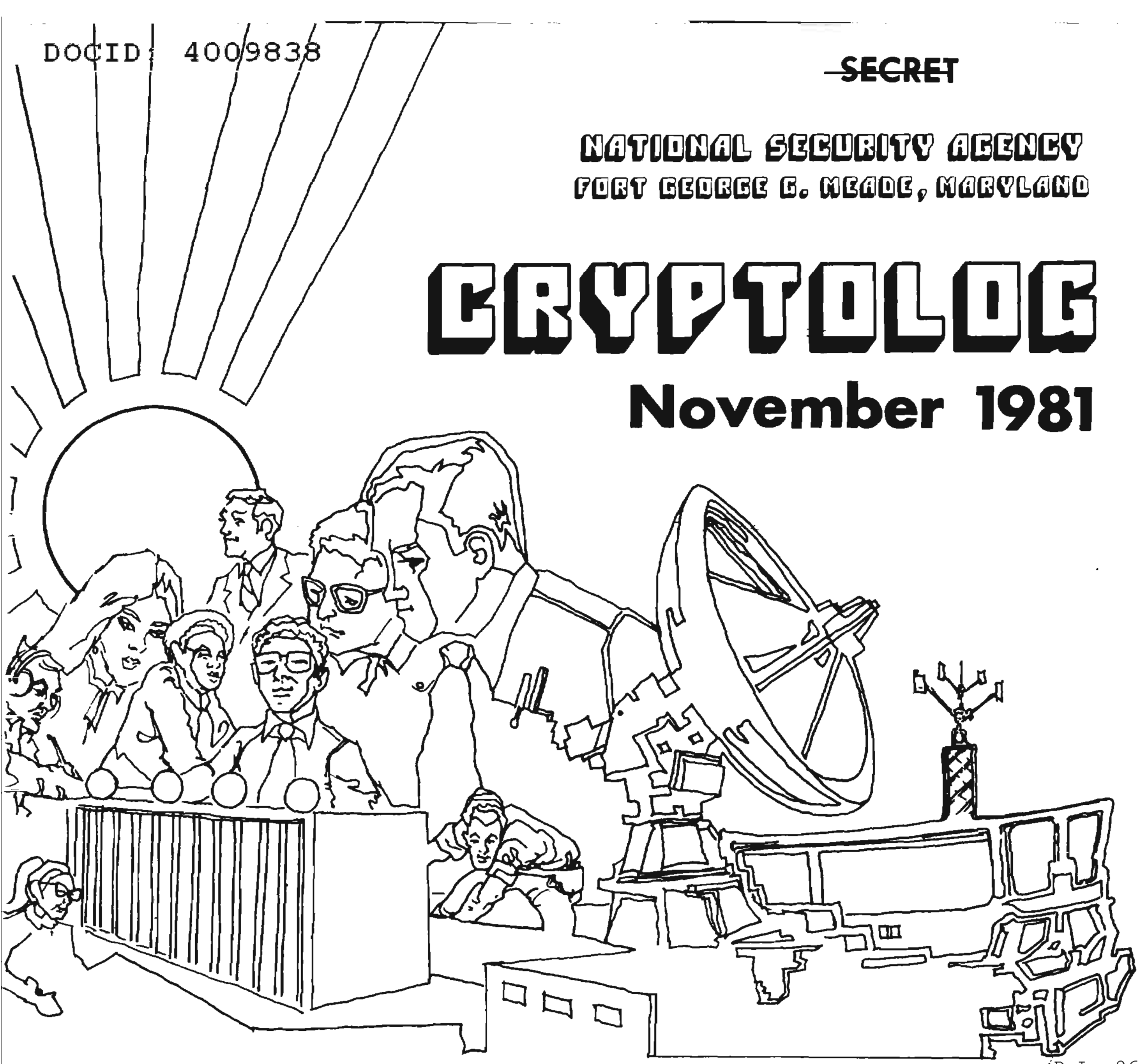
~~SECRET~~

NATIONAL SECURITY AGENCY
FORT GEORGE G. MEADE, MARYLAND

CRYPTOLOG

November 1981



P.L. 86-36

~~THIS DOCUMENT CONTAINS CODEWORD MATERIAL~~

~~SECRET~~

~~CLASSIFIED BY NSA/CSSM 123-2~~
~~REVIEW ON 10 Nov 2011~~

CRYPTOLOG

Published by P1, Techniques and Standards,
for the Personnel of Operations

VOL. IX, No. 11 NOVEMBER 1981

PUBLISHER [redacted]

For individual subscriptions
send
name and organizational designator

to: CRYPTOLOG, P1
or call [redacted] 3369s

To submit articles or letters
via PLATFORM, address to

cryptolg at bar1c05
(note: no 'O' in 'log')

P.L. 86-36

EDITORIAL

The response to our cry for help in keeping our subscription lists up to date has been heartening. Thanks. One "active" name on our list has been out of the agency for several years; many others have moved.

Along with the responses, we have been getting questions along the line of "Is CRYPTOLOG still alive?" (we think so) and "Is it going to be merged with some other publication?" (we have no plans to merge with any other publication, and none has so far expressed any interest in merging with us). We are a DDO (Operations) publication, but it is clear from our subscription list and our author list that we range outside the physical confines of DDO.

If we don't seem to be publishing any articles about your area of interest, it is either because the editor is biased against your area of interest, or because he isn't getting anything about your area of interest that can be published.

Most of the layout and editing of CRYPTOLOG is now being done on a computer - actually on several computers. Using the UNIX system, with some help from PINSETTER, and the PLATFORM network between various host computers, the original keystrokes (often the author's) are retained throughout the process. A lot of retyping, as well as cutting and pasting that characterized the earlier issues (all done on a typewriter) is being avoided. One item, [redacted] [redacted] piece last month on Technical Support Catalogs, was coordinated with him in final form just before publication via the network (Ken is now stationed in Germany).

About half of the items now being worked on for this and future issues have come in over the network. We are still interested in receiving items from people not on the network. We don't mind typing, even though it's nice to have some items that don't require it.

FUTURE POWERFUL PERSONAL COMPUTERS:
An Overview of the Technology
by [redacted]

P.L. 86-36

THE PPC IS COMING

Scientific and analytic computing, especially at NSA, has evolved from the batch environment of the 1960's to the timesharing and multiprocessing environments of the 1970's. In the 1960's, typical programmers submitted a deck of punched cards to the batch system and later received the deck and a listing of the program execution. In this environment, both the computing power and the user's access to this power were remote and non-interactive. In the timesharing environment of the late 1970's (and of today), the programmer has direct, timely, interactive access to his or her computing processes through a terminal. In this environment, the computing power is still remote, whether in the next room or far away across a network, and is shared. However, the user's access to this power is potentially local and definitely interactive; hopefully, the access terminal is on or near the user's desk.

