Above: **Philips was the main European proponent of MSX.**

Above right: ***Eggbert* is an attractive and eggciting free MSX2 game, on our cover disc.**

using a real joystick, a mouse or a mouse emulating a stick.

Command line switches allow serial and parallel port redirection – to 'stdout', by default – logging of audio output to a file, and display timing control. MSX model, ROM and RAM expansion options select variants from an MSX1 with 32K VRAM and 64K main memory to MSX2+ with 128K of each.

Runtime control is by function keys - F6 toggles autofire, F7 loads and – with Ctrl – saves the state of the emulator to a '.STA' file. F8 logs disk changes, F11 invokes the debugger, and F12 quits the emulator. *fMSX* turns off keyboard autorepeat while it runs, to avoid interaction with the MSX system repeat code, and you don't get it back automatically unless you use F12 to quit, or exit with Q from the debugger. To reset the MSX system without reloading, press F11 then type **j 0** to run code from the Z80 reset vector, normally in ROM. Other debugger commands implement absolute and relative breakpoints, memory dumps and Z80 code disassembly. **-diska** sets the disk in the first floppy drive. You can specify several images, each preceded by **-diska** or **-diskb**, and swap between them with F8.

Disk access

MSXDOS uses a conventional MFM format that most modern disk controllers can read, though some MSX systems used 3" media. If you have a compatible drive, the following makes a disk image from an MSX floppy:

```
dd if=/dev/fd0 of=MSX.dsk
```

You can then transfer files to Linux with *rddsk*. The simplest option is **rddsk MSX.dsk** which copies all the files – including the MSXDOS system ones – from the disk image to your current directory. Add **-d** and a directory path to put them somewhere else, and wildcards to select files:

```
rddsk MSX.dsk "*.com"
```

picks all executable files, including the MSXDOS shell *command.com*. Names are converted to lower-case and if a file already exists you are asked if you want it overwritten. Dates are converted as well as filenames.

wrdsk copies files into an image, given their Linux paths as parameters after the name of the '.dsk'. If the image file does not exist, *wrdsk* creates a 720K one automatically before putting the indicated files in it. Once you've unpacked any bootable MSXDOS disk, this makes an empty system disk:

```
wrdsk system.dsk *.sys *.com
```

All the command files extracted so far will be on it. However you will end up with duplicates, and potential problems, if you add a file whose name is already used on the disk image.

Java MSX emulator

I could enter and run small programs with the *Java MSX* emulator and JDK 1.3 for Linux, but problems with the keyboard handling made it rather a frustrating experience. I compiled the class files with *javac*, which insisted that I rename the CPU emulator 'z80.java' to 'Z80.java' and the main file 'msx.java' as 'MSX.java' to match public classes they contain. Despite warnings that the code uses deprecated APIs, class files were created and could be launched with *appletviewer* and HTML startup files.

The first few key presses were ignored but then I was able to enter commands in MSX BASIC. My SHIFT keys were ignored, but shifted characters could be obtained by holding down CAPS LOCK. But each time I tried the emulator got stuck in a loop after a few commands had been entered, acting as if a stream of random characters was being typed, and after that there was no way to regain control without resetting the emulator.

The MSX cartridge port of Sega's *Zaxxon* diagonal scrolling game looked good in demo mode but lacked sound, managed two to four frames per second.and didn't respond consistently to the keyboard. Konami's *HyperSports 3* has less demanding graphics and managed much smoother animation, but as it's a severe test of keyboard or joystick handling on the real MSX system it did not prove playable in the Java environment.

TMS9128 graphics seemed solid, if a bit slow, and Z80 CPU emulation is well-tested. The lack of sound is no surprise as the class to emulate the 8910 PSG only contains about 20 lines of stub lines. But until the keyboard handling – in *MSX.java*, coded with Spanish identifiers and comments – can be debugged to suit the JDK this emulator is not really viable on Linux.

The others

fmsxfan is an advanced emulator for the MSX 2+ systems, with 128K video RAM and 4 megabytes of video memory. It looks in the ROMS sub-directory for the MSX 2 plus ROM, but will emulate earlier systems with the **-msx1** or **-msx2** switch.

