ersatz11

pdp-11 system simulator

ersatz-11 beta version 1.1
release date 22 march 1995
copyright © 1995 by john wilson
all rights reserved

COPYRIGHT NOTICE

Ersatz-11 is Copyright © 1994, 1995 by John Wilson. All rights reserved. Distribution of this document and/or the E11.EXE executable file, in unmodified form, without charge, is allowed without restriction. Anything else is strictly forbidden unless you contact me to work something out first.

ACKNOWLEDGMENTS

That should cover most of the important stuff. A lot of stuff has gone right since the initial beta release, mostly thanks to input either from people who have experience with PDP-11 emulation (Bob Supnik at DEC and Alan Sieving at QED), or from people who have helped me debug Ersatz-11 with their configuration (many people, Frank Borger's (U. Chicago) efforts with RT-11SJ and IAS have been particularly impressive, as have Paul Koning's (Chipcom) insights into RSTS and Eduard Vopicka's (Prague University of Economics) help with RSX). Paul Koning also did the layout and graphics for this manual.

FEEDBACK

I would really appreciate continued feedback on the program, good and bad; bad news is usually more useful (I'm very eager to fix bugs, but I have to find them first), but I'm always happy to hear that things work right too, especially if you've gotten it working with an OS that hasn't been tried before, or if you've found a novel way to bootstrap a PDP-11 OS onto your PC. Also I'd be very interested in hearing suggestions for improvements (besides the obvious 22-bit addressing).

As always, make a backup of anything you'd regret losing before running this program.

Yeah I know you won't (I didn't) but think about how you'll feel if something important gets destroyed... Anyway once you're set up it's a nice change being able to throw a couple of dozen RL02 images on a QIC-80 tape for safekeeping, might as well do your other data the same favor.

John Wilson
11 Bank Street
Troy, NY 12180
USA
+1 (518) 271-1982

wilsonj@rpi.edu	(most likely not to change soon)
wilson@tats.wizvax.net	(unless the hostname changes again)
John_Wilson@MTS.RPI.EDU	(going away 06/30/95)
USERHA8G@RPITSMTS.BITNET	(going away 06/30/95)

Contents

1. INTRODUCTION 1

1.1 PLANNED ADDITIONS: 2

2. COMMANDS 2

2.1 INITIALIZATION FILE 2
2.2 ALPHABETICAL LISTING OF COMMANDS 2
2.2.1 @ FILE 3
2.2.2 ASSIGN DDU: { FN | { COMN: | LPTN: } [/IRQN [/SHARE]] } 3
2.2.3 ASSIGN XEN: PKTD[=VV] [PROTO1 PROTO2 PROTO3 ...] 4
2.2.4 BOOT DDU: [/SWITCHES] 5
2.2.5 CALCULATE EXPR (& EXPR IS A SYNONYM) 5
2.2.6 DEASSIGN DDU: 5
2.2.7 DEPOSIT ADDR VAL 5
2.2.8 EXAMINE ADDR 5
2.2.9 FPREGISTER [R V1 V2 [V3 V4]] 6
2.2.10 GO [ADDR] 6
2.2.11 HALT 6
2.2.12 HELP [COMMAND] 6
2.2.13 INITIALIZE 6
2.2.14 LIST [ADDR] 6
2.2.15 LOG DDU: [FILENAME] [/APPEND] 6
2.2.16 LOG { DLN: | DMN: | DXN: | HD: } [FILENAME] [/APPEND] 6
2.2.17 LOG XEN: [FILENAME] [SWITCHES] 6
2.2.18 MOUNT DDU: PATH/FILENAME.EXT [/SWITCHES] 7
2.2.19 MOUNT PP: FILE 8
2.2.20 MOUNT PR: FILE 8
2.2.21 PROCEED [BREAK] 8
2.2.22 QUIT 8
2.2.23 REGISTER [R VAL] 8
2.2.24 SET CPU ITEM [ITEM...] 9
2.2.25 SET DELAY DEVICE C1:N1 C2:N2 ... 9
2.2.26 SET DISPLAY { NONE | PORT N | LPTN: } 10
2.2.27 SET DISPLAY { DR | R0 | PC } 10
2.2.28 SET FLOPPY RX[2]11 10
2.2.29 SET { HERTZ | HZ } { 50 | 60 } 11
2.2.30 SET KEYBOARD [NO]SWAP 11
2.2.31 SET [NO]SCOPE 11
2.2.32 SET SCROLL { HARD | SOFT } 11
2.2.33 SET SWITCH { N | PORT N } 11
2.2.34 SET LPN: CSR=NNNNNN VECTOR=NNN 11
2.2.35 SET PR: REWIND 11
2.2.36 SET TTN: CSR=NNNNNN VECTOR=NNN 11

2.2.37 SET XEN: BOOTSTRAP = { DISABLE | DDU: [/OS] } 11
2.2.38 SHOW CPU 12
2.2.39 SHOW DELAY DEVICE 12
2.2.40 SHOW MMU [{KERNEL | SUPERVISOR | USER} [INSTRUCTION | DATA]] 12
2.2.41 SHOW DDU: 12
2.2.42 STEP 12

3. HISTORY 12

4. NOTES 12

4.1 INTERRUPTS 12
4.2 KEYBOARD 13
4.3 DISK IMAGES 13
4.4 HOST SYSTEMS 14
4.5 SOURCES 14

Tables

TABLE 1: SUMMARY OF COMMANDS 2
TABLE 2:CPU FEATURE KEYWORDS 9

1. Introduction

E11.EXE is a beta test version of a full system emulator for the PDP-11. It runs on any AT-class machine with an 80186 or better CPU, and requires around 350KB of free memory. It is intended to boot and run any PDP-11 operating system (it has been tested with RT-11 (all flavors), RSX-11M, RSTS/E, IAS, 2.9BSD UNIX, Fuzzball, and XXDP+; it seems to have some problems with VM under TSX+, and no attempt has been made to test DOS/BATCH, MUMPS, or any flavor of UNIX other than 2.9BSD; 2.11BSD will definitely not work due to its need for "cadillac" hardware).