What will scientific and analytic computing be like in the 1980's? While it can be argued that very large-scale super computers like the Cray 1 will definitely be needed for many complex problems [10], advances in several areas of computer technology have spurred efforts to design and produce extremely powerful, extremely compact computer systems for scientific and analytic use. Such systems will be small enough and inexpensive enough to be single-user systems located at the user's desk. In a sense, these systems will enable users to have their own "VAX" or "370" instead of a terminal. In this environment, both the computing power and the user's access to it will be local, personal, and highly interactive.

The purpose of this paper is to discuss capabilities being proposed for such a computing system, how it may be realized, and what its impact on NSA scientific and analytic computing might be. How should this future system be described? Some papers on the subject call it a personal computer [4, 12]. While it will be personal, this label conjures up images of the TRS-80 or the Apple II — a totally inappropriate image. Other papers [13] refer to it as an intelligent terminal. At NSA, this term fits the Delta Data 7000, for it is a terminal with its own microprocessor. The powerful future system is NOT a terminal; it is THE computing system and may be more powerful than systems to which we interface intelligent terminals today! For lack of another name, this paper will refer to this system as a Powerful Personal Computer (PPC).

The PPC has the potential to revolutionize scientific and analytic computing at NSA. Even with the GTSS and other timesharing systems of today, analysts use terminals to gain access to remote, shared computing power and data over relatively low-speed connections (whether network or communications lines). The PPC will give the analyst access to significant local, individual computing power and data. Networks and communications lines today are used to gain access to all computing power, all data, and personal communications. In the PPC environment, high-speed networks will be used for access to very large data bases and shared resources and for electronic personal communications. This will be a drastic change from our present networking philosophy.

II. Characteristics of a Powerful Personal Computer

Although timesharing systems have given access to remote, general-purpose computing rather than to local, personalized computing, the environment which has been created is a rich one for timesharing users. This environment has promoted a large set of programming languages, large file storage capabilities, sharing of programs and data, a cooperative user community, and other benefits. The environment of the PPC should preserve and enrich the good characteristics of the timesharing environment, while bringing many totally new capabilities to its users.

A number of efforts are underway to specify and/or produce a PPC and its environment at institutions like Xerox [12], Carnegie-Mellon University [6], MIT [15], Convergent Technologies [16], and Three Rivers Computer Corporation [13]. While these efforts do not completely share common technologies, their broad goals are remarkably similar and these goals apply for many scientific and analytic institutions (including NSA). The individual PPC environment of the mid-1980's should be reasonably priced ($10,000 to $20,000), should exist in a small and attractive package for office use, and should have the following characteristics:

1. a very powerful processor or processors (while this will be implemented on one chip, the term "microprocessor" seems too limited); it should have 32-bit data paths and use 1-bit, 8-bit, 16-bit, 32-bit, and 64-bit operands;
2. a smoothly addressable virtual address space using as many as 32 bits of address;
3. a very large multiport primary memory-- 1 Mbyte or more;
4. at least 100 Mbytes of high-speed local secondary storage;
5. a 1024x1024 raster display, probably color with several bits per picture point (pixel);
6. good interactive devices (keyboard, graphics pointer, lights, function buttons);
7. audio input and output;
8. ease of interfacing other peripherals if desired;
9. a very high-speed local network connection;
10. a powerful local operating system which can be personalized;
11. powerful, easily-used programming languages, utilities, and DBMS techniques.

Given a PPC with the above characteristics, an office environment built around several such PPC's would have these additional characteristics:

12. a local, high-speed network connecting all PPC'S throughout the office;
13. a gateway to other networks;
14. an "office" PPC to support expensive peripherals which are needed occasionally (e.g., quality printers, massive disks) and to perform support functions (e.g., mass data transfers from distant data bases, local office coordination);
15. an "office" file system for commonly used databases;
16. a global (to the local network) operating system to allow easy inter-PPC sharing of programs, data, and resources.

Items 13, 14, and 15 could be implemented in a distributed manner on several PPC's across the local network or in a centralized manner using one physical PPC as the "office" machine to support all office resources. This paper will assume the latter implementation. The global or network operating system would be distributed.

The personal computing environment described above is more powerful in both hardware and software than almost all timesharing systems in use today. Should a computing environment that powerful really be used by only one person? Can institutions like NSA afford to allow such a powerful computing engine to stand idle between the keystrokes of its single user? Yes. Economics today show that the hardware to implement a PPC will be reasonably priced in the mid-1980's. Some estimates for a PPC as described above are in the range of $10,000 [6]. (Of course, an implementation today would be much more expensive.) Economics today also show that the people who do scientific and analytic computing are becoming more and more expensive. If such people become only moderately more productive when given a PPC, the investment is worthwhile. The cost of any "wasted" machine cycles is insignificant compared to the productivity gained.

While the packaging of the characteristics of the PPC in the form described will be a major effort, each characteristic by itself is not completely new. Each already exists in some form at some price. Thus the development of the PPC is more of a hardware and software engineering project, rather than a research project [6]. This does not mean it will be any easier; it simply means that the areas to be explored and developed are not un-known.

III. Hardware Technology for the Powerful Personal Computer

The combination of hardware and software technologies needed to successfully implement a PPC with the 16 characteristics listed in section II does not yet fully exist. It is important to point out here that we must have both advanced hardware technology and advanced software technology to successfully implement the PPC environment. One without the other will lead to failure. This section will discuss in some detail the hardware technology which will enable the PPC to be built; section IV will discuss the software technology which will enable the PPC to be successfully used.

The hardware issues fall mainly in characteristics (1) through (9) and (12). The technology exists today to supply the capabilities listed in these characteristics, but at substantial cost and in very large packages not at all suited for an office setting. One could attempt to meet these characteristics with the following set of today's standard hardware:

(1)-(4),(8) a DEC VAX 11/780 computer system ($160K);

(5)-(6) a Genisco or Ramtek raster graphics system ($30K);

(7) input-- 64-word vocabulary system by Heuristics ($259);

output-- VOTRAX voice synthesizer ($3K) or Texas Instruments TM990/306 179-word system ($1K);

(9),(12) an ETHERNET or Mitre bus system ($6K);

If this hardware configuration were assembled, it would cost about $200K and would require about 200 square feet of floor space and special electrical connections and air conditioning. It would not be suitable for a personal system on one's desk.

As LSI and VLSI circuit design technology continues to make advances, the hardware pieces needed to satisfy these requirements will continue to get smaller and less expensive. The remainder of this section will explore coming technological advances which will help realize the PPC.

A. Processor and Address Space

If the PPC is to truly give its users the power of current multi-user machines like the DEC VAX 11/780 or the IBM 370, its processor must have a powerful instruction set, must be fast, must have a large address space, and must have wide internal data paths. Single-chip processors of the late 1970's (traditionally called microprocessors) have not met these criteria. Although their instructions sets may have been reasonable, their execution speeds have been moderate, internal data paths have been either 8 or (sometimes) 16 bits wide, and direct addressing has been limited to 64K bytes of memory. Because the term "microprocessor" has been traditionally associated with these earlier single-chip processors, it is inappropriate when discussing the type of processor needed for the PPC of the mid-1980's.

