Unearthing The PDP-15's Operating Systems

Bob Supnik, 03-Mar-2003

Summary

On 13-May-2001, the PDP-15's Advanced Monitor operating system was successfully booted on the SIMH (history simulator) emulation system. On 31-Jan-2003, DOS-15 was recovered as well. The discoveries and events leading up to these milestones illustrate the vital role that the Internet plays in enabling computer history enthusiasts world-wide to work together for computing preservation.

Background

DEC's 18b computing family (the PDP-1, PDP-4, PDP-7, PDP-9, and PDP-15) was of significant historic interest. The PDP-1 was DEC's first computer, and the base for the first "video game" (SpaceWar). The PDP-7 ran DEC's first mass storage operating system (DECsys), and the first version of UNIX. Despite their historic importance, the 18b family was a limited financial success and was the first of DEC's computing families to go out of production. Development ceased in the mid 1970's, and the last 18b computer was produced in 1979. As a result, functioning 18b systems are rare, and many of the key software systems (DECsys, UNIX V1) have been lost. At the beginning of 2000, none of the later 18b software systems was available.

The Computer History Project is an Internet-based collection of computer history enthusiasts. Its goal is to recreate historic systems via emulation and to collect and transcribe software that ran on those historic systems.

Initial Documentation

The first step in preservation and recreation is to collect documentation about the target systems. For the 18b family, this was very challenging. The User Manuals for the early systems (PDP-1, PDP-4, PDP-7) are inaccurate, misleading, and sometimes just plain wrong. Peripheral and option documentation is sketchy or non-existent. The Maintenance Manuals and logic prints are really the only authoritative sources. For all these systems, documentation is rare, as fewer than 200 were produced in total.

Documentation for the later systems (PDP-9, PDP-15) is more plentiful but not necessarily better. The PDP-15's User Manual, in particular the First Edition, is notorious for its inaccuracies. Again, the Maintenance Manuals, logic prints, and where available, diagnostics are the best source of accurate information.

The starting point for the documentation search was the DEC Archive, while it still existed. The Archive contained these documents:

- PDP-1 Handbook (second edition)
- PDP-1 I/O Manual
- PDP-1 Maintenance Manual
- PDP-4 Handbook
- PDP-7 User Manual (preliminary)
- PDP-7 Maintenance Manual (including logic prints)
- PDP-9 User Manual
- PDP-15 User Manual (first edition)

This sufficed to write a pair of preliminary 18b simulators, one for the PDP-1, the other for the rest.

Initial Software

With first pass simulators available, the next step was to collect software for testing and demonstration purposes. This proved to be as difficult as finding the documentation. For the PDP-1, the sources to Lisp and SpaceWar had been published. For the later 18b systems, no software was available on public sources. The restoration of Lisp illustrates the importance of the Internet in historic computer salvage. The source came from the Internet (transcribed by Gordon Greene); the PDP-1 macro assembler was derived from a PDP-8 cross-assembler (by Gary Messenbrink) found on the Internet; and the final debug was done remotely (by Paul McJones).

The initial PDP-7 software came from an old PDP-10 backup tape found by Dave Waks. Again, the Internet played a vital role in salvaging the code. The tape was transcribed on a 7-track tape drive by Paul Pierce. He shipped the raw bits over the Internet to Tim Litt, who decoded the PDP-10 backup formats. Tim in turn sent the transcribed bits to the author for debugging.

From 1998 to 2000, nothing was found for the PDP-9 or PDP-15. In 2000, Al Kossow found and transcribed a set of paper tapes from the McMaster physics lab. Among these tapes were some diagnostics and several copies of FOCAL for the PDP-15. The diagnostics sufficed to debug the PDP-15 simulator and get FOCAL running. At the same time, David Gesswein found a set of DECtapes for the Advanced Software System.

DECtapes

To revive the Advanced Monitor System, the SIMH team would have to find a way of dealing with DECtapes. In the 60's, DECtapes were the principal form of mass storage on DEC minicomputers; the only affordable alternative was fixed-head disk, which was non-removable. DECtapes posed multiple challenges:

- DECtapes must be simulated with great precision. DECtape software was timing-dependent; in addition, it relied on the ability to examine individual words as they were read into or written from memory.
- DECtapes are difficult to transcribe. The only way to transcribe a DECtape is on a real DECtape drive, which is a complex mechanical device and difficult to maintain. In addition, the DECtape format for the PDP-9/15 requires special handling on the PDP-8 and PDP-11, the systems most likely to have survived and to have working drives.