Emulated configuration:

- PDP-11/34a CPU with selectable extensions
- FP11A floating point processor
- 248KB main memory
- RL11/RL01-02 disk drives (up to 4)
- RK611/RK06-07 disk drives (up to 8)
- RX211/RX02 or RX11/RX01 8" floppy (using image files and/or 1.2MB floppies)
- DL11 terminal ports; console and up to 15 others (VT100, and/or uses COM ports and/or LPT ports)
- LP11 printer ports; up to 4 (same devices as DL11)
- PC11 paper tape reader/punch (using files)
- DELUA ethernet ports; up to 4 (using packet drivers)
- KW11L line clock (50/60 Hz, settable)
- display register (using special hardware)

Ersatz-11 is available by anonymous FTP from TATS.WIZVAX.NET, or perhaps TATS.WIZVAX.COM depending on how screwed up the name server is when you try, anyway it's 199.181.141.53. The directory is pub/e11, filenames e11.exe, e11.hlp, and e11.doc.

The emulator speed depends on the application (changing MMU registers is an expensive operation so multiuser OSes tend to run more slowly than RT-11FB does, for example), but in general it runs noticeably above the speed of a real PDP-11/34a (with cache) on a 486-DX2/80. Your mileage may vary. It has successfully booted and run RT-11FB on a 4.77 MHz IBM PC with a V20 processor (80186-compatible), but it barely stayed ahead of the clock interrupts.

The FP11 emulation requires a math coprocessor. If the PC has none, then the emulated PDP-11 will have no FPP either. Intel Pentium CPUs that have the floating point divider bug are detected and a workaround is used to get correct results at a slight speed penalty (for DIVF/DIVD only).

I consider this program to be a beta test release because there are still some reported bugs that I've been unable to duplicate, and there are many unfinished or unstarted features that I plan to add, most of which hinge on porting the program to run in 80386 protected mode so as to be able to implement 22-bit addressing efficiently. My apologies to anyone who wants to run it on an 80286 or earlier, the final version will most likely require a 386. Besides, the program is growing so rapidly that nothing has been left untouched long enough to be considered 100% tested. However in my own experience the emulator has been more reliable than real PDP-11s.

1.1 Planned additions:

- MSCP disks. Don't hold your breath, but it will happen eventually in some form. Can anyone tell me the interleave scheme (if any) used on RX23 and RX33 disks?
- Some kind of tape support. ASPI, QIC-80, FILETAPE, or maybe something else.

The system has been tested under the XXDP+ diagnostic monitor. It passes the KD11EA diagnostics DFKAA, DFKAB, and DFKAC, and the FP11A diagnostics DFFPA, DFFPB, and DFFPC. It fails all MMU diagnostics due to the absence of the maintenance mode.

2. COMMANDS

Ersatz-11 recognizes a number of keyboard commands. These are entered at the "E11>" prompt, which appears when the PDP-11 is halted but may be brought up at any time by pressing Alt-SysReq (or by pressing the BREAK key on a serial terminal if the console terminal (TT0:) has been ASSIGNed to a COM port). Commands (and parameters and switches) may generally be shortened to any unique abbreviation.

2.1 Initialization file

When E11 is first started, it looks for a file named "E11.INI" first in the current directory, and then in the directory where E11.EXE is located (or, it searches the PATH if you're still running DOS V2.X). If this file exists than a command is read from the file each time E11 would otherwise prompt for a user command; input for the console DL11 still comes from the keyboard, so the init file may be used both to start up the system and to shut it down (E11 will continue after the BOOT command if you type Alt-SysReq). If a line's first non-blank character is ';' or '!', it is treated as a comment. Processing of the init file can be disabled by starting E11 with the "/NOINIT" switch, and a different file may be specified using the "/INIT:file" switch; the default extension is ".INI" and the above search rules apply unless the filename contains a drive or pathname specification.

Typical E11.INI file:

```
mount dl0: rt11.dsk
; uncomment the ASSIGN command to use a COM port for console I/O
; (baud rate must have been initialized from DOS using a MODE command)
;assign kb0: com3: /irq5
boot/rt11 dl0:
; control returns to the next line when the user presses Alt-SysRq
quit
```

2.2 Alphabetical listing of commands

Table 1: Summary of commands

Command
@ file
ASSIGN ddu: { Fn | { COMn: | LPTn: } [/IRQn [/SHARE]] }
ASSIGN XEn: PKTD[=vv] [proto1 proto2 proto3 ...]
BOOT ddu: [/switches]
CALCULATE expr (& expr is a synonym)
DEASSIGN ddu:
DEPOSIT addr val
EXAMINE addr

FPREGISTER [r v1 v2 [v3 v4]]
GO [addr]
HALT
HELP [command]
INITIALIZE
LIST [addr]
LOG ddu: [filename] [/APPEND]
LOG { DLn: | DMn: | DXn: | HD: } [filename] [/APPEND]
LOG XEn: [filename] [switches]
MOUNT ddu: path/filename.ext [/switches]
MOUNT PP: file
MOUNT PR: file
PROCEED [break]
QUIT
REGISTER [r val]
SET CPU item [item...]
SET DELAY device c1:n1 c2:n2 ...
SET DISPLAY { NONE | PORT n | LPTn: }
SET DISPLAY { DR | R0 | PC }
SET FLOPPY RX[2]11
SET { HERTZ | HZ } { 50 | 60 }
SET KEYBOARD [NO]SWAP
SET [NO]SCOPE
SET SCROLL { HARD | SOFT }
SET SWITCH { n | PORT n }
SET LPn: CSR=nnnnnn VECTOR=nnn
SET PR: REWIND
SET TTn: CSR=nnnnnn VECTOR=nnn
SET XEn: BOOTSTRAP = { DISABLE | ddu: [/os] }
SHOW CPU
SHOW DELAY device
SHOW MMU [{KERNEL | SUPERVISOR | USER} [INSTRUCTION | DATA]]
SHOW ddu:
STEP

2.2.1 @ file

Accepts input from the specified file as if it had been typed at the E11 prompt. The default extension is ".CMD", and search rules are the same as for the E11.INI initialization file. Lines read from the file are not echoed, and indirect files may not be nested.