The newest generation of single-chip processors has made several major advances over the earlier generation as LSI technology has grown. As technology continues to grow, further advances are sure to come. Before discussing what the mid-1980's may produce for single-chip processors, a look at current state-of-the-art processors is in order, since these processors are being used in current projects to implement PPC's. See Figure 3 for a quick comparison of the Intel 8086, the Zilog Z-8000, and the Motorola M-68000.

All three of these processors have been built with some concern for the operating systems and higher-level languages that must run on them. Thus, they have instruction sets to support byte-string operations, bit manipulation, re-entrant code, dynamic relocation, etc. They all have well-designed interrupts, register sets, and other expected hardware features. Although a ranking of the three may not be fair, their applicability for a PPC processor could be ranked in decreasing order of applicability as (1) M-68000, (2) Z-8000, (3) 8086. At this point in LSI evolution, the capabilities of the processor chip will depend heavily on the surrounding support chips and coprocessors. When the processor chip of the mid-1980's includes many functions which are now off-chip, this will not be true.

LSI state-of-the-art technology in 1980 puts about 70K devices on a chip to produce a Motorola MC-68000. VLSI technology (VLSI is usually accepted to mean 100K or more devices per chip) will greatly impact the development of more powerful single-chip computers because cf increased design density, increased chip size, and improved layout techniques [7]. VLSI state-of-the-art in about 1985 will put 1M devices on a single chip. The single-chip processor of 1985 (dubbed P1985 in [7]) will be a much more powerful one than that of today. When the P1985 architecture can be realized, a single-chip processor will indeed be equivalent in functionality to many large commercial CPU's of today (e.g., the VAX 11/780). With such a processor, the PPC as described in this paper will be realizable.

B. Primary Memory

Given a good virtual memory operating system for the PPC, significantly less physical memory is required than could be supported by the address space. However, the amount of physical memory to nicely support multitasking and to provide image memory for the raster display is still significant. Because memory chips will be very inexpensive in the mid-1980's, a primary memory on the order of 1 Mbyte will be an economically sound way to reduce local operating system swapping overhead. Image memory for the raster display could take an additional 0.1-0.5 Mbytes, depending upon the choice of black-and-white or color.

With present, proven 16K-bit memory chips, it would take 500 chips to provide 1 Mbyte of primary memory; this would occupy several physical boards (perhaps 10) and would occupy too much space for a PPC. With the 64K-bit chips now coming into production, only 125 chips are needed and they can be configured in a much smaller package (perhaps two boards). With 256K-bit chips on the horizon [4,7], this shrinks to approximately 32 chips. Depending upon other design considerations, this entire 1 Mbyte memory might be placed on the processor board, considerably reducing packaging size. Texas Instruments predicts that these components will be available by 1985 at a cost of less than $2000 for the 1 Mbyte capacity [4].

C. Secondary Storage

For the PPC environment to be successful, a high-capacity, fast secondary storage system is needed at the individual PPC to hold personal utilities, programs, data, and text files. For this storage system to fit neatly into an office environment, it needs to be compact. In an office with several PPC's and an "office" PPC networked together, the "office" PPC may be required to supply additional bulk secondary storage. That can be done with more traditional disk systems and will not be considered here.

Examining current work in storage technologies shows advances in charge-coupled devices (CCD's), magnetic bubble memories (MBM's), video disks, AND magnetic recording [3]. At first glance, one might be inclined to discount magnetic recording as a continuing attraction for mass storage. However, many of the same technological advances that are advancing CCD's and MBM's are also advancing the state-of-the-art in magnetic recording. In the past 25 years, device capacities have increased over 100-fold and recording densities have increased over 1000-fold; similar dramatic advances continue to be predicted [3]. Especially with the introduction of devices like the Winchester disk, which can store in excess of 30 Mbytes of data on an 8-inch platter for about $2500, it seems that magnetic recording will be the appropriate technology for the PPC.

D. Raster Graphics and Interactive Devices

In the timesharing environment of today, a raster graphics system is a peripheral that is often used in conjunction with a more standard alphanumeric systems terminal. In the PPC environment, the graphics display will be the ONLY visual presentation to the user and it will be an integral part of the PPC, not a peripheral [11]. Thus, use of the graphics display will be an inherent part of any program which interacts with the user. The PPC raster display should have the following features:

high resolution -- the display should have approximately 1024x1024 addressable picture elements (pixels);

frame buffer -- the image memory (bitmap) should be organized as a frame buffer which can be accessed on a pixel basis directly by the PPC processor; the frame buffer should be seen as main memory by the PPC processor;

graphics processor -- functions like vectors, characters, and other graphics primitives should be implemented by either a special graphics processor or by special microcode for the PPC processor;

color -- depending upon the amount of PPC memory to be devoted to the frame buffer, color could be an option; if chosen, at least four bits per pixel should be used with a video look-up table for greater color definition [11]; 40-60 Hz refresh is desirable;

video I/O and processing -- digitized video input to the frame buffer should be possible; under control of a video processor [11], output from the frame buffer to the screen could be zoomed, scrolled, pseudocolored, etc.;

keyboard -- a flexible keyboard is needed which reports to the PPC processor which specific key is depressed, not a specific ASCII code; this allows total redefinition of the keyboard by the program;

pointing device -- a pointing device with dynamic cursor is needed for accurately indicating positions on the screen by the user.

All of these features are available in present commercial raster graphics systems which are tied to present computer systems as peripherals. The technological issues which must be resolved to put these features in a PPC are two: size and integration. The bulk of current color raster graphics systems is physically in the image memory, the interface to the host, and the graphics processor. In the PPC, graphics will be integrated into the entire package; it will not be a peripheral and no interface is needed. The image memory will be organized as a part of the PPC's main memory. A separate graphics processor is not needed if the PPC's microcode supports primitive graphics functions. If not, a graphics processor in this technology would be extremely small. Thus, if the integration of raster graphics into the PPC is done correctly, size is not an issue. Even with a separate graphics processor and a sophisticated video processor, the extra hardware associated with the raster graphics should be confined to one board at most.

E. Audio Input/Output

The concept of talking to your PPC and having it talk back to you may seem far-fetched and perhaps unnecessary, but audio I/O seems very attractive from a human factors point of view. Advances in heuristic techniques for speech recognition, advances in LSI, and the home computer market have been driving forces in producing the audio I/O devices available today. Several companies now offer speech input and output systems for under $1,000 each. The popularity of the Texas Instruments "Speak and Spell" toy attests to the value of audio I/O.

F. Interfacing

Given an office environment with a number of individual PPC's and an "office" PPC to support a large office database and a high-quality document printer, extra peripherals for an individual PPC may not seem needed. However, given the diversity of talents and

interests which may use the environment, some new device will soon be suggested as a peripheral to a PPC. When that time comes, the interface to the PPC should be straightforward and easy. The hardware interface could be via a standard communications port or directly to the internal bus of the PPC; both should be available.