I had to rely on output from the **-help** switch and some English load-time messages to work out usage and its dependencies. *fmsxfan* detects the *Enlightened Sound* audio mixer, and made reference to the *Allegro* portability library when it shut down - I had *liballeg 3.9.32* installed.

A 32K MSX1 ROM fired up and showed the initial text screen announcing the ROM version but nothing more. The help message is very similar to that for *fMSX*, and *fmsxfan* seems to be a reworking of that emulator.

SVGAMSX uses *SVGAlib* to direct graphics directly to PC hardware, which makes it more efficient on IBM clone systems but less portable than *fMSX*. It seems similar to *fMSX*, shorn of recent updates and utilities. LXF

A studio-quality FM synthesiser cartridge was the unique selling point of Yamaha's CX5 Music Computer.

Links

Follow these links for the latest emulator sources and related information.

AdamEm: **http://www.komkon.org/~dekogel**
ColEm: **http://www.komkon.org/fms/ColEm/ColEm.tar.Z**
Coleco PD: **http://www.geocities.com/newcoleco**
Java MSX: **http://www.classicgaming.com/jmsxemu**
fMSX: **http://www.komkon.org/fms/fMSX**
fMSXfan: **http://www.file hunter.com/emu/fmsxfan/fmsxfan.tar.gz**
MSX tools: **http://www.math.utwente.nl/~metselaa/msx**
SVGAMSX: **http://www.zophar.net/unix/Files/svgamsx.tar.gz**
VirtualColeco: **http://www.classicgaming.com/vcoleco**
Zodiac: **http://prdownloads.sourceforge.net/zodiac**

Coleco emulators

Dig out those old game cartridges.

You have four Coleco emulation options. The most comprehensive is Marcel De Kogel's *AdamEm*, which emulates the Coleco Adam computer as well as the ColecoVision console which is the target of Marat Fayzullin's vintage *ColEm*.

The others are *Virtual Coleco*, a portable emulator written in Java, and *Mission*, the emulator that reveals the common roots of MSX and ColecoVision by running on MSX systems hardly more powerful than Coleco's console. *Mission* is not a generic emulator, requiring small patch files to redirect specific Coleco games to use the MSX hardware, but works well with the ROMs it supports and lets them piggy-back onto any reasonably-compatible MSX 1 emulator.

Coleco's Adam was a substantial if quirky home computer.

AdamEm

The Coleco Adam was an ambitious product bankrolled by the Cabbage Patch doll franchise. It took the established ColecoVision console hardware and expanded it into an eccentric and buggy home computer, bundled with a slow but letter-quality printer and cartridge tape drives.

AdamEm emulates up to four of these drives, four floppies, and a variety of digital and analogue Coleco input controllers. If you've also got the 32K 'WP.rom' you can experience the peculiar attributes of *Smart Writer*, Coleco's bundled word processor which took the desktop metaphor to heights, or depths, Apple never reached. *Smart Writer* displays characters as you enter them on a picture of a typewriter platten, pushing previous lines up the remainder of the screen as you enter new ones, accompanied by unearthly sound effects.

Adam options

Switches or lines in a '.cfg' file determine the speed of CPU and drive emulation, printout redirection, keys, sound, controller and memory expansion. At runtime function keys reconfigure sound channels and joystick or keyboard support. The emulator can also save snapshot files which you can load later to get back to the exact state you were in when the snapshot was taken.

AdamEm can display Coleco graphics via X or *SVGAlib*. Four commands can be used to start the emulation, depending whether you want a window or custom screen and Coleco console or Adam computer emulation. *AdamEm* doesn't work in a 24 bit X setup - it demands 8, 16 or 32 bits per pixel. Your X desktop resolution is immaterial, though the window is very small on a modern desktop as the Adam's 50,000 or so low resolution pixels easily get lost on a high-resolution screen, and the window cannot be scaled up.