2.2.2 ASSIGN ddu: { Fn | { COMn: | LPTn: } [/IRQn [/SHARE]] }

Assigns a physical PC device to emulate a particular DL11 (TT0:-TT15:) or LP11 port (LP0:-LP3:). The first (and possibly only) argument after the PDP-11 device name may be either the name of a function key F1-F12, or a serial port COM1:-COM4:, or a line printer port LPT1:-LPT4:.

If a function key is given, then the specified port is connected to a simulated VT100 which can be put up on the screen by pressing Alt and the specified function key (note that the screens assigned to F11 and F12 are not accessible if you still have the 84-key AT keyboard). When one screen is being displayed on the PC screen, the others (up to 11) are maintained invisibly in memory, so they will be up to date when you switch the display to them by pressing Alt and the function key for the screen you want.

Untested feature: If there are two video adapters on the PC (e.g. an SVGA and a Hercules mono card), then one DL11/LP11 pseudo VT100 may be displayed on each. The Alt-function keys choose which of the 12 possible screens is displayed on the primary monitor, and the Ctrl-function keys choose which is on the secondary monitor. Note that it is not possible to display the same port on both monitors at once. I no longer have such a setup so the code is untested.

If the name of a COM port is given, then the specified DL11/LP11 port is connected to that port. Any IRQ from IRQ0 to IRQ15 may be specified, or if the IRQ is omitted then IRQ4 is used for COM ports whose I/O base address is 300h or more, IRQ3 for 2FFh or less. These defaults have been in use since XTs were the standard, and most multi I/O boards will be set up this way. However it's not uncommon with newer boards for COM3 to use IRQ5 and COM4 to use IRQ2 (or IRQ9 really, which is effectively the same on an AT); you will have to use the /IRQ5 and /IRQ2 switches in this case. The /SHARE switch (the /IRQn switch is required if /SHARE is given) means that the serial port has special IRQ sharing hardware (as documented in the IBM AT Technical Reference Manual) so that more than one device may use the same IRQ at once; this hardware is relatively rare, so if you don't know whether you have it, you probably don't. Note that without this hardware it's not possible for more than one device to be actively using the same IRQ at the same time; so for example if you have a mouse attached to COM1 using IRQ4 and it has been initialized by MOUSE.COM, E11 will not be able to use COM3 if it also uses IRQ4 (an error message will be given if you try).

The baud rate, number of data bits, etc. for a COM port should be set with a MODE command from DOS before Ersatz-11 is run.

If the name of an LPT port is given, then the specified DL11/LP11 port is connected to that port. /IRQn and /SHARE may be given as for COM ports; the default IRQ for all LPT ports is IRQ7. E11's LPT handler uses interrupts; I've seen vague references to interrupts not working correctly with some LPT ports, or maybe it was with some printers, but having polled I/O as an option would be horribly slow (either printing speed or CPU emulation or both would suffer) so it's not supported. Interrupts work great with my brand-X LPT port and 13-year-old NEC Spinwriter. Trouble is most likely if you have multiple LPT ports, since they may all try to drive IRQ7 at once.

LPT output is passed transparently, so you'll need to make sure that your OS and printer agree on whether lines end in <CRLF> or just <LF>, and on whose responsibility it is to interpret tabs and form feeds.

Note that it is possible to ASSIGN a TT: port to a printer, or to ASSIGN an LP: port to a screen. Why you'd want to do this I don't know but who am I to stop you. The reason for making them both use the same pool of devices was so that they could both access COM ports, since serial terminals and serial printers are both reasonable devices. LP: ports attached to COM ports or screens respond to XON/XOFF flow control.

The ASSIGN command fails if the specified COM or LPT port doesn't exist, or if the specified (or default) IRQ is already in use and the /SHARE switch is not given, or if you're trying to steal TT0:'s device for some other port (there must always be a TT0: since that's E11's console terminal).

2.2.3 ASSIGN XEn: PKTD[=vv] [proto1 proto2 proto3 ...]

Assigns a packet driver to emulate the specified DELUA ethernet port. If specified, "vv" is the hexadecimal interrupt vector to which the packet driver is attached. If the interrupt vector is not given, then the range of vectors from 20 to FF is searched until a packet driver is found that isn't already in use by another simulated ethernet port. A list of hex 16-bit DIX protocol numbers, up to ten total, may be specified, in which case E11 asks the packet driver to pass only frames of those types. This may allow E11 to coexist with other protocol stacks running on the same PC, as long as they each use different sets of protocol numbers and neither one needs to change the station address after the other is loaded.

DECnet requires the ability to change the address (it wants the station address to be based on the protocol address to avoid the additional overhead of performing address resolution over the network, as TCP/IP

does), so in general running DECnet under E11 will require either that no other network software be running on the PC, or that you install a second ethernet board for E11's exclusive use. This is because the packet driver refuses to change the station address once the board is already in use so as not to surprise protocol stacks that were already running when the change was made and thought they knew the address.

Freeware packet drivers are available via anonymous FTP from many sources including FTP.FTP.COM. Ersatz-11 requires packet drivers that conform to version 1.10 or later of the packet driver specification as published by FTP Software.