As a prerequisite to implementing DECtapes, the unresolved issues in the 18b simulators needed to be cleared up. Via the Internet, Al Kossow, Max Burnet, and David Gesswein provided additional critical documents:

- PDP-1 Handbook (third edition)
- PDP-4 Maintenance Manual
- PDP-9 Maintenance Manual
- PDP-9 Schematics
- PDP-15 User Manual (sixth edition)
- KE09A (EAE) Reference Manual
- KF09A (API) Reference Manual
- TC02 (DECtape) Instruction Manual
- RF15 (DECdisk) Maintenance Manual
- PDP-9 Advanced Software Systems Monitors Manual
- PDP-15 Foreground/Background Reference Manual

These sufficed to answer the outstanding questions.

To cope with the expectations of DECtape software, the SIMH DECtape emulator implemented a word-by-word, time-based model that provided full simulation of acceleration, deceleration, and tape turn-around. By source code count, it was the most complicated peripheral simulator in SIMH. The simulator successfully ran the DECtape exercisers for the PDP-9 before any attempt was made to run DECtape-based software.

David Gesswein was able to transcribe PDP-9/15 DECtapes, including both the Keyboard Monitor System and the Foreground/Background System, using a PDP-8/E with TD8-E controller. The TD8-E was a highly simplified version of a traditional DECtape controller. It read every frame off the tape and left the decoding of the format to software. Thus, it was the ideal device for reading PDP-9/15 DECtapes; it disregarded the format differences and delivered raw bits from the tape for software to decode.

The Keyboard Monitor System

All the prerequisites for reviving the PDP-15's DECtape operating systems – documentation, simulator, transcribed DECtapes – seemed to be at hand. There

was one last problem. Unlike contemporary systems on the PDP-8 and PDP-11, the Advanced Software System did not bootstrap by reading DECtape block 0 into memory and jumping to it. Instead, it required a bootstrap paper tape that was loaded by the hardware read-in facility. For more than a year, all attempts to find this paper tape failed. Appeals on the Internet brought no response. The various private collectors on the Internet, and the Computer History Center, drew a blank.

In May 2001, an email exchange with Hans Pufal in Grenoble France about the PDP-10 revealed that Hans had a paper-tape bootstrap for the PDP-9. He had no direct way to transcribe it. Ingeniously, he scanned the paper tape in sections on a standard optical scanner, wrote a program to decode the pattern of holes, and verified the results by hand. He sent the results to me on May 10, 2001, and I immediately tried it with David Gesswein's DECtape images.

Unfortunately, it didn't work. One issue was a lingering bug in the DECtape simulator. A second was that the Foreground/Background System required the Automatic Priority Interrupt option (API), which hadn't been implemented. But the major hurdles were undocumented software changes that occurred between the PDP-9 and PDP-15. As I wrote on May 11:

> I tried bootstrapping the ADSS [Keyboard Monitor] as well as the F/B monitors. For the former, the starting address is a SKP HLT. For the later, it's all 0's. The very next set of instruction picks up the starting address, masks the address with 070000, and performs other manipulations - clearly reconstructing the bootstrap address, provided that the BOOTSTRAP EXITS WITH A JMS RATHER THAN A JMP. So I changed the exit instruction (at 17745) to be JMS I 17755, and suddenly I am a lot further - not running mind you, but further. ADSS, in particular, prints out a nice error message IOPS03 021400, which indicates that the basic I/O system is alive if not well. (Somewhere I have documentation on its error messages.) I think F/B requires the API option, which isn't implemented.
>
> So this is tremendous progress! I am fairly sure that the boot process is:
>
> 1. read 36(8) DECtape blocks, starting at block 0, into memory, starting at location 100
> 2. use location 105 of the loaded image as the starting vector
> 3. if booting pdp-9 software, jump to it; for the -15, jms to it

The next day, the error message was traced to a customization in the interrupt skip chain:

