Ground System Planning for Long Duration Space Missions Helped by Lessons Learned Resurrecting Obsolete Computers

Steven E. Gemeny

Johns Hopkins University
Applied Physics Laboratory
Laurel, MD 20723

Steve.Gemeny@jhuapl.edu

Michael W. Gemeny

Office of Information Technology
Prince Georges County Schools
Largo, MD

mgemeny@pgcps.org

Abstract

Multi-decade space missions face a new challenge—planning to keep the ground system operational for decades. While early planning for such items as the availability of spare components, accurately documenting the "as-built" configuration of the system, and transition planning for key individuals with unique knowledge are vital to longevity planning, eventually some aspect of the system will no longer be functional. Resurrecting a decade old system in hardware, or as a software simulation, may well become the only methods of assuring ground system functionality.

With few resources available on lessons learned in the resurrection of decade-old technology specifically to support ongoing mission operations, alternative sources for lessons were discovered in an unlikely arena—the online world of "Retro-Computing".

Introduction

This paper chronicles the results, pitfalls and lessons learned from the successful resurrection of two computing systems from the 1970s. By studying the significance of fortuitous circumstances related to these two systems today, designers may better understand and prepare for the future needs of ground systems to be used by long duration space missions. The unavailability of any substantive information specific to spacecraft ground systems longevity planning amplifies the relevance of this work.

A Tale of Two Processors

One of the systems chronicled in this paper, based on the first microprocessor used in a satellite command system, is brought back to full functionality in a hardware implementation using 25-year-old components. Recovering functional software along with the necessary documentation and software development tools, then translating 25-year-old data storage media into formats compatible with present-day systems all proved to be significant challenges. The use of the Internet and a distributed collaboration of geographically distant individuals sharing talent and expertise contributed to the success of the effort in many areas.

The other system, a multi rack "mini-computer" having only one known instance of complete physical hardware still in existence, is virtually resurrected as a fully functional, multi-platform software simulation. The fortuitous archiving of source code for the operating system, generation of system backup tapes, and the necessity to reverse engineer the actual functioning of obscure components of software and hardware that were never properly documented proved nontrivial to the effort. Recovering key software from both paper tape and magnetic tape in formats that are archaic by today's standards turned out to be crucial to the success of the project. The accomplishment of this endeavor in the absence of access to operational hardware is owed primarily to the quantity and quality of documentation and archival data that had been preserved.

The COSMAC ELF

In the early days of satellite guidance, command, and control, discrete component logic was the standard method used by the designers. This method, while effective and robust, became far too rigid and limiting as the complexity of spacecraft increased. Advances in microprocessor technology through the Seventies made it only a matter of time before microprocessors would replace the discrete logic designs. The challenge for designers was that of power availability for the operation of these emerging systems.

All this changed in the spring of 1976 with the introduction of the CDP-1802 by RCA[1]. The 1802 was constructed using CMOS technology, which required significantly lower power than any other microprocessor of the era and was inherently robust against the effects of radiation when fabricated using silicon-on-sapphire (SOS) techniques.

Ground System Planning for Long Duration Space Missions Helped by Lessons Learned Resurrecting Obsolete Computers

The Space Department of the Johns Hopkins Applied Physics Laboratory began an evaluation of the 1802 for spacecraft use shortly after the introduction. In less than a year the analysis was complete and the decision was made to use the 1802 as the command processor for a U.S. Geological Survey spacecraft, MAGSAT, being designed under a NASA contract.[2] The trend continued with other spacecraft design, the most notable being VOYAGER, which included 17 CDP-1802 microprocessors.

The architecture developed by RCA for the 1802 continued using the acronym COSMAC, following the design for the 2-chip 1801 introduced a year earlier. The COSMAC architecture was supported by RCA with a full line of development environments, high-level software, and myriad CMOS peripheral components. One interesting system available from RCA to promote the 1802 was referred to as the Micro Tutor. This simple little trainer featured 256 bytes of memory, eight toggle switches for input and a pair of hexadecimal displays for output. It was enough to give designers a taste of how simple a machine the 1802 could be. So simple indeed that the 1802 based Micro Tutor spawned a series of construction articles in magazines such as *Popular Electronics Magazine* promoting the idea of a build-it-yourself computer for under $80[3]. Thus was born the COSMAC ELF, a simple wire wrap computer trainer that in retrospect appears to have been an introduction to microprocessors for thousands of young engineers—this author included.