2.2.4 BOOT ddu: [/switches]

Boots the system from the specified disk. The disk must have been mounted with the MOUNT command. The optional switch is an OS name; for now the only meaningful ones are /RT11 and /RSTS, /RSX is defined too but has no effect. This has to do with the method used to pass time and date information to a newly booted monitor. RT-11 ignores the time and date passed at 005000 unless the NOP in the first word of the bootstrap is cleared to 0 (HALT) and the bootstrap is entered at 000002. RSTS believes the time and date at 001000 (in a different format from RT-11) regardless of whether its NOP was nuked, but later versions of RSTS save the first word of the bootstrap and execute it later (anyone know why?), so they will halt if the system was booted the RT-11 way. Hence the need for the switch. Note that the OS switches are meaningful only on block devices. If you like typing the time and date manually (or your PC has no RTC) then don't worry about the switch. I don't know enough about RSX or any other OS to set up the time/date for it, if that's even possible; however I am planning to add emulation of the PDP-11/94 TOY clock if I can figure it out, then a privileged program could read it out and set the RSX clock.

There is also a /HALT switch, which means to go as far as loading block 0 into core and setting up the registers, but to stop there. This can be handy for debugging boot blocks.

The BOOT PR: command expects a tape image in absolute binary format, as produced by the "LINK /LDA" command under RT-11.

2.2.5 CALCULATE expr (& expr is a synonym)

Calculates the value of a 32-bit octal expression and displays the result in octal, decimal, ASCII and radix-50. The operators are `* / + -`, unary `+ -`, and `( )`, with the usual precedence. Numbers are either octal digit strings, or decimal if they contain 8 or 9 or end in ".", or hex if preceded by "^X", or radix-50 triplets if preceded by "`^R`"; or general register contents may be specified using the names R0-R5 (with a "′" suffix to indicate the other register set) or SP or PC, R$ or PS means the process status word, and something of the form "′*a*" means the ASCII value of the character *"a"*.

2.2.6 DEASSIGN ddu:

Disables the specified DL11 port (TTn:), LP11 port (LPn:), or DELUA port (XEn:). Deassigning TT0: is not allowed (either explicitly, or implicitly by ASSIGNing its PC device to another PDP-11 device).

2.2.7 DEPOSIT addr val

Deposits the word <val> at absolute 18-bit address <addr>, which is forced even. An error message is returned if an attempt is made to access a nonexistent CSR in the I/O page.

2.2.8 EXAMINE addr

Examines the word at absolute 18-bit address <addr>, which is forced even. An error message is returned if an attempt is made to access a nonexistent CSR in the I/O page.

2.2.9 FPREGISTER [r v1 v2 [v3 v4]]

Sets or displays the FPP registers. r is the FP accumulator number, 0-5, and v1-v4 are two or four 16-bit octal words to write in the register (sorry, not decimal). If no arguments are given then the octal contents of all six ACs are given, along with octal displays of the FPS, FEC, and FEA, and also a bit-by-bit display of FPS.

2.2.10 GO [addr]

Starts the machine at the specified address, or at the address currently in the program counter if none is given.

2.2.11 HALT

If the machine is running, halts it and displays the registers. Otherwise a no op.

2.2.12 HELP [command]

Explains use of Ersatz-11 commands. Just type "HELP" for a list.

2.2.13 INITIALIZE

Initializes all emulated I/O devices, disables the MMU, sets the CPU mode to "kernel."

2.2.14 LIST [addr]

Disassembles eight instructions starting at the specified 16-bit address (within the current I-space map) if it is given, or otherwise at the first address following the last one disassembled by the most recent LIST or REGISTER command.

2.2.15 LOG ddu: [filename] [/APPEND]

Logs all output to the specified character device (TTn: or LPn:) in the specified file. If no filename is specified, any existing log file for that device is closed. The /APPEND switch means to append to an existing log file, rather than replacing it.

2.2.16 LOG { DLn: | DMn: | DXn: | HD: } [filename] [/APPEND]

Logs commands sent to the RL11, RK611, or RX[2]11 controller, or the HD_SYS.EXE pseudo-controller, to the specified file. If no file is specified, the current log file, if any, is closed. The unit number is insignificant, all commands to the controller are logged regardless of the currently selected unit. The /APPEND switch means to append to an existing log file, rather than replacing it.

2.2.17 LOG XEn: [filename] [switches]

Controls logging of ethernet events. If a filename is specified then the log file is opened. If switches are specified they specify what events are to be logged; "/[NO]COMMANDS" controls logging of port commands, "/[NO]RECEIVE" controls logging of received frames, and "/[NO]TRANSMIT" controls logging of transmitted frames. The switches may be specified when the log file is first opened, or later in LOG commands with the filename parameter missing to change what is being logged without having to open a new log file. If neither the filename nor any switches are specified, any existing log file for that device is closed. If no switches are specified when the file is first opened, the default is "/RECEIVE /TRANSMIT". In addition, the /APPEND switch means to append to an existing log file, rather than replacing it.

2.2.18 MOUNT ddu: path/filename.ext [/switches]

Mounts the specified DOS file on the specified PDP-11 disk drive (DL0:-DL3:, DM0:-DM7:, DX0:-DX1:, HD0:-HD7:); that is, PDP-11 reads and writes to that drive access the specified file, which is a byte-by-byte image of a disk, which you presumably loaded from a real PDP-11 using Kermit or DECnet or some equivalent, or built using FLX.EXE or RT11.EXE or PUTR.COM.

For DL:, the drive is set up to emulate an RL01 or RL02 depending on the size of the file; each drive is set separately, so you can have a mix of RL01s and RL02s. The /RONLY switch has the same effect as pressing the WRITE PROT button on a real drive (/WPROTECT is a synonym).

For DM: things are similar to DL:, each of the 8 drives can be either an RK06 or RK07 depending on the file size. /RONLY works as above.

For DX: (DY: is a synonym), the drive is set up to emulate an RX01, RX02, or RX03 disk depending on the size of the file or whether the /SSSD, /SSDD etc. switch was given. The file size is also used to determine whether the file is a block-by-block image of the PDP-11 floppy starting at logical block 0 (such as one gotten by using Kermit to read the disk under a PDP-11 OS), in which case the interleave needs to be un-done since the PDP-11 device handler will re-do it; or a sector-by-sector image starting at track 0, side 0, sector 1. The latter format yields a slightly larger file because the PDP-11 floppy device handlers skip track 0 (something to do with proposed ANSI compatibility that never materialized as far as I know).