The *SVGAlib* version only works properly in a 320 pixel wide mode; the VESA default on my G400 card gives a barely readable quarter-screen view with only half the pixel lines rendered, in a sea of junk. If you've got an old ISA VGA card on the list of those supported by *SVGAlib*, you may fare better.

AdamEm requires a system ROM image. By default it looks for the file 'OS7.rom' in the same directory as the program. This is an 8K file typically called 'COLECO.ROM' by other emulators.

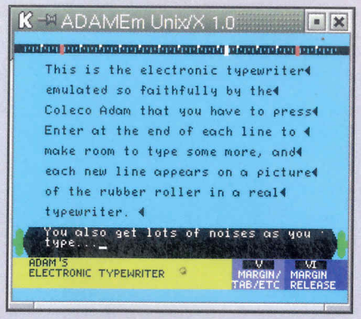
ColEm

The ColecoVision console arrived well after Atari's VCS and Mattel's Intellivision, with higher graphics resolution as it took advantage of the same chips that powered MSX. Sega and Nintendo also contributed to the success of the ColecoVision console, porting simple versions of their arcade games to the platform, and that experience later influenced their proprietary console designs.

ColEm is also at version 1.0, and emulates the ColecoVision console but not the Adam computer. *ColEm* is written and supplied in portable C source. The standard Unix makefile needs slight tweaks for Linux - add the switch **-DLINUX** to the **DEFINES** line so it knows where to find the soundcard header file. Remove the switch **-DLSB_FIRST** if you run Linux on a big-endian system such as a PPC or 68K processor.

Coleco's Adam brings arcade hit *Buck Rogers planet of Zoom* down to earth with a bleep.

Change **CFLAGS** from **-O2** to **-O3** and add **-fomit-frame-pointer** to get faster, more optimised code, and **-ZLIB** if you want transparent access to GZIPped files. Remove **-DEBUG** if you can live without the debugger actioned by the F1 key - this makes *ColEm* a little smaller and faster. If it's too fast, raise the bound on **vperiod** in case 2 on line 89 in ColEm.c - an upper limit of 99 million tames even Gigahertz systems.



The uncommonly literal word-processor was a big selling point of Coleco's Adam computer system.

Color Quirks

ColEm requires an 8, 16 or 24 bit deep colour X display. The default is 8 bit but you can choose others with **-DBPP16** or similar options in the makefile. The source implies that 32 bit colour is also supported, though the 'ColEm.doc' text does not list this.

Unlike Marat's other emulators *ColEm* shows display problems on all the Linux systems I tested. The 8 bit display code gave a blank window and others rendered a row of false-colour images across an X window, rather than the expected high colour display.

When I reported this the author replied, "I do not use Linux, sorry. *ColEm* has been tested on FreeBSD and Solaris." My Matrox G400 may be implicated, but I've seen nothing like this in dozens of other emulator tests, and the same faults occur in SuSE Linux 7.2 with *XFree86 version 4*, and the stable Debian 2.2 distro with *XFree 3.3.5*.

I did get an 8 bit display up on a borrowed GForce video card, but direct colour modes showed the same problems as on the Matrox, so I suspect it's an emulator fault. "I haven't touched *ColEm* for a while", Marat concedes, adding "I am adding Adam support to *ColEm*, but there is not much time available to spend on this." Perhaps a LXF reader can lend a hand?

Java Coleco

There's a Java ColecoVision emulator, though so far this only seems to have been tested on Windows Java hosts. Source is available online, but its restrictive licence prevents us putting it on the cover disk. As usual you'll need a fast virtual and physical machine to get good results out of such an emulator, but *Virtual Coleco* merits attention from Java or J++ aficionados. Its home URL is in the box at the end of this feature.

Olive has her doubts; Popeye challenges Brutus on the Coleco.

NEXT MONTH

Next issue I test emulators for Acorn computers, from the humble Atom to the influential 32-bit RISC Archimedes, a capable Linux platform in its own right whose ARM architecture powers many PDAs, Set-Top-Boxes and other appliances. This home computer range takes in emulators for BBC Micros, once a mainstay in UK schools, and educational programs that bolster a gap in Linux software provision.