The hobby computer industry blossomed in the Seventies as various companies began introducing low cost trainers and software for nearly every processor available; the 1802 was no exception. Software was available in various formats for text editors, assemblers, and even higher-level languages such as BASIC and FORTH could be found, but by 1983 with the introduction of the IBM-PC it all faded into obscurity.

In early 2002, 25 years after the introduction of the 1802, impelled by a wave of nostalgia I began an effort to recover my own 1802-based systems from obscurity.

The first challenge, that of unearthing the construction article, was quickly overcome. I was able to locate my original copy of the construction article, safely tucked away in a box in the basement along with an enhanced 1802-based product from 1978, the Super ELF. Apparently I had boxed it all up to make room for my first PC, thinking that one day it might be fun to show people the way things used to be. Fortunately I had included a complete archive of documentation and program data on cassette tape.

A little bit of scrounging in the "junk box" revealed a full complement of chips needed for the project, pilled [piled? filled?] with sawdust, but still in the crumbling antistatic foam (who said CMOS chips are fragile?)!

Parts and schematics in hand, the wire wrapping went quickly and the hex display came to life under switch control without event after about 6 hours.

The next step was to fire up the much more capable Super ELF which was equipped with cassette I/O and a whopping 4K of RAM to recover the software from the tapes. Unfortunately, the Super ELF did not function. Careful review of the operating manual and some troubleshooting lead to the discovery of significant corrosion on the memory board. The effort to repair the tracks, some completely corroded away, was substantial. I cannot imagine why the manufacturer originally reasoned to include a full-sized copy of the blank circuit card documenting the track patterns as part of the manual[4], but I do know that without this piece of documentation the Super ELF would not have been recoverable and subsequent steps would have been doomed.

With the Super ELF running again I began the process of recovering the software archive from several cassette tapes. The Super ELF was also configured with two pieces of software on EPROMS, a monitor program (1K) that functioned like a rudimentary operating system, and a Tiny BASIC (2K) interpreter. The monitor program included the ability to record program memory onto (and play back from) audiocassette tapes in a format very similar to a popular "Kansas City Standard" of the era. The first attempts to read any of the archived tapes revealed a very daunting problem—the quality of the magnetic recording had degraded to a point that the software was unable to read the data without significant errors.

Degradations in magnetic recording media can occur for a number of reasons. The most significant cause in this case was as a result of a process known to the audio recording world as "print through". The recorded bits from one part of the tape, lying over top of the wrap before it, had caused the oxide to magnetically imprint a ghost image. This ghost signal on the tape had the effect of "smudging" the intended bit making it less likely to be properly detected by the hardware/software involved in reading the data. Without some form of significant filtering and thresholding of the audio, the 25 year-old data would be lost.

It is at this point in my story that I encountered a users group of similar-minded folks online who were all sharing their stories of the 1802 and how it affected their careers[5]. Several of them were having similar problems with recovering data from 1802-based systems around the world. One had even been attempting to load a full ANSI standard version of BASIC (Super BASIC, requiring 12K of RAM) from cassette and in frustration had recorded it as a .WAV file for others to attempt recovering. I joined the effort and soon the distributed collective of 1802 users had located news letters from the late '70s describing the data storage

formats used by various 1802-based products[6]. Another member of the group was able to produce a Windows application based on the documentation that could read and write .WAV files compatible with the various 1802-based systems[7]. By adjusting the filtering on this "ELF Tool" even the most damaged tapes were eventually recovered, error free.

A full ANSI BASIC for the 25-year-old 1802 was within reach, but still not finished. The program could be loaded into one participant's hardware, but without the original documentation and no one in the group having clear recollections of the startup procedure, it was still unusable. Another member of the Super BASIC team was finishing up an 1802 emulator running under Windows. Super BASIC was loaded into the emulation, which permitted extensive "code sleuthing" to uncover the startup procedure from the software perspective. Still, the behavior of the software appeared odd and did not seem to yield a solution. The final set of clues came from a copy of the original brochure on Super BASIC, which included details of memory allocation structure, a command summary, a description of the capabilities including both serial and parallel I/O routines, and most encouragingly, the name of author of the code.

One team member, living in Uruguay, was able to locate an address and telephone number for this author of the code. The international call was quite a shock for the author, but not as much of a shock as the realization that people were still interested in his 25-year-old "first paid effort". Unfortunately, nothing had been saved from the development, and the author had little memory of the code's functioning. Code sleuthing would have to do.