A second option for DY: is to give the name of a 1.2MB floppy drive instead of the name of a DOS image file, i.e. A: or B:. Accesses to that RX02 unit will be redirected to the specified floppy drive (and there's no need to re-MOUNT the drive when you change disks) using a 26-sector format equivalent to that of a real 8" disk drive. In fact, if you somehow connect an 8" drive to your PC floppy controller then it should be able to handle RX01 media directly (I haven't had an opportunity to test this though). DEC's MFM format is weird so RX02 disks wouldn't be compatible. The RX211 "set density" command is implemented as a full format, so you can format the weird 1.2MB disks using FORMAT.SAV or SPEC%() or whatever you'd use on your real PDP-11. Two problems: (1) On a 1.2MB 5.25" drive the index hole is in the same place for SS and DS media (8" drives use the index hole to tell them apart) so E11 decides whether to report the disk as double-sided by depending on the /SSxx or /DSxx switches to the MOUNT command (/SSDD is the default); the program detects disk density changes like a real RX211 (well, more or less, occasionally it takes a few tries to notice a change because the PC controller doesn't have a "wrong density" error flag, it's not that simple with IBM-style MFM) but still needs to be told if the number of sides changes. (2) Many, if not most, PC floppy controllers have broken single density modes (the PC BIOS is hard coded for double density so no one cares), so FORMAT/SINGLE or anything else to do with actual single-density media may not work on your PC. FDC chips made by SMC are known OK (FDC37C65LJP for one, the kind that takes one 16/32MHz xtal and one 9.6MHz one), as are at least some by Goldstar. In my testing Intel's 82077 can't write SD disks but seems to be able to read them with some retries; according to the data sheets for the NS PC8477B (which is supposed to be compatible) this is because the chip requires an external pin (MFM) to be grounded to enable SD mode, and for normal AT use it would be reasonable to let it float high (required by RESET mode ID anyway) so I imagine that's what most designers do.

An HD: device exists for compatibility with the HD_SYS.EXE hypothetical hard disk device in the demo version of the so-called "Russian" PDP-11/03 emulator. This way E11 can boot RT11SJ images built for that program (you should type SET HZ 50 before booting them). It supports up to eight units of arbitrary sizes. I have written an RT-11 device handler for this device (not based on the Russian one, some extensions were necessary for future compatibility with planned upgrades to E11) with files up to 32MB, and Eduard Vopicka is working on one for RSX-11M with a higher limit; it is intended that this device act as a stopgap for handling variable sized image files of varying sizes until the MSCP emulation code is completed. If you want to write a driver for some other OS let me know and I'll get around to documenting the device.

IMPORTANT NOTE

The simulator has no control over any caching of disk writes that DOS may do, so it is important that you QUIT out of the simulator to make sure all the pseudo-disk files get closed properly, rather than simply switching the computer off, after shutting down your PDP-11 OS. Otherwise there is no guarantee that all data written to disk by the PDP-11 has really made it onto the DOS disk (in practice I've had no problems, but trouble is theoretically possible so you've been warned). But, the RX211 1.2MB floppy emulation controls the hardware directly (in fact it even intercepts INT 13h to keep DOS's fingers out of things so you can't crash the machine by trying to MOUNT an image file located on a DOS floppy while 1.2MB floppy emulation is active) so as long as the PDP-11 is done with the disk it's OK to take it out of the drive.

Also, the simulator has no control over "Green PC" BIOSes which spin down hard drives after a specified period of inactivity. When E11 accesses the drive after it's been shut down, the BIOS will pause several seconds while it spins the drive back up. During this period E11 is not running, so the simulated PDP-11 may drop incoming characters (but maybe not, E11 maintains a small FIFO buffer on each line) and its clock will lose a few seconds. If this is a problem you may have to disable this BIOS feature. If anyone knows of a way for E11 to handle spinning the drive up asynchronously after a timeout, without confusing the BIOS, I'd love to hear about it, what little APM documentation I have appears to be wrong, at least from my testing (that and/or my BIOS is broken). That way the PDP-11 could keep running and it would just see a slow disk transfer once in a while.

2.2.19 MOUNT PP: file

Mounts the specified file to receive output sent to the PC11 high speed paper tape punch. The default extension is ".PAP".

2.2.20 MOUNT PR: file

Mounts the specified file to provide input read from the PC11 high speed paper tape reader. The default extension is ".PAP".

2.2.21 PROCEED [break]

Continues PDP-11 execution at the address currently in the program counter. If <break> is specified, then it is the virtual address of a single hard breakpoint, where the PDP-11 is guaranteed to stop if an instruction fetch is attempted starting at that address, regardless of what mode the computer is executing in, and regardless of whether the contents of that location have changed since the breakpoint was set. This can be handy for tracing code that hasn't been loaded yet. Note that hard breakpoints and single stepping with the STEP command interfere with the operation of the CPU T bit, so don't combine them with a debugger (or CPU traps diagnostic program) running on the PDP-11 or you'll get strange behavior.

2.2.22 QUIT

Exit the simulator, closing all image and log files and resetting all devices that were in use.

2.2.23 REGISTER [r val]

If <r> and <val> are given, sets register <r> (0-7) in the current register set to contain <val>. Otherwise displays the values of all eight registers, the condition codes, the current and previous processor modes, and the current interrupt priority level. Note that the only way to set the PSW value is by depositing it into location 777776.

