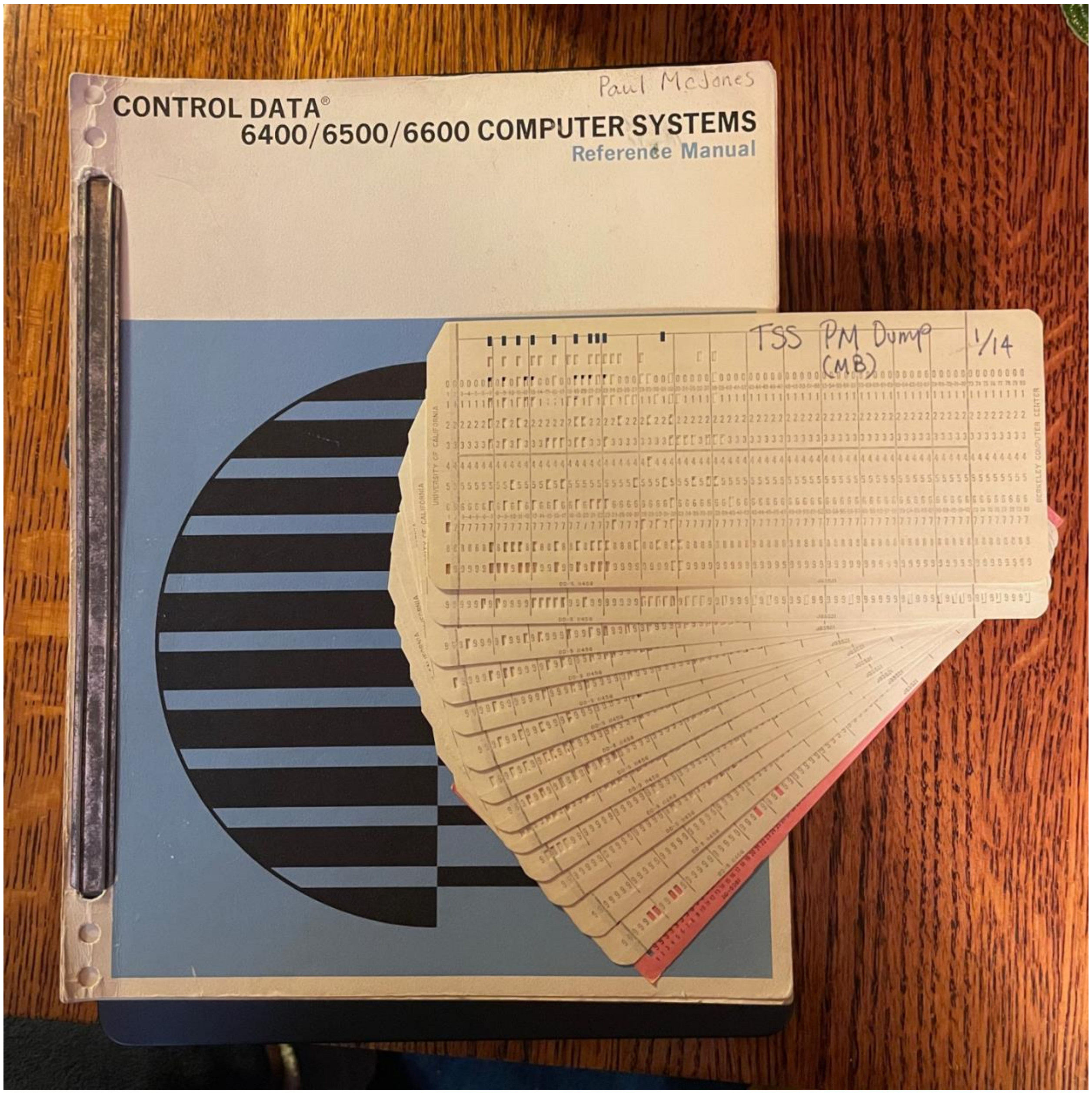
A CAL TSS debugging tool

Paul McJones

22 April 2023
Originally posted as https://mcjones.org/dustydecks/a-cal-tss-debugging-tool



One of the artifacts preserved from the CAL Timesharing System project is a deck of 14 80-column binary cards labeled “TSS PM DUMP”. This is a program for a CDC 6000 series peripheral processor unit (PPU) to perform a post mortem dump to magnetic tape of the complete state of a (crashed) system: PPU memories, CPU memory, exchange package, and extended core storage. Another system utility program, TSS PP DUMP-TAPE SCANNER, allows selective display of portions of the dump to either the teletype or one of the system console displays. I believe Howard Sturgis wrote the PPU program and Keith Standiford wrote the CPU program.

I suspected this card deck was a version of a PPU program called DMP, for which a listing exists. Carl Claunch very generously offered to read (digitize) the cards. He doesn't live nearby so we used the postal mail. He produced a binary file tss.hex. I queried the ControlFreaks mailing list re a PPU disassembler, and Daiyu Hurst sent me a program ppdis.c that generates a listing with opcodes resolved and addresses, octal, and textual representations of each 12-bit word. After upgrading it to eliminate some K&R C function definitions and changing its input format to match tss.hex, I ran it, captured the output, and then began annotating it. As expected, it matched the DMP listing very closely, so I used those variable names, labels, and comments to update the output of ppdis.c, and added a few additional comments, including slight differences from the DMP listing.

The first card is a loader that loads subsequent cards up until one with a 6-7-8-9 punch in column one is encountered. The first card is loaded via the deadstart panel. Page 14 of the CAL TSS Operator's Manual explains:

> HOW TO MAKE A DIAGNOSTIC DUMP OF A SICK SYSTEM
>
> Unfortunately, the dump program requires a different deadstart panel from the system dead start program. Reset the deadstart panel to CAL TSS I, push the deadstart button, read the deck "TSS POST MORTEM" into the card reader, mount a tape on unit 0, and stand back and watch it go. After the tape unloads, reset the deadstart panel to CAL TSS II and dead start the system as usual. Record the reel on which the dump was made along with the other information relevant to the crash.

The Operator's Manual also contains a set of CAL-TSS FAILURE LOG forms recording crashes and attempts to diagnose them.

The program on the cards is very similar to the DMP listing (which doesn't include the loader card), with slightly different addresses and one or two small changes in the code.

The general structure of the program is to dump PPU 0 (whose memory is partially overlaid by the DMP program), then use this working space to dump PPUs 1-9, the exchange package (CPU registers), the CPU memory, extended core storage, and finally write a trailer record. The console display is used for operator messages: mounting a tape on drive zero, progress messages indicating which phase is taking place, and several error messages.

This doesn't sound like a terribly difficult task, but it requires about 1000 instructions on the PPU, which has 12-bit words, one register, no multiply or divide, and an instruction time of 1 to 4 microseconds. There are some additional complications:

1. A PPU can't access the memory of another PPU. When the overall system is deadstarted, PPU 0 begins running a 12-instruction program loaded from toggle switches on the deadstart panel, and the other PPUs are each suspended on an input instruction on a different I/O channel. Thus PPU 0 sends a short program to each one instructing it to output its own memory on a channel, which PPU 0 inputs and then outputs to the tape drive.
2. Similarly, a PPU can't access extended core storage (ECS). So PPU 0 repeatedly writes a short program to the CPU memory that reads the next block from ECS to CPU memory, then does an "exchange jump" to cause the CPU to execute that program. The PPU then reads the block from CPU memory and writes it to tape.

Here is the annotated listing.