With more clues in hand, the emulation analysis of Super BASIC at startup revealed some memory setup, and reading a one-bit input flag as part of a several-second loop. The condition of this flag determined the destination of a subsequent branch, into either serial or parallel I/O memory area. It was necessary to hold down the "input" key at startup to tell Super BASIC to use serial I/O, otherwise it launched into parallel I/O, which generated no output in the simulation or in the hardware implementation. Applying this information to the hardware implementation effort produced immediate and gratifying results by enabling the serial I/O and a fully functional ANSI BASIC on a 25-year-old hardware platform.

Now that Super BASIC was available again and a significant amount of software for the 1802 was obtainable from the distributed tape archives around the world, it was time to focus on an effective development environment for modifying existing software and developing new components for the 1802.

A normal situation in the early days of hobby computing was that each system tended to be somewhat unique in the details of hardware implementation. This was, and still is, true of 1802-based systems. Most software included complete code listing and procedures for customizing it to a user's hardware environment. This level of documentation is not (yet) available for Super BASIC, but is within reach for several variations of the reduced-function Tiny BASIC. In order to improve the transportability of Tiny BASIC across these platforms it was necessary to translate the code listing on paper into an electronic form compatible with some form of assembler.

Much of the 1802 software was developed using an RCA assembler and the majority of the available listings show instruction mnemonics and assembler directives in RCA format. To date, no one in the 1802 / COSMAC Elf Users Group has been able to locate a copy of this assembler. It became apparent that the best approach for reestablishing the 1802 development environment would be through the use of a PC-based cross-assembler. This approach would also provide for easy distribution and archiving of all resultant software. It would also allow the assembled code to be exported to hardware in .WAV file format via the "ELF Tools" utility. But seeking a PC-based cross-assembler for a 25-year-old processor seemed a bit anachronistic.

Again, online resources among the retro-computing world proved to be the solution. For unknown reasons, members of this clan have been archiving shareware products for nearly every microprocessor through the Eighties. One product in the collection, PseudoSam A18 was a full-featured DOS compatible cross-assembler for the entire family of CDP-180X processors, and freely distributable under the terms of the shareware license included in the package[8]. All that was left was to convert the paper listing from Tiny BASIC into a text file and translate the directives from RCA format to PseudoSam format.

While there are numerous methods applicable to this part of the task, kismet provided an elegant solution. Sometime in the early Eighties, the author of Tiny BASIC allowed a third party to include the core code as part of a development effort for another commercial product. That company subsequently ceased operations, but one principal, the key software developer retained an electronic copy of the entire code library[9]. Moreover, he also had a release for use of the Tiny BASIC code and was willing to pass on the same courtesy for his version of the code. By clipping the entire Tiny BASIC code from this "open source" version and comparing it to the paper listing of the original Tiny BASIC, it was possible to back out the changes and return to the original functionality of Tiny. All that was left was to alter the assembly directives and mnemonics to produce error-free code, and to verify the resulting binary file.

The net result of the entire effort was the complete recovery of several 1802-based operating systems, higher-level languages, and the establishment of a team of capable experts for this 25-year-old, pioneering microprocessor. The effort and challenges involved in this recovery should provide insight into the scope and significance of effective archiving along with the importance of refreshing media for electronic data storage. It is not likely that the 1802 recovery effort provides any technological advancement or other benefit beyond these lessons, and the indulgence of nostalgia.

The HP 2000

The HP 2000 systems, introduced in the late 1960s and marketed through the late 1970s, were popular among educational institutions for their ease of use and often provided these institutions the first example of interactive computing. As a result, many tens of thousands of students are believed to have written their first programs on an HP 2000. These systems also helped transition HP into the data processing market, paving the way for the success of the HP 3000.

The HP 2000 family of minicomputers all used a Time Share Basic (TSB) operating system based on Prepare Timeshare Basic[10]. These systems used one or two processors from the HP 1000 family, which were intended for real-time computing and instrument control applications.[11]

A system was often configured for 16 or 32 concurrently connected users, and often used a bank of modems for remote access. Connection speeds of 110, 300, and later 1200 baud were common for dial-up users, and directly connected video terminal users could enjoy 2400 baud.

With only limited availability of actual hardware for the HP processors and the questionable condition of that equipment in the year 200X, it became clear that the HP 2000 could most practicably be brought back as a simulation.

The simulator used, SIMH[12], was originally a simulator for early DEC processors. The design philosophy of the simulator was to be flexible, and had already been modified to emulate several HP 1000 CPU types and peripherals[13].

HP had long discontinued the software and documentation, and several attempts by various people to engage an HP archivist to search for information were unsuccessful. Instead, various archives of private collectors were used to assemble the necessary binaries, sources, listings, and documents.

In the case of one key piece of software, which existed only as a paper tape, no paper tape reader was readily available to recover the software. The ten feet of paper tape was transcribed by hand and keyed into a modern computer with a hex editor, using a double-blind approach for error correction.

The most commonly used, and therefore desirable version of the 2000 would be "Access", as this version was the most full-featured and most installations were eventually upgraded to Access. Consequently, most of the archived user-contributed program libraries are from Access systems. But Access requires dual CPUs interconnected by a channel-like interface called an interconnect kit. One of the processors, the I/O processor (IOP), would also require simulation of special micro-coded instructions[14].

The descriptions of the IOP micro-coded instructions were eventually found in an Operating System (OS) internal document, which was not widely published[15]. But analysis of OS binary tapes for different processor types revealed that the micro-coded instructions were different between 2100 processors and 21MX processors, and that the documentation was only applicable to the 2100 processor.

The 9-track magnetic source tape for the operating system had been safely stored over a decade ago by one of the authors, a fortuitous situation. Through connections within the online retro-computing groups, an expert in recovery of data from aged magnetic tapes was found in a nearby city. The tape was read and converted to a modern format. The tape image was then provided to another team member to extract the individual files, since it was an image of a tape that was an image of a system directory that contained files that were images of the source files.

The set of source files was found to be only applicable to the 21MX version. To resolve this, an actual set of 21MX Access micro-code PROMs were read and analyzed. With the differences between the two understood, work could proceed with simulating these instructions and we will eventually be able to reconstruct the source for the 2100 version.[16]

Having a simulation of both the main processor and the I/O processor, the project team now needed to connect the two virtual machines together.

The interconnect kit posed its own set of challenges. Since the SIMH simulator is portable between Windows, Mac, and Unix hosts, a socket approach was chosen to simulate the interconnection between two instances of the simulator, one functioning as the IOP, and the other functioning as the main processor. The alternative would be to have one instance of a simulator capable of simulating both CPUs, using shared memory for the interconnect kit.

Another problem was caused by a deficiency in the documentation for the interconnect kit.[17] The sections on Programming and Theory of Operation both clearly implied that each half of the channel was simplex, and that the two combined constituted a single full-duplex channel.

However, one team member recalled from decades-old experience with actual hardware that there was something more to it then was documented. Through careful examination of cable drawings and interface card schematics he was able to convince himself that each channel side was in fact a half-duplex link. However he was unable to convince the simulator developer of the importance of this.

After the initial implementation of the socket-based interconnect simulation was tested, it was shown that not only did the capability exist to send data in a reverse direction on a channel side but also that the operating system was taking advantage of this additional capability.

The hardware documentation makes no note of this additional capability, but the software engineers using the interface seem to have been well aware of it. While each side of the interface is generally used in apposing "forward" directions, information can be and is passed backward on a channel side.

Clearly, the capability was known to both design teams decades ago. The lack of proper documentation of this capability demonstrates a hole in the documentation regime applied by a major manufacturer—the need to document excess capability above that necessary to meet the requirements.

While the quest to simulate Access continues, attention was focused on the "E" version of the 2000 family.

The “E” version was a step backward in the HP 2000 line. It seems that HP wanted to move the price for entry-level systems from the $100k range to the $50k range.[18] This reduction in price was accomplished by going back to a single processor configuration. Because of memory constraints in a single processor, the product was short on features and was not popular.

The established contacts made locating the binaries and documentation a fairly simple task. The scripts and test environment from the Access tests were simplified and reconfigured for “E”. It was found that the versions of available binaries, sources, and documentation all bore different date codes and did not directly relate to each other. The I/O device addresses expected by the available binaries were deduced by disassembling sections of code, since they clearly were different from the available documentation.[19]

A troubling lesson learned working with this operating system was that the disk drive had to be imparted with unrealistic speed in order to successfully generate and boot the system. A similar problem and work-around had been seen much earlier between the Access loader and the tape drive simulation. It should be noted that these device simulations were written based on the documentation and tested using the original HP diagnostics. But clearly they are not yet a faithful simulation of the original equipment. We suspect that the software may be asking the devices to perform operations outside of the normal documented sequence, or are using features that were poorly documented. For example, what would a real tape drive do if a program were to command it to rewind, while it was already rewinding? Or how should a disk drive respond if it were commanded to seek while it was busy searching a track?

With the “E” version, operational attention was again focused on Access. Closer examination of the data flowing over the interconnection between the two processors revealed that changes in timing affected what the two processors were saying to each other. The documentation allowed us to interpret what was being said, but did not provide an overall example of what a conversation should look like. With some guessing about what seemed reasonable in this conversation, and adjusting of timers, we were able to boot the system, but not log in. The best we could do was to get the “Please Log In” prompt to the user terminal, but we could not communicate the log in attempt from the IOP to the main processor. In addition, we would also see the IOP informing the main processor that users had “hung up” on ports that we were not using.

At this point, we have indications of possible problems in the modem control logic, questions about the interconnect implementation, and special micro-coded IOP instructions which have never been proven beyond passing the diagnostics. With this number of variables involved we were forced to consider the possibility of reassembling a system form real components and constructing a device, which would allow us to capture the conversations between to IOP and main processor. As an alternative, we decided to explore the “F” version of the operating system.

The HP 2000 TSB Version “F” was a dual processor system, which was the direct predecessor to Access. “F” did not use any special instructions in the I/O processor.[20] The sources for “F” have not yet been located, but the binaries and documentation were located with little difficulty. The operational procedures were reverse engineered and reconstructed to integrate with the simulator environment. Again, the Access simulation environment was used as a template, and the best timer settings for the tape drive, disk drive, and processor interconnect kit were used.

The “F” version became operational with much less effort than anticipated, however without the source for “F” much less was learned than had been hoped. This leaves us again in a position of having to consider restoring a system from original parts to resolve our outstanding questions.

It should be noted that our goal is to simulate every version of the operating system, throughout its eight or nine year

development cycle. Unlike many space ground systems, it is not our goal to prove under simulation that a program or sequence of commands, such as the operating system itself, will have the desired effect when applied to original hardware. If this were our goal, we would have to first prove that the simulator will respond in the exact same way as the original systems given every possible erroneous condition, a certainly impossible task.

However, since it is our goal to demonstrate the functionality of known working operating systems, we should eventuality be able to establish, with reasonable confidence, that user programs running under these operating systems would behave in the same manner under simulation as they would in a real environment.

This layer of insulation should not be overlooked when anticipating what can be, and what cannot be, expected under simulation.

Conclusion

In the words of George Santayana, "Those who do not learn form history are doomed to repeat it."

Clearly there is significance to the lessons learned from these two efforts that can be applied in a preparatory manner for long duration missions. Key among these is the need to archive information in redundant and technology independent formats (analog paper or microfilm/fiche). Another significant lesson involves the degradation of media and its effect on data availability. This may seem irrelevant in an era of optical storage devices, but recent discussions have revealed that even CD ROMs are not immune to a form of bit erosion after several years. Still to be considered is the dependency of software upon a hardware environment. Without persevering the hardware in an operational state, software cannot be expected to operate unless it can be ported to another platform or placed into a simulated environment of sufficient fidelity.

Machine-readable information is dependent upon a compatible machine to permit its reading. In congruence with this axiom, and given the predictable and increasingly accelerating obsolescence of information technology machinery, the National Archives and Records Administration (NARA) prescribes specific policy for the preservation of federal and public electronic records. Preservation requirements entail timely and periodic alterations to the digital format of the records, including rewriting stored data to new physical media, and maybe changing the way the records are digitally encoded from an obsolescent to a persistent format[21]. Digitally encoded operating systems and software to drive data processing hardware are just as much an electronic record as the data content produced and manipulated by the systems, and are therefore subject to the same preservation policy.

The area of documentation yielded another unanticipated lesson. How is it possible that software and hardware engineers hit upon an undocumented method of using a hardware interface yet allowed it to remain totally undocumented across an entire product line? The effect of this one discovered omission is still being felt and has impeded the restoration of Access on the HP 2000. Obtaining operational hardware to complete the reverse engineering of this interface may be necessary to resolve these remaining issues. Clearly one solution to avoid future occurrences of this type of omission is to require that subsystem designers document "capability". The designers must step back from familiar perspective of "how this component *will* be used" and document the capability of the subsystem from the perspective of "how this component *can* be used". The generation of a capabilities document seems a reasonable method for capturing this information.

There are many other lessons related to the archiving of hardware, software, documentation and other areas that might become obvious to readers as they relate these situations to their own efforts and experiences.

The lessons learned from these two efforts are having significant impact on the longevity planning efforts for the New Horizons mission to Pluto[22].

The New Horizons ground system, including the observatory simulator, is required to support operations that could continue for more than two decades after launch. Planning for the preservation of key components, custom software & hardware, and the preservation of the operating environment within which these systems are designed to function represent a significant effort. The cost of this effort in the early stages of the program is expected to result in significant cost savings in later years. Quantifying these savings is not possible during the early years, however—quantifying the cost of recurring software development throughout the mission could be.

Without planning to preserve the hardware environment needed by the software, systems would fail and eventually be replaced by more modern systems. These new systems would provide enhanced performance by virtue of their newer operating systems, but the new OS is not guaranteed to provide functionality identical to the old OS, thereby necessitating a software rewrite to maintain operational capability. This situation will cascade, as every workstation must then be upgraded to maintain compatibility in a large ground system environment. Even worse, this cycle can be expected to repeat every 5 years or so. Clearly this situation shows the potential cost associated with the absence of longevity planning.

The New Horizons mission is being designed at the Johns Hopkins University Applied Physics Laboratory in Laurel, Maryland. New Horizons is planned to launch in January

of 2006, with encounter planned for the 2015–2016 timeframe. Longevity planning for this mission has become a significant issue during several recent program reviews.

Acknowledgments

The Authors of this paper wish to acknowledge the efforts of, and the inspirations from members of the retro-computing world; specifically, Bob Supnik, and the SIMH group for their work leading to the HP 2000 simulation. We also wish to acknowledge the contributions of Dave Ruske, Bill Rowe, Nelson Grodzicki, Richard Dienstknecht, Lee Hart, and the COSMACELF group for their individual and combined efforts resurrecting 1802 software.

Contributing editor: McDonald R. Stewart, Doctoral Candidate in the Management of Science, Technology and Innovation, The George Washington University School of Business and Public Management.

References

[1] RCA Integrated Circuits Data book 4-76 pg, 692.

[2] The Microprocessor Based MAGSAT Command System, Ark L. Lew JHU/APL CP-077 February 1980

[3] Build the COSMAC "ELF" Part 1, Popular Electronics August 1976 Pgs 33-38

[4] SUPER ELF An 1802 based Micro Computer, Quest Electronics, 1977

[5] COSMAC ELF Users Group
http://groups.yahoo.com/groups/cosmacelf

[6] QUESTDATA Newsletter Archive
http://groups.msn.com/
RCACDP1802CosmacComputers/newsletters.msnw/

[7] http://groups.yahoo.com/groups/files/ElfTools.zip

[8] http://huizen.dds.nl/~gnupic/assemblers_pseudosam.html

[9] Lee A. Hart, Technical Micro Systems, Inc.

[10] PREPARE TIME-SHARE BASIC
http://oscar.taurus.com/~jeff/2100/hpbasic/index.html

[11] "25 Years of Real-Time Computing"
http://www.interex.org/tech/csl/RTE/archive/poyner1.htm

[12] The Computer History Simulation Project
http://simh.trailing-edge.com/

[13] HP 2100 Simulator Configuration
http://simh.trailing-edge.com/hp2100.html

[14] HP 2000 Access Operator's Manual,
HP Part No. 22687-90005

[15] HP 2000 Computer System Sources and Listings Documentation, HP Part No. 22687-90020

[16] HP's IOP Implementations: 2100 vs 21MX Bob Supnik, 22-Nov-2002, http://simh.trailing-edge.com/docs/hpiop.pdf

[17] Operating and Service Manual, Processor Interconnect Kit, Sept. 1973 HP Part No. 12875-90015

[18] Systems Analyst's Seminar guide, Introduction to the 2000E, Al Pare, August, 1972

[19] 2000E Time-shared Basic System Operator's Guide August 1972, HP Part No. 02000-90049

[20] HP 20854A Timeshared Basic/2000, Level F, System Operator's Manual, Nov. 74, HP Part No. 02000-90074

[21] Electronic Records Archives Requirements Document (RD), Section 2.7.2 – Preservation
http://www.archives.gov/electronic_records_archives/about_era/requirements.html

[22] Gemeny, S. E., "Longevity Planning, A Cost Reduction Strategy for Ground Systems of Long Duration Space Missions", 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations (RCSGSO) 2003