2.2.24 SET CPU item [item...]

This command changes the emulated CPU type, either by changing to a new model all at once, or on a feature-by-feature basis. SET CPU 34A sets the emulation to be 11/34a compatible, SET CPU 35 sets it to be 11/35 compatible (mostly—the 11/20-style weirdness with MOV SP,-(SP) pushing the decremented SP, etc. is not emulated). EIS and NOEIS tell whether the EIS instructions exist, FPP and NOFPP turn the FPP emulation on or off (it's permanently off if your PC has no 80x87), and MXPS and NOMXPS enable/disable the MFPS and MTPS instructions.

There are many other items but they correspond to features that either don't exist yet (so even though they appear in SHOW CPU you get an error message when you try to turn them on), or haven't had serious testing. If you want to try them please let me know how it turns out, I believe they work, but it wouldn't be fair to claim I'm sure, that's why this is a beta release.

Table 2:CPU feature keywords

Keyword	Feature
<number>	Set all features as they are in this model number PDP-11.
CPUERR	CPU error register.
CSM	CSM instruction (requires SUPMODE to work).
DSPACE	Split I/D space.
DUALREGSET	Dual register set.
EIS	Extended instructions (MUL, DIV, etc).
FPP	Floating point processor instructions.
MMTRAPS	11/45,55,70-style memory management traps, 3-bit ACF.
MXPS	MFPS, MTPS instructions
PIRQ	11/45-style 7-level software interrupts.
SPL	SPL instruction.
SUPMODE	Supervisor mode.
TSTSET	J-11 TSTSET, WRTLCK instructions.

This gives you the ability to roll your own CPU, which may not correspond to any actual existing PDP-11 model. This may annoy your OS, and changing CPU types while running will almost definitely startle it. Anyway SHOW CPU will show you what you've done. The list of configurable features will grow in future versions to include more actual PDP-11 models, this is just what I have now.

2.2.25 SET DELAY device c1:n1 c2:n2 ...

Sets the number of instructions that the specified command opcodes appear to take to complete on the indicated device. The device may be DELUA, DL11, LP11, PC11, RL11, RK611, RX11, or RX211 (synonym for RX11). There may be an arbitrary number of parameters of the form "c:n" or "c=n", where c is the opcode for the command and n is the number of PDP-11 instructions to delay before signaling completion of the command. Both numbers are octal by default, but may be specified in decimal if they contain the digits 8 or 9, or if they're terminated with "." (actually any expression that would work with the CALCULATE command works here).

The reason that device commands, such as "read sector" on an RX02, or "transmit character" on a DL11, delay signaling completion (by raising the "ready" flag and/or triggering an interrupt) instead of completing right away (which would seem natural in an emulator) is that some OS software contains assumptions that at least a certain number of instructions are guaranteed to be executed before a device is able to interrupt, even when interrupts from that device are enabled. The default interrupt delays are set for the "worst case", so that each one is long enough to avoid any known (or suspected) problems with DEC OS software. The SET DELAY command may be useful in cases where your OS needs a longer delay than the default, or in cases where your OS's treatment of a device is "clean" and you can gain a noticeable I/O speed increase by setting all the delays for that device to 1, or in cases where you're debugging OS soft-

ware and want to test against variety of interrupt rates. If this means nothing to you, you can safely forget about it, this command is intended mainly for advanced users.

Note that some devices don't have command opcodes per se, but the SET DELAY command syntax requires one anyway for consistency. On DL11 SLUs, opcode "0" refers to the delay between reading a character from the receiver buffer, and getting the interrupt for the next character (only if it's the second or later character of a function key sequence on an emulated VT100, all other keyboard interrupts correspond to actual asynchronous events); and opcode "1" refers to the delay between writing a character to the transmitter buffer, and getting the completion interrupt (for emulated VT100s; COM and LPT ports use real completion interrupts). Similarly, PC11 opcode "0" refers to how long it takes to read a character from paper tape, and opcode "1" refers to how long it takes to write one. An LP11 has only one opcode, which is "0" and corresponds to the same thing as opcode "1" of a DL11. An RK611 has only opcodes 0-17, but the SET DELAY command defines an opcode "20" which refers to the delay between the interrupt that acknowledges reception of a head movement command (which is itself delayed), and the ATTN interrupt which signals completion of the head movement.

2.2.26 SET DISPLAY { NONE | PORT n | LPTn: }

If PORT is specified, specifies the octal 80x86 I/O address of a word port which when written as a word, sets the 16-bit display register. Building the trivial hardware to support this is left as an exercise to the reader.

If a PC LPT port name is given, it specifies a port which has a multiplexed LED board plugged into it, and E11 will refresh each half on alternate 60Hz (50Hz) clock ticks; there's a little flicker but it works and requires no chips or power supply, just build your board so that the D0-D7 lines (pins 2-9 of the DB25) drive the anodes of the both the D0-7 and D8-15 LEDs (through the same set of eight 100 ohm resistors since only one set of LEDs will have their cathodes grounded at a time). Then add two NPN switching transistors (2N3904 etc.) with the emitters grounded (pin 25), the collectors connected to the cathodes of all 8 LEDs for that byte, and the bases connected through a 1K current limiting resistor to either /STROBE (pin 1) for the D0-D7 LEDs or /INIT (pin 16) for the D8-D15 LEDs. A bare PCB board is available from the author at cost ($14.68 plus shipping for the current batch, 12/94).

If NONE is specified, then the current DR value is available only from the SHOW DISPLAY command (the default condition).

2.2.27 SET DISPLAY { DR | R0 | PC }

Sets what register is displayed on a hardware LED display register (either the parallel port kind described above or the kind that plugs into a bus slot and is addressed by a word OUT instruction). By default the DR is displayed (i.e. the last word written to 777570), but R0 or the PC may be selected instead, since the null jobs in some operating systems display a pattern in R0 (and the PC in some cases) during a WAIT instruction. The pattern may be used to get a rough idea of system load, and the R0/WAIT method was a standard way to display a number on the PDP-11/70. For completeness, registers R1-R5 or SP may be selected too.