G. High-speed Local Network

High-speed local networks exist today. There are several different configurations of topology, control structure, and transmission media which can be chosen, depending upon the applications and the distances involved [1]. For an office PPC environment, a ring or bus topology (see Figure 4) with a contention control structure seems promising [1,15]. Examples in current technology include the ETHERNET and the Mitre bus.

IV. Software and Environment Technology for the Powerful Personal Computer

Very strong emphasis must be placed on the software and the environment for the PPC. If the hardware technology described above is successful beyond our wildest dreams, the result will not be practically useful without an equally successful software technology. If hardware technology can be viewed as supplying the raw power needed, the software and environment supply the ease of use and control necessary to harness and direct that power.

The user interface to the PPC is all important. The hardware technology discussed above can provide interactive and communications devices with very interesting human factors implications. A PPC which can listen to you and talk back, draw colored pictures for you, and communicate with others in your office for you could become a very powerful extension of yourself. However, the software and environment of the PPC must be carefully constructed for this potential extension to become reality. The use of audio I/O, color displays, and the local network must be innately a part of all software components. If these capabilities are thought of as occasionally desired peripherals, rather than as an integral part of the system, the resulting environment will be much less human and less powerful than it could otherwise be.

A. Operating Systems

The local operating system will be the primary interface between the user and his or her PPC. It should be friendly, easy to use, helpful, and as forgiving and tolerant as possible. It should support a multitasking, virtual memory environment with interprocess communication. Any hardware feature of the PPC should be as useable as possible from the operating system level.

In the PPC environment with a high-speed local network, resources used by a given task may be distributed between the personal PPC and the "office" PPC, or they may not. The location of resources (files, peripherals, gateways, etc.) should ideally be transparent to the user. In order for this to happen, a global or network operating system must exist to coordinate this communication and resource sharing. Depending upon implementation, it could reside on the "office" PPC or be distributed throughout the PPC's in the office network.

B. Programming Languages

Programming languages will be the second interface between the user and the PPC. Programs will be one of a user's major products. Programs must be coded, modified, debugged, made efficient, and (finally) executed. A programming language and its surrounding environment should be designed to facilitate this process and to make it as pleasant and efficient as possible. Alan Kay and his SMALLTALK work on the Xerox ALTO system [2] have shown that novice programmers can quickly become proficient if the programming language is designed appropriately.

Since the PPC as described herein is designed for scientific and analytic programming, the proposed users are not totally novice. However, languages for the PPC should be designed for people who traditionally think of themselves as non- programmers. A number of current languages are often proposed for use as a basis for a PPC programming language: PASCAL, C, ALGOL, Ada. The environment built around a language should support a compiler, linker, powerful symbolic debugger, and extensive runtime library.

C. Utilities

Utilities are normally invoked by operating system commands. They include things like an editor, various word processing programs (speller, formatter, etc.), a file system, language compilers and interpreters, debuggers, an electronic mail system, etc. These utilities MUST be implemented with the total PPC environment in mind. The text editor should take full advantage of the raster graphics for font definition, color, and perhaps illustrations. All utilities that could use the local network and any gateways to other networks should use them as transparently as possible. As with the operating system, the utilities should be as friendly, easy to use, helpful, forgiving, and tolerant as possible.

D. Servers and Network Gateways

In a local network PPC environment where users interact with one another frequently, the concept of servers has proven important [12]. In this environment, a server is a machine on the network which performs some widely-used service for all users who desire to use it (e.g., document printing). In some network environments, there are several or many servers distributed around the network. In the PPC environment, one server has been postulated, the "office" PPC.

The concept of gateways to other networks is especially important at NSA. If a PPC local network becomes a replacement for current GTSS systems, the interconnections currently supported over PLATFORM would need to continue. The local server or "office" PPC would handle PLATFORM-like communications for overall network mail and file transfers.

E. Environment

When the software described above is implemented, the environment created for the individual scientific or analytic user will be very powerful, extremely easy to use, and tailorable to closely suit the individual's personality and needs. Even though audio I/O and color graphics are integral tools at all levels, it is obvious that some people will use them and others will not. Some people will make constant use of the local network and others will generally remain in the shell of their own PPC. The overall PPC environment should be flexible enough to gracefully allow use of all, some, or none of these special features. It should gently encourage their use without penalizing a person who insists on using only the keyboard input and alphanumeric text output. A Powerful Personal Computer must be just what its name implies: powerful, yet personal.

V. NSA and the Powerful Personal Computer

It will take the research and industrial community several years to complete a commercially-available PPC which meets most of the specifications of section II. Such a system may not be available until 1985, if then. Before 1985, several versions of a PPC will be available in one of two forms: (1) a commercial form which uses 1980-1981 technology to meet many of the section II specifications or (2) a research form which meets all of the specifications. The commercial versions will be realistically available in 1981-1982 with the necessary software. A more powerful research version with newer technology might be available in 1984-1986.

Given that these predictions come true, what should NSA be doing to prepare for the advent of the PPC? NSA should be planning for it and experimenting with those versions of the PPC which will soon be available. Several offices in NSA (e.g., R53) are now using timesharing systems in a very personal way. Terminals are at the users' desks, various inter-user communications systems exist, the computer serves as phone book and personal text preparation system, and many working documents are kept on the system. Most importantly, the users of the system have adapted their way of life around the system in personal ways; they have made the system an integral part of their work environment. Such offices are excellent candidates to experiment with the PPC environment.

Parts of the DDR and DDT organizations are already closely following the development of the PPC externally. R53 is now assembling an initial prototype system for experimentation in the use of PPC's. This system will initially include two PPC systems from Apollo Computer (built around the Motorola M-68000), one system from Convergent Technologies (built around the Intel 8086), and a high-speed local network built by Sytek, Inc. The present R53 timesharing resources will be integrated into this system via the local network. This total PPC environment in R53 will be used in part to gain experience and to help determine possible architectural configurations for the T4 User Interface System project.

What areas of NSA are likely candidates for a PPC environment? Problems where massive amounts of computational power must be applied will still require systems like the CDC 7600 and its successors [10]. However, algorithm development for these problems is an excellent candidate for a PPC environment. Environments which now use the Generalized Terminal Subsystem (GTSS) timesharing concept are obvious candidates for the PPC. Any scientific or analytic computing would be a candidate for the PPC. Non-technical functions like word processing may eventually benefit from the PPC environment, depending upon the final cost of the PPC and the coupling between technical and administrative people within an office. In short, any computing environment where people are doing interactive computing or algorithm development is a candidate for the PPC environment. Thus, NSA has a lot to gain in productivity from successful development and application of a Powerful Personal Computer environment.