> With Hans' bootstrap tape, modified (as I think) for the PDP-15 (exit instruction is JMS rather than JMP), I was getting an IOPS03 error - invalid interrupt. Tracing through the interrupt skip chain, I got to:
>
> IOT 1041
> JMP* handler
>
> I have no idea what IOT 1041 does; it's not any standard DEC device, and it is backwards from normal I/O tests, which are always:

> PSF
> SKP
> JMP* handler
>
> So it must have been a custom device SYSGEN'd into this version of the monitor for the installation. I nop'd the tests out, and the tape got to the keyboard monitor prompt! I don't know how the keyboard monitor works, so I typed in PIP, that resulted in
>
> .SYSLD 1
> IOPS03 ...
>
> so there's still more debug to do, but this is the first sign of life out of an 18b operating system!

The next day, this bug was traced to another undocumented change between the PDP-9 and PDP-15 bootstraps:

> Second difference in bootstrap for the 15 vs the 9: the load image is one block longer. The 9 bootstrap loads 17000(8) words from 100 to 17100. The -15 monitor is 17400(8) words long, from 100 to 17500. The failure to load the last 400 words accounts for all the crashes on keyboard monitor commands. I can now take a directory, print the information message, print the SCOM region, etc.
>
> I <still> can't load or run a system program, so there's more work to be done. One possibility is that the system is gen'd for more memory than I am allowing.

And that indeed proved to be the case. Loading the bootstrap in upper memory allowed the Keyboard Monitor System to run correctly. By running SYSGEN, references to custom devices were eliminated, creating a clean DECtape image.

From Keyboard Monitor To Foreground/Background

With the Keyboard Monitor System running, the next challenge was to bring up the Foreground/Background System. This required additional hardware: memory protection, automatic priority interrupts (API), and (although I didn't realize it) a second terminal. All of these had their issues:

- Memory protection, though implemented, had never been tested and contained serious bugs.
- API required intrusive changes throughout the CPU simulator.
- SIMH had no capability for multiple terminals.

Fortunately, the PDP-9/15's API closely resembled the PDP-10's, allowing the latter to be used as a model for the implementation. The second terminal problem was solved by a kludge that allowed multiple terminals to access the controlling window and keyboard on a sequential, rather than a simultaneous, basis. With these changes, the Foreground/Background System was successfully run on 28-May-2001.

DOS-15

Once again, there was a long interval with no apparent progress. However, Hans Pufal and a team in Grenoble France had been restoring a real PDP-9 (part of the collection of La Cite des Sciences et d'Industrie in Paris). On 30-Aug-2002, they succeeded in booting the Advanced Monitor System on real hardware, for the first time in two decades. The availability of a working system enabled Hans to go, as he put it, “DECtape fishing”. One of the first discoveries was a complete source set for ADSS. But on 30-Jan-2003, he reported an even more significant find:

> I've been doing some more DECtape fishing and have recovered three tapes which appear to be DOS-15 V2A. I have another set of restore tapes which I have so far not been able to dump successfully.
>
> Having read the manuals and checked out the tapes all appeared to be in agreement and I wrote a C program to perform the same functions as the DOSSAVE program would do to reload the tapes onto the disk.
>
> I've looked at the disk structures and all appears to be coherent, I see the MFD at disk block 1777 and the UIC's appear at their proper places also. So I THINK I have an image if an RF single platter disk containing DOS-15.
>
> How to boot it? I have no bootstrap!

Looking through the DOS-15 System Manual (which Al Kossow had scanned and made available on the Internet), I noticed that the calling sequence for the resident bootstrap looked very similar to the DECtape bootstrap for ADSS. Hans had the sources for the ADSS bootstrap and tried it on the DOS image, but it didn’t work. He wrote:

> Actually, I've just been reading RFSBT source and it seems to be "converting" DT units numbers to platter addresses - I don't think we want that. Can you take a look around label PLAT3.

Based on the DOS-15 Manual, I suggested a patch to the boot program:

> You're right, the code is wrong. The DOS15 System Manual says that the unit number should be ignored for the RF15; instead the disk is numbered sequentially from block 00000 to 17777.
>
> Assuming that the "4 word parameter block" for TRAN ends up in .dtblk through dkword, the address calculation needs to do something like this:
>
> <code snippet excised>
>
> This would at least be consistent with the DOS15 documentation, and would function identically to the current code for bootstrapping the system.