2.2.28 SET FLOPPY RX[2]11

Sets the type of floppy controller that will be emulated. This command is necessary because the RX11 and the RX211 use the same CSR and vector addresses, otherwise E11 could just emulate one of each. The default is RX211 (with a so-called "RX03" drive); the RX11 is inferior in every way to the RX211 (and probably doesn't work with your floppy controller anyway so you'll be stuck with image files) so normally you wouldn't use this command, but if you're trying to boot something that expects an RX11 (e.g. an RT-11 distribution with RX01 boot blocks) you'll need this command since the RX211 is not software-compatible with the RX11 (although it's very similar).

2.2.29 SET { HERTZ | HZ } { 50 | 60 }

Sets the frequency of the KW11L line clock (startup default is 60); Ersatz-11 reprograms the PC timer chip for this rate to simulate line time clock interrupts, and then maintains a count in software so that it knows when to trigger BIOS 18.2 Hz interrupts; there is some jitter due to the BIOS interval being rounded down to the previous KW11L interrupt, but there is no cumulative error so the DOS clock is still correct when you exit out of E11.

2.2.30 SET KEYBOARD [NO]SWAP

SWAP sets the keyboard handler to exchange the functions of the Caps Lock and left Ctrl keys for people who (understandably) don't like the IBM Enhanced Keyboard. NOSWAP sets the handler back so that the keys work as marked.

2.2.31 SET [NO]SCOPE

Sets whether the console terminal is a scope or a hardcopy terminal, for the purpose of handling rubout characters typed at the "E11>" prompt. Mainly useful if the console is redirected to a COM port with a DECwriter (etc.) plugged into it. Also determines whether typing ^L at the command prompt will attempt to clear the screen.

2.2.32 SET SCROLL { HARD | SOFT }

Selects the mechanism used for video scrolling. HARD scrolling offers superior performance (it works by programming the video board(s) to change the start address of the screen each time a full-screen scroll is needed), but may expose problems in video virtualization in GUI DOS boxes, or cause problems with TSRs that use the display. SOFT scrolling does things the slow obvious way, by copying the whole screen a line up on each line feed, and ought to work with anything. HARD scrolling is the default; you should try putting SET SCROLL SOFT in your E11.INI file if your display goes nuts a screen or so into each attempted E11 session.

2.2.33 SET SWITCH { n | PORT n }

If PORT is specified, specifies the octal 80x86 I/O address of a word port which when read as a word, gives the current 16-bit switch register value. Otherwise (PORT not specified), sets the value of the emulated SR to the octal number n.

2.2.34 SET LPn: CSR=nnnnnn VECTOR=nnn

Sets the octal CSR and/or vector addresses for the specified LP11 port (values will be saved for use when the port is created using ASSIGN if it doesn't currently exist, otherwise effect is immediate).

2.2.35 SET PR: REWIND

Rewinds the file currently MOUNTed on PR: (the PC11 reader) so that subsequent input will start over at the beginning of the file.

2.2.36 SET TTn: CSR=nnnnnn VECTOR=nnn

Similar to SET LPn:, but for DL11 ports.

2.2.37 SET XEn: BOOTSTRAP = { DISABLE | ddu: [/os] }

Selects whether incoming MOP boot frames will be honored. Default is DISABLE. If a device (and optional OS switch) is specified, incoming boot packets will act as a keyboard BOOT command with those parameters. Remote booting is not yet supported, so this command has no visible effect yet.

2.2.38 SHOW CPU

Shows emulated CPU type, along with breakdown of features, as well as the CPU type of the host processor (and whether the Pentium FDIV bug is present).

2.2.39 SHOW DELAY device

Shows the currently active list of interrupt delay counts for the specified device, starting with the delay for opcode number 0. See SET DELAY for details.

2.2.40 SHOW MMU [{KERNEL | SUPERVISOR | USER} [INSTRUCTION | DATA]]

Shows the current mapping for the specified space. Defaults are kernel and instruction space.

2.2.41 SHOW ddu:

Shows the configuration of the specified device; this is the function key or COM/LPT port, CSR and vector if it's a DL11 or LP11 port, or the disk type and write protect flag for disks, or the station addresses and portal types for ethernet ports.

2.2.42 STEP

Executes a single instruction step and displays the updated registers.

3. HISTORY

V0.8 BETA, 29-Mar-1994; initial release.

V0.9 BETA, 05-Jul-1994; many bug fixes (mainly trap handling, MMU emulation, DIV instruction, and VT100 reverse video), added RX211 emulation, multiple DL11s, and 50 Hz KW11L mode.

V1.0 BETA, 14-Nov-1994; more bug fixes, added FP11A, RK611/RK06-07, LP11, D-space, and supervisor mode emulation. Also CALC, HELP, INIT, LOG, SET/SHOW CPU, SET DR LPTn:, SET SCROLL, SHOW MMU commands, VT100 graphics/underline, changed to .EXE file (ran out of space in unified code/data segment in .COM file).

V1.1 BETA, 22-Mar-1995; still more bug fixes (IAS finally works), DELUA ethernet emulation, disk LOGging, indirect command files, workaround for Pentium FDIV bug, help text moved to file, Russian HD: device (and RT-11 driver), PC11, display general registers on parallel port LED board.

4. NOTES

This program is dedicated to RSTS V7.0-07 Digby's Bitpile.

Ad Majorem Digby Gloriam.