References:

1. Clark, David D., Kenneth T. Pogran, and David P. Reed. "An Introduction to Local Area Networks," PROCEEDINGS OF THE IEEE, vol. 66, no. 11, November 1978.

2. Kay, Alan C. "Microelectronics and the Personal Computer," SCIENTIFIC AMERICAN, vol. 237, no. 3, September 1977.

3. Hoagland, A. S. "Storage Technology: Capabilities and Limitations," COMPUTER, vol. 12, no. 5, May 1979.

4. Isaacson, Portia, Robert C. Gammill, Richard S. Heiser, Adam Osborne, Larry Tesler, and Jim C. Warren, Jr. "Personal Computing," COMPUTER, vol. 11, no. 9, September 1978.

5. Morse, Stephen P., William B. Pohlman, and Bruce W. Ravenel. "The Intel 8086 Microprocessor: A 16-bit Evolution of the 8080," COMPUTER, vol. 11, no. 6, June 1978.

6. Newell, Allen, Scott Fahlman, and Robert Sproull. "A Proposal for Personal Scientific Computing," Department of Computer Science, Carnegie-Mellon University, Draft of 13 July 1979.

7. Patterson, David A. and Carlo H. Sequin. "Design Considerations for Single-chip Computers of the Future," IEEE TRANSACTIONS ON COMPUTERS, vol. C-29, no. 2, February 1980.

8. Peuto, Bernard L. "Architecture of a New Microprocessor," COMPUTER, vol. 12, no. 2, February 1979.

9. Stritter, Edward and Tom Gunter. "A Microprocessor Architecture for a Changing World: The Motorola 68000," COMPUTER, vol. 12, no. 2, February 1979.

10. Sugarman, Robert. "'Superpower' Computers," IEEE SPECTRUM, vol. 17, no.4, April 1980.

11. Tarbell, Lawrence C., Jr. "The Potential Impact of Raster Graphics at NSA," PROCEEDINGS OF THE CISI SPRING CONFERENCE, May 1979.

12. Thacker, C. P., E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs. "Alto: A Personal Computer," Xerox PARC report CSL-79-11, 7 August 1979.

13. Three Rivers Computer Corporation. Preliminary documentation for the Perq Intelligent Terminal, enclosures to correspondence dated 13 June 1979.

14. Trifari, John. "Backing Up the New Winchester Disk Drives," MINI-MICRO SYSTEMS, vol. 12, no. 8, August 1979.

15. Ward, Steve, Chris Terman, Jon Sleber, and Rae McLellan. "NU: The LCS Advanced Node," MIT Internal Memorandum, 28 February 1979.

16. Wegbreit, Ben. Private communication of 23 April 1980.

FUTURISTIC REPORTING (u)

by [redacted]

P.L. 86-36

Reporting describes a wide variety of different activities. Within NSA, we serve our customers in many different ways. We issue informal ~~(S-CCO)~~ reports such as TACREPs, formal reports and translations in both hard copy and electrical versions, and we support community data bases through the COINS system. Among our reports and translations, which form the bulk of what we generally refer to as reporting, we cover many different subjects and [redacted] I could go on listing the variety of reporting for some time. The variety of reporting we do requires a lot of different ways of doing it. We tailor our product to some extent today and are always looking for ways to improve it, but we do almost all our reporting via electrical, narrative reports: few hard copy reports, few graphics, few briefings.

~~(C-CCO)~~ With the availability of computer technology, many parts of NSA's mission have been affected. Where we used to copy Morse code on six-ply paper, and the analysts back at NSA would scan the raw traffic a month or two after it was intercepted, today we routinely forward traffic electrically to NSA for nearly instant analysis. [redacted] Traffic analysis, signals analysis, cryptanalyis, telemetry analysis are all done routinely on computers.

(U) With the advent of the MESSENGER computer system in NSOC, we even prepare reports on a computer. But wait a minute. Let's consider how that is done. The reporter scans his incoming traffic, prepares his report, submits it to a chain of reviewers who eventually deliver the draft to the typist in the flex room. The text is retyped (possibly for the fourth or fifth time) and then released. Although MESSENGER is a computer based report preparation system, it only performs the typing and releasing functions and does not serve the person who actually prepares the report - the reporter.

(U) Will we solve this problem, this lack of support to the reporters of NSA, in the next decade? I certainly hope so, and I think it is well within our power to do so. The technology is available today to enhance the reporters', function beyond the wildest dreams of most reporters. Many people recognize both the problem to be solved and the means of solution, and in several areas, they are already working to develop computer systems to serve reports in the preparation of reports.

EO 1.4.(c)
P.L. 86-36

(U) This paper will attempt to describe some of the problems which are inherent in the reporting field, both those now felt by reporters and those which must be addressed in the development of a reporting computer system. It will then look at current projects underway which are developing computer systems capable of supporting reporters. Some of these systems are not intended to serve reporters but could do so with little additional effort. We will look at the possibilities available for reporting computer systems given today's technology, and then discuss some of the ways in which future technology might further enhance a reporter's life. Finally, we will discuss some of the possible changes in the structure of the reporting field caused both by the computer itself and by policy changes in the intelligence community.

Problems To Be Addressed (U)

~~(C)~~ In today's intelligence reporting world, we have reached the dubious position of inundating our customers with reports. We have a great need to tailor our reporting more carefully so that the important pieces of information which U.S. decision-makers need are not lost in the sea of information we are capable of producing.

(U) We have always had a problem insuring quality control. No one wants to publish an erroneous report, but sometimes we don't have time to check all the facts. Sometimes the typist introduces an error into a report that was already carefully checked. And sometimes out intelligence sources present us with erroneous information in the first place. We have established over the years a complex coordination-review process aimed at getting anyone who has information relevant to the subject of the report involved in the production of the report. Of course, this coordination takes time. Sometimes the reviewer makes changes that are wrong, and doesn't take the report back to the originator. Sometimes the report has to be revised heavily and therefore must be retyped from scratch.

(U) In the research that goes into an NSA product report, there are a number of onerous tasks that must be performed, which must seem to many reporters as needlessly time-consuming: checking the spelling of placenames, finding the coordinates (because the report goes out electrically and has no maps), getting people's names spelled right, conforming to the myriad regulation about format, preparing the coversheet so that the accounting system will work. All these are tasks which must be done but are not a part of what the reporter thinks of as his primary function - presenting relevant facts to a customer so that the customer can make informed decisions on behalf of the country.

~~(C-CCO)~~ Accounting for what we do is very important to the proper management of limited resources. We are now trying to connect formal requirements and their satisfaction through the use of computers, but reporters are finding that it takes a lot of extra time to prepare the complex coversheet that puts the needed data into the management program. And that doesn't include the time spent keypunching all that information. Hooking the reporter to the computer could save time and at the same time eliminate the keypunching.

