. Changes

- July 9, 2006: Modified the alarm discussion to describe the "midnight barrier" issue with the rtc. New scripts, binaries, and source code for rtc.c uploaded to alarms.tgz tarball.
- June 8, 2006: Added link to qemu emulated ARM system as a cross compiler. Issues with compactflash disk recognition and kernel type.
- May 18, 2006: Using a serial mouse in X11. Using the toolbar with xpsitouch. Link for proboot on the 5MX-PROs.
- May 14, 2006: Minor additions/corrections; a few internal links fixed. Added an octave-2.0.17.tar.gz binary tarball; this older version of octave is lighter and works better on the 5MX. Added an xbill binary for sizes down to 240x240.
- May 9, 2006: Added a "Memory" section; pipe and grave keys in FAQ; /proc/powerhook; minor fixes.
- May 8, 2006: More on /proc settings; development of sound section.
- May 5, 2006: Added section on the real time clock, and setting an alarm. Sample sound files available. Removed CF disk benchmark table - archaic.
- May 4, 2006: Removal of all references to PsiLinux.
- May 3, 2006: Sound section, link fixes, kernel sources links.
- April 29, 2006: Finished translating the old HOWTO to SGML format.
- April 26, 2006: More conversion; SGML tables for CF benchmark table.
- April 16, 2006: Started sgml version of 5MX HOWTO.