4.1 INTERRUPTS

The interrupt system ended up having to be a little weird, due to some shameful assumptions in DEC OSes (mainly RSX and RT-11 SJ) about how many instructions are guaranteed to be executed after writing a command to a device CSR, before the device will complete the operation and interrupt. Since MS-DOS doesn't support asynchronous I/O (unless you go to extremes which wouldn't have made sense in a CPU hog like an instruction set simulator), it's natural to have all emulated device I/O appear to the PDP-

11 to be instantaneous (not really, the PC takes a break between PDP-11 instructions to do the transfer), with the completion interrupt occurring on the instruction following the one that started the transfer. Unfortunately this catches some OS code with its pants down; in my testing, RSX appeared to issue WAIT instructions for TTY output which was assumed not to have completed yet a few dozen instructions after writing a character to a DL11 (thus hanging the system), and similarly the RT-11 SJ (but not FB/XM) keyboard interrupt service routine runs with interrupts enabled on the assumption that another keyboard interrupt couldn't possibly happen before the current ISR finishes (when this does happen the ISR recurses and the characters are put in the buffer in reverse order, which was happening with VT100 keypad keys in E11). My solution to these problems was to put in a queueing system, so that the interrupt (and in some cases the transfer itself) doesn't occur until a set number of instruction fetches (which in most cases I set on the order of hundreds) after the instruction that started the transfer. This slightly slows down I/O, and at some point I may add commands to make each of the delays settable, for tuning in cases where my hard-coded delays turn out not to be enough, or cases of "clean" code which makes no assumptions in which case things can be sped up by setting the all delays to 1. I'd be interested to see how these OSes handle really fast serial lines on actual slow PDP-11s. All of this applies to disks too, although I haven't run into any cute problems yet of code that executes within a buffer that a disk is reading to and counts on the CPU's ability to "outrun" the disk and JMP out in time; but just to be safe the disk code delays actually doing the transfer until it's about to post the interrupt anyway. RK06/07 seek attentions are delayed still further beyond acknowledging the seek command so as not to confuse overlapped seek drivers; however you'll get faster results using a non-overlapped driver if one is supplied with your OS. Since usually all your emulated disks are on one physical DOS disk with one head carriage, there's nothing to overlap anyway.

4.2 KEYBOARD

The keypad layout may take a little getting used to but it's intended to be familiar if your fingers are already comfortable using KED or EDT on a real VT100; just don't look at it. The digits and '.' key work as marked in Num Lock mode (otherwise only the arrows work). The keys around the top and right edges of the keypad are *not* as marked, but correspond to the PF1-PF4, hyphen, comma, and ENTER keys of the VT100 (the comma key is missing unless you have an 84-key AT keyboard; Northgate Omnikey 102 keyboards have an = key where the VT100 comma belongs but unfortunately it transmits the same scan code as the =/+ key on the main keyboard so E11 can't use it as a comma). To get the normal function of the Num Lock key (and Esc, Scroll Lock, and Sys Req on an 84-key AT keyboard), press Alt, Ctrl, or Shift at the same time. The keypad hyphen, comma, ENTER, and period keys are also available as F6, F8, F10, and F9 keys. Backspace is CTRL/H, line feed is CTRL/J.

4.3 DISK IMAGES

Getting a snapshot of a bootable disk for your OS into a huge DOS file is Your Problem, I don't want to get involved in pirating software from a company whose legal department is probably bigger than any *company* I've ever worked for. BTW, my local (Albany, NY) DEC office has no conceptual problem with selling a real license for a PDP-11 that doesn't really exist. Anyway I used Kermit to slurp the disk images out of a real PDP-11, and I've heard of people using Process Software TCP/IP with a live PDP-11, or DECnet and Pathworks, or Mark Aitchison's RT11.EXE (a fine program) with an RX50 distribution kit.

PUTR.COM, a companion program to E11, is available by anonymous FTP from TATS.WIZVAX.COM, in the pub/putr directory and knows how to read and write RT-11 format volumes on a variety of media, as well as how to read RSTS/E and OS/8 formats. It can write blank container files with the serial numbers and (empty) bad block files filled in correctly, which can be useful with any OS.

Paul Koning (former RSTS developer) has written a very nice program named "flx" for manipulating files in RSTS disk images, among other things it can build a bootable disk given the files from [0,1]. It's available from FTP.UPDATE.UU.SE in pub/pdp11/rsts/utils/flx and is written in portable C, so it can be

used with any emulator (or with real disk packs on a VAX). A DOS executable is included with the sources.

WATCH OUT

The RT-11 DL: and DM: device handlers expect to find a bad block replacement table in block 1 of a disk. If something else is there (like the pack label in Files-11 and RDS 1.1 and later, or the MFD in RDS 0.0), they will replace blocks at random and you'll get a corrupted disk image (yes, I'm sure :-). So either hack your Kermit (etc.) to use the appropriate .SPFUN instead of .READ, or don't use RT-11 to read non-RT-11 disks.

4.4 HOST SYSTEMS

"Why didn't you write a version to run on the {machine} under {OS}?" The program is written entirely in 80x86 assembly language (over 28,000 lines); porting the devices, debugger, interrupt system etc. to another architecture would be straightforward but the instruction set processor depends heavily on similarities between the 80x86 and the PDP-11 (most notably byte order and the condition flags; correcting the byte order and/or deriving condition flags "by hand" on another machine would be very inefficient). The FP11 simulation likewise relies heavily on the 80x87 data formats. Also the VT100 emulator gets pretty intimate with the video hardware. So it seems a little unreasonable for you to ask me to spend months MORE for no pay, writing the whole thing over again for your machine, just to save you paying a couple of hundred bucks for some cheesy mono 486DLC machine if you don't already have one. Of course, if someone with a financial interest in keeping PDP-11 code alive at above full speed wants to donate an Alpha or PowerPC system to motivate me to port the whole mess to it, that's another story!

Meanwhile, if what you want is a UNIX-based PDP-11 emulator in C, three (or more?) have already been released, by Bob Supnik of DEC (J-11, in /pub/mbg/simulators on FTP.STD.COM), Eric Edwards of RIT (11/40? /pub/csh/mag/pdp.tar.Z on FTP.CSH.RIT.EDU) and der Mouse of McGill (FP-less J-11? /pub/people/mouse/pdp11 on FTP.CIM.MCGILL.CA). You'll need a faster machine to get the same results, but at least you get portable source code.

4.5 SOURCES

At this time I am not making a public release of the source code, because it contains proprietary code from a possible forthcoming commercial product. Also the program is changing rapidly and I like having all the bugs where I can see them.