That sufficed to get to the DOS-15 $ prompt, as Hans reported on 31-Jan-2003:

Progress:

```
sim> load rfsboot.rim 77637
sim> at rf dosv2a.rfa
RF: buffering file in memory
sim> go


DOS-15 V2A
  ENTER DATE (MM/DD/YY) -
```

But none of the system programs seemed to work; they all aborted with an IOPS21 message. Hans traced the problem to the device driver for the RF15, which was sitting in a strange loop:

> I'm having difficulties firguring ot the following code:
>
> ```
> 75072:  CLA
> 75073:  IOT   7045
> 75074:  IOT   0
> 75075:  IOT   0
> 75076:  IOT   0
> 75077:  IOT   7065
> 75100:  IOT   0
> 75101:  IOT   0
> 75102:  DSSF
> 75103:  JMP   75106
> 75104:  DSCD
> 75105:  JMP   75113
> 75106:  DSCD
> 75107:  TAD   75401
> 75110:  SAD   75722
> 75111:  JMP   75231
> 75112:  JMP   75077
> 75113:  DAC   75072
> 75114:  SNA   CLL
> ```
>
> In particular the disk IOT's., also the IOT 0s seem somewhat strange I assume they are time delays or does IOT 0 do something?
>
> The code is executed right before the IOPS error is declared, my tentative assumption is that the code does something different on the emulator than on the real hardware causing failure.

I found a similar piece of code in the ADSS RF15 driver. Based on the source, I concluded that the code was attempting to size the number of platters and failing due to an emulator bug. Hans verified this assumption by patching the code and was able to get further, but he still hit the IOPS21 error (because there were multiple copies of the code, as it turned out). Even though the extent documentation did not mention this sizing capability, it clearly worked. I revised the RF15 emulator accordingly, and on 02-Feb-2003 Hans was able to get into and out of the system programs. By 03-Feb-2003 he had run the DOS checkout package. A few days later he was able to generate systems on larger disks.

Shortly thereafter, he demonstrated that DOS-15 contained a bug that prevented maximum-sized RF15's from running correctly and generated the first DOS-15 "patch" in more than 30 years! But that's another story.

From image discovery to complete recovery had taken less than two weeks, thanks to tapes from France, documentation from California, simulation from Massachusetts, and Internet-based collaboration and debugging.

Next Steps?

The Keyboard Monitor, Foreground/Background Monitor, and DOS-15 do not exhaust the variety of environments available for the PDP-15. The advanced disk-based operating systems -- RSX-PLUS-III, and MUMPS -- represent even higher levels of capability and would be of great interest historically. Unfortunately, no machine-readable copies have been found (a source listing of MUMPS turned up in the DEC Archives). DEC junked its PDP-15 media archive at the end of the 80's, when the last systems went off contract. DECdisks and RP02's were huge and have all ended up on the scrap heap. Unless there is a complete save set on magnetic tape somewhere, the advanced disk-based operating systems are irretrievably lost.

Acknowledgements

The revival of the Advanced Monitor Systems and DOS-15 demonstrates the critical role of the Internet in creating a virtual community of computer history enthusiasts. The project would not have succeeded without the help of individuals whom I know mostly or exclusively through the Internet. In addition, the Internet allowed for rapid interchange of documents, software images, and folklore.

I am particularly indebted to:

- Max Burnet (Australia), for hardware documentation on the 18b systems.
- Al Kossow (California), for hardware and software documentation on the 18b systems, as well as paper tape images.
- David Gesswein (Maryland), for hardware and software documentation on the 18b systems, as well as the ADSS DECtape images.
- Hans Pufal (France), for finding the critical missing ADSS bootstrap tape and transcribing it without a paper tape reader; and for finding, transcribing, and debugging DOS-15. Hans filled in an enormous number of missing pieces, including reconstruction of DOSSAVE from a description of its functional behavior.

The Grenoble PDP-9 can be seen on the Web at http://www.aconit.org/hbp/PDP9.