(U) How fast can we report information? How fast do we need to? Without trying to definitively answer those questions, let us say that there are numerous times when our reporting was not fast enough to suit the situation. On a limited basis, we have the capability to speed up reporting but it takes a heavy toll in resources. How much of the delay is in the report research and preparation process itself? Some might not agree with me but I would claim that today most of the delay is in the report preparation process. We have enabled intelligence to be expeditiously intercepted and decrypted, but we have made little progress in translating and reporting quickly.

~~(C-CCO)~~ Some problems which are introduced with the computer are the dependability of the computer system, and the security of the data. If the computer goes down, do we have all our analysts sit on their hands until it comes back up? And in the security area, we have potentially horrendous problems. We have built up over the years an incredibly complex system of compartments, codewords, and clearances, to the point that many people don't know which things they are cleared for. Can the computer help us deal with this problem, or will the potential for inadvertent access to someone else's data exacerbate the problem beyond belief?

Current Systems and Projects (U)

(U) There are almost as many covernames in NSA as people, and a person could be forgiven if he got confused. But let's look at a few of the names in the field of analytic computer systems.

P.L. 86-36

[redacted]

[redacted]

Possibilities With Present Technology (U)

(U) The TRS-80 and similar so-called "personal" computers have more than enough power to satisfy most reporter's needs today. With communications interfaces, such computers provide the technological basis for a reporting computer network capable of revolutionizing the way we do reporting at NSA.

[redacted]

(U) It goes without saying that there are a number of problems that would have to be solved before this utopian picture can be developed. The present tubes are rather small and can only display a limited amount of information. The present computer systems have numerous problems with both turnaround and dependability. A couple of developments in the computer field may help in this area: failsoft technology, in which pieces of a computer can work independently of one another; and distributed processing, in which each user or small group of users has an independent computer tied to other computers only for data transfer. All the problems are solvable. The major question is whether our institution will solve the various problems; whether it, or we, are committed to improving the effectiveness of reporters through the use of computers. Can we? Will we?

Future Technology (U)

(U) The TRS-80 has been called the "Model T" of the computer industry. The era of cheap computing power is here today, and the pace of technology development has been increasing for several years. There is no reason to expect anything other than a continuing fast development rate in new computer technology over the next ten years.

(U) Already there are rumors flying, about computers that will interpret the spoken word. Can you imagine simply talking to your computer terminal, to give it instructions or to "write" reports? Undoubtedly, such machined will be on the market in the next few years.

~~(S-CCO)~~ Developments in microtechnology and high speed computer circuits promise to produce desktop computers with more power than 25 Cray-1's. (The Cray-1 is the fastest general purpose computer available today, and sells for about $10 million. We use a Cray-1 to attack the most sophisticated cipher systems.)

(U) More compact terminals may result from developments in the plasma display field, using a flat display instead of a cathode ray tube. This will make the "terminal on every desk" concept more practical. Combined with the extra power available, each user might have a complete processing system on his desk, tied to a central system only for data transfers.

(U) High quality facsimile transmission at a reasonable cost is just around the corner. With consumers tied to NSA through a facsimile/data network, "electrical" reports with graphics become possible. Our local computer might help to generate the maps, requiring no more instructions than a list of the placenames to be identified. Charts and graphs will also be practical in such a system.

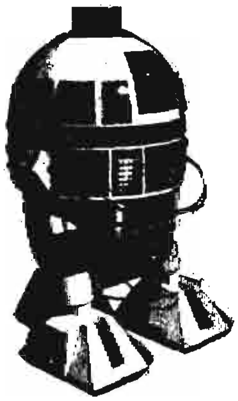
Possible Re-Structuring of the Reporting Function (U)

~~(C-CCO)~~ Last year, NSA was studying a system that could result in a massive restructuring of the way in which reporting is done. This was not generated by technology but rather represented an attempt to simplify the world of codewords for intelligence consumers. The program, called APEX, was a matter of some confusion here at NSA. APEX called for "decompartmentation" of intelligence, meaning generally the sanitization of material so that it could be distributed without codewords. APEX is now dead, but some of the ideas contained in the project live on. There is still a hugh demand for sanitized SIGINT. This might mean producing some additional reports with detach lines, or it could involve producing two versions of a single report, one of them sanitized.

~~(C)~~ Imagine, if you will, applying such a system to all SIGINT reporting. I am sure you will agree that a computer would be an invaluable tool in editing and reviewing reports which must be sanitized for wider distribution. If we stop and take a look at the possibilities, we might even be able to redesign the reporting system with an eye on the technology, and take advantage of the technology instead of using it to play catch-up.

~~(C)~~ Ways in which we might restructure the reporting function to take advantage of technology include putting more information into data bases, and making more of that data base information available to users at multiple access levels through sanitization. We might tailor our reporting to fit the needs of individual users, by having the computer scan the available intelligence information and select items by using a dictionary of relevant terms. Our requirements process might be different in that consumers could simply input their keywords into their computer terminals, instantly updating the requirements dictionary. The NSA system could automatically compare the consumers' input and access level with the available NSA information and route the appropriate information as it becomes available. Management reports could be available instantly on which user requirements were being satisfied and which were not. Analysts at NSA might refer to the unsatisfied requirements data base to help them prioritize their workload. Supervisors might use it to assign work to analysts. This information might even be used to alter our tasking of intercept resources on a real-time basis. [redacted]

EO 1.4.(c)
P.L. 86-36



Conclusion (U)

(U) In conclusion, let us consider the challenge of the 80's: to integrate technology that is available, and that is becoming available, to improve the efficiency and the effectiveness of the reporters at NSA, and to alter the ways we do our reporting to better serve our present and future consumers within the budgetary constraints place on us. My contention is that we can do our present job of serving intelligence consumers better and more cheaply by taking advantage of the existing technology. The decreasing number of secretaries at NSA is already a problem, and one that seems unsolvable. Lack of staff people to coordinate reports and a continuing need to maintain quality control create pressures to use computers to assist the reporting staff in maintaining the quality for which NSA is renowned.

(U) How to do this? First, we need a coherent policy regarding the use of technology to serve the reporting function. If my contention is correct - that technology can enable us to do our present job better and more cheaply - then we are wasting valuable resources by our continuing failure to use the technology available to us already.

(U) Second, we must have a driving force. The purpose of this paper is to pull together ideas from throughout NSA and from the computer world, and to present them to reporters and managers in NSA as a meahs of helping to create such a driving force - namely, the reporters and managers in NSA. If we sit around waiting for the T organization to recommend new and better uses for technology in support of the reporting function, we will grow old and gray before anything happens. This is not intended to be an indictment of the T organization, merely a statement of the realities of life. The T organization exists to serve the other organizations of NSA, including Operations. If we want to update the reporting technology here, we must ask - demand - that it be done. And we must take an active role in specifying in great detail how the job is to be done.

(U) The possibilities are almost endless, but the challenge is ours. The technology is there and more is coming. But we must take an active role in developing our understanding of how the technology can help us and in seeing to it that we make the fullest possible use of the technology.

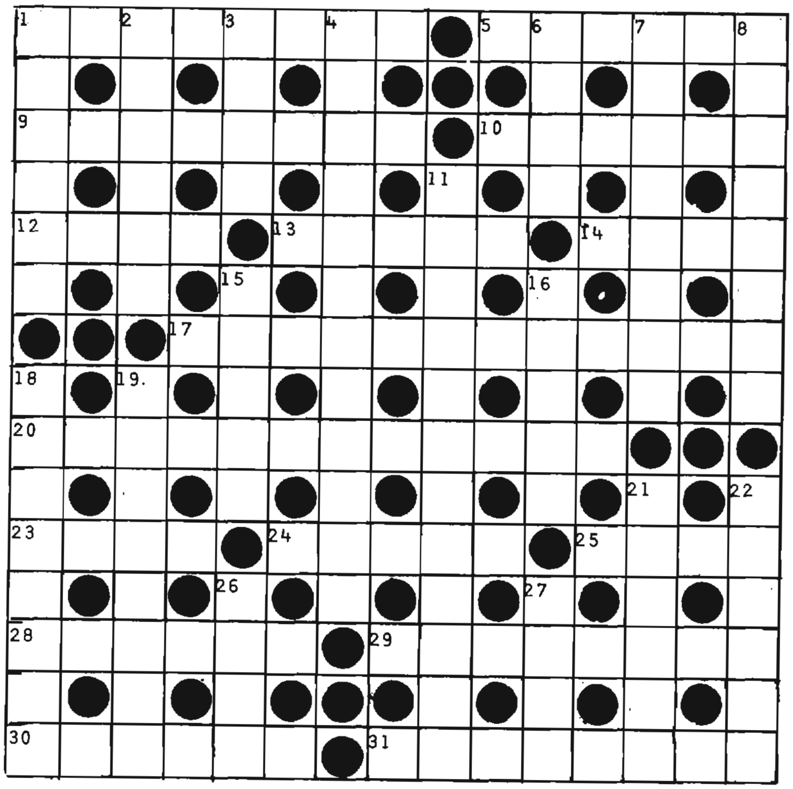
CRYPTIC CROSSWORD

BY [redacted]

Puns.
Anagrams.
Constructions.
Double definitions.

[redacted] *CRYPTOLOG's Information Science Editor, has constructed this excellent crossword puzzle, done in the cryptic, or British, style. (But beware of 5 Across; that one's pretty awful.)*

P.L. 86-36



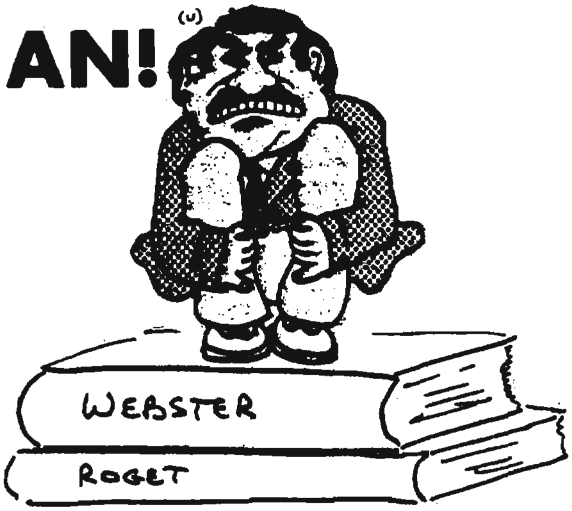
ACROSS

1. Average second bites are vegetables. (8)
5. Could this be where amputated limbs are thrown? (6)
9. Small overthrows combined in great poems. (8)
10. A child or a place in Mexico. (6)
12. Consumes teas in a sloppy manner, (4)
13. Weasel out of orders to attack. (5)
14. A kick from the team's top untried draft choice. (4)
17. I hear the sea left the fiend's farm machine so he could show us his wares. (12)
20. Otherwise calm mediator possessing emotional appeal. (12)
23. It's not often the meat isn't overcooked. (4)
24. Less confused about a tradition at Easter. (5)
25. A peachy coat for policemen? (4)
28. Myth of the ankle, perhaps? (6)
29. Religious gentleman takes a note back to provide work for the secretary. (8)
30. Did the bug clear his throat for the *Star Spangled Banner*? (6)
31. Snake has Richard Henry confused about a small prisoner. (8)

DOWN

1. A sentry stationed at the fence? (6)
2. The right side ejects from bed. (6)
3. Somehow the line forms in the river. (4)
4. *Another* ewe into paint; need we ask again? (8, 4)
6. Enough space to anchor around. (4)
7. The faction favoring pipes put together the merchandise. (8)
8. I sort art for Arnold's kind, among others. (8)
11. Gather #50 scheme for the European Recovery Program. (8, 4)
15. Rushes, we hear, through the book. (5)
16. Draw off five hundred droplets. (5)
18. Destroy the rum label? No! It'll be useful next April. (8)
19. Very good! Every one is not left behind. (3, 5)
21. Roman god adds eyes, we hear, to restore a flat. (6)
22. Blossoms as a sound heard over the meadow? (6)
26. At one time, at the induction center. (4)
27. Scandinavian in a Japanese rickshaw. (4)

SAY What You MEAN! (U)



by David W. Gaddy

(U) Over time, jargon becomes accepted usage, but in its application there can be confusion of meaning. "Sanitization" is an example. Part of the confusion arises from our traditional view of the COMINT handling system, a view which must be modified if we are to communicate effectively among ourselves and our colleagues in the Community. What follows is a reflection on "lessons learned" during the APEX study of the past two years and an attempt to clarify terminology now in wide (but often differing) use.

~~(C-CCO)~~ To set the stage by stating the obvious (so obvious it may be overlooked), the body of information under discussion, COMINT, is classified. It is TOP SECRET, SECRET, rarely CONFIDENTIAL. (Unthinkable a few years back, there is also unclassified COMINT of a historical nature, but we are concerned here with current COMINT.) It also has at least one additional attribute: it usually has a codeword or a restriction that it be handled only in COMINT channels, the COMINT "compartment." This is the information which, from World War II U.S. Army usage, is frequently called "special intelligence," or SI. Since SI is now limited to COMINT, it has become a euphemism - some even mistakenly (but with the same result) think it equates to "signals intelligence."

~~(C-CCO)~~ For years most of us have thought of "compartments" as those small, cloistered efforts, usually distinguished by a covername, which are now, for the most part, covered under the VRK (Very Restricted Knowledge) system. (See USSID 16 for details.) It still comes as a mild shock to be reminded that the COMINT handling system is itself a compartment (or "special access program," as compartments are termed in Executive Order 12065). Here is the definition of "compartment" as developed under APEX and approved by the DCI/NFIB. It represents but a slight modification of that contained in the 1978 DCI/NFIB "Glossary of Intelligence Terms and Definitions":

> compartmentation: Formal systems of restricted access established and/or managed by the Director of Central Intelligence (DCI) to protect the sensitive aspects of sources, methods, and analytical procedures of foreign intelligence programs.

The generic term in Community use is "sensitive compartmented information," SCI. COMINT, or SIGINT handled under the COMINT system, is therefore a form of SCI. (Although we are concerned only with special access programs in an intelligence context, there are other such programs, especially those created for military operational purposes.)

[redacted]

EO 1.4.(c)
P.L. 86-36

EO 1.4.(c)
P.L. 86-36

[redacted]

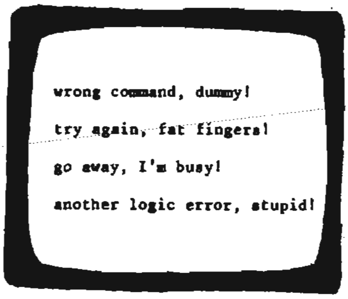
[redacted]

Ed Note: It may have been the conductor on the Orient Express who said it first, as the detachment of Turkish soldiers got off the train at Vienna: "I think we shall have to sanitize that compartment."

HOW TO CREATE A USER-*Un*FRIENDLY SYSTEM

by [redacted]

P.L. 86-36



wrong command, dummy!

try again, fat fingers!

go away, I'm busy!

another logic error, stupid!

(CISI Workshop, May 1981)

None of us sets out deliberately to bring unfriendly human/machine interfaces into the world. Somehow, even though we are trying our hardest to design and build good software, the user interface all too often turns out to have some serious defects when it is delivered and people start using it. Some of the defects can result in increased error, waste of expensive man-hours, and waste of machine resources. If the users have any choice, and things are bad enough, they may simply refuse to use the system, and find other ways to get their work done. If they don't have a choice, their work efficiency may suffer significantly. The worst aspect of the situation is that the losses are hidden; the machine is not down, some amount of work is flowing through the system, and there is no obvious stoppage or breakdown that can be singled out to warn us that a lot of time and effort may be going down the drain unnecessarily.

My main purpose here is to raise your consciousness about the needs of the user, and how some kinds of design decisions can affect the convenience and supportiveness of the user interface in an interactive system. I have a strong feeling, based on studies I have made of several Agency systems, that many of the unfriendly features are unnecessary. Some, it is true, are forced on us by prior commitments to specific formats or procedures, or by file security and file integrity requirements. The majority of the features that make problems for users come about, however, simply because the designer and programmer were optimizing other variables, without thinking about the effects on the user. Their priority lists are headed by other things, and the user is way down in the stack. They are concentrating on saving space, getting around weaknesses in the programming languages and operating systems, meeting demands of the sponsor for performing given functions, and beating deadlines.

In the midst of these pressing preoccupations, it is all too easy to forget that we are designing a system that will interact with a user. From his point of-view, the system will exhibit behavior, just like another person or animal. If its behavior is puzzling, contradictory, and frustrating, the user will have a lot of trouble getting along with it. If it leaves him hanging, not knowing what to do next, and he has to dig through a badly-written manual while his work waits, his time and the system resources are being needlessly wasted, when we could have told him what he needed to know in a simple message on the screen. If system messages mislead the user, or if data-entry procedures are confusing and inconsistent with normal usage, we are designing in a source of constant error. All it takes, in many cases, is a slight re-wording of a message, addition of information to an incomplete message, or standardization on one set of field labels or procedures, to solve these problems for the user. Very rarely will the changes toward user-friendliness require any major sacrifices in efficiency, running time, or ease of debugging or maintenance. In fact, the same changes that make a a system more predictable and convenient for a user are likely to make it easier to maintain and debug as well.

I guess what I am trying to sell to you is the need for a lot more EMPATHY TOWARD THE USER. Interactive programs are basically different from the kinds of programs that read in a batch of data, chew on it a while, then spit out a batch of answers to be read later at a user's desk, or run on another computer. At least half the work in an interactive system is being done by that user out there, who is carrying out a continuous exchange of information and instructions with the system. When we design such a system, we cannot afford to let ourselves forget that we are creating BEHAVIOR. The interface that the user sees will have characteristics that significantly affect his work efficiency. We have to find a way to keep in mind what the user is trying to do, what expectations he brings to the task, and how he will perceive what the system is saying to him. At any given point in an interactive dialog, the user has certain expectations in terms of information he needs and timeliness of response to avoid breaking his train of thought. You, the system designer and programmer, have built up that set of expectations (whether knowingly or not) in the sequence of dialog steps that preceded the screen the user now sees. Whether we like it or not, when we design and implement an interactive system, we are creating behavior, and we are creating a conversation. If we are going to do it right, we must somehow get into the habit of empathy, imagination, putting ourselves in the user's place, at all stages of our work.

WHY IS IT SO HARD TO SEE THE USER'S VIEWPOINT?

Unfortunately, there aren't very many good tools and techniques yet to help designers change their point of view from the old "batch" way of designing programs. Most of us still tend to approach an interactive system design task pretty much as if it were a batch program. A task is specified by a sponsor who has certain requirements; we know what computer system and programming language we will use; we go ahead and write A PROGRAM which will do the job within those constraints. We treat the interactive user as if he were a tape drive, a card reader, or any other input device that we get data and parameters from. Instead of sending a seek to the disk, or a read to the tape drive, we send a message to the user. This is a very poor way to look at an interactive task! People are not like disks or tape drives, for better or for worse. Somehow we must develop diagramming and planning techniques, modelling and prototyping skills, and useful practices and guidelines for this new and special kind of programming and design involving dynamic give-and-take between user and system.

WE NEED A COURSE IN INTERACTIVE SYSTEM DESIGN

I would very much like to see a course in "Design of Interactive Computer Dialogs" taught at our School. There are a number of courses being taught at Universities and Colleges, and in private industry. Videotapes are available from at least one source I know of, Dr. Ben Schneiderman at the University of Maryland. An excellent course is offered by Dr. J. D. Foley at GW. There are also a number of research efforts under way in several places to develop guidelines for interactive system design, and they have published useful papers (e.g., those by Ramsay, et al. and Smith, et al. in the references). I believe that such a course should be practical in its orientation. It should include at least one real design project. And it ought to be required in our Data Systems Professionalization program!

TOOLS TO HELP THE DESIGNER

In the near future, there will be new aids for designers of interactive systems. We will be able to use the power of interactive systems themselves in the design process, with rapid prototyping and planning packages similar to the PSA/PSL system currently in use by T-Group for program design. I believe that we could gain useful techniques and tools right now from the Computer Aided Instruction (CAI) field. Designers of computerized courses have developed a lot of experience in building one type of interactive dialog. A study of CAI packages and techniques, and an attempt to transfer useful ideas to interactive system design, would amply repay our effort. Unfortunately, our need is pressing and we don't have these tools at our fingertips today. There are still some informal methods we can use to help us visualize and manipulate the essential structure of an interactive session from the user's point of view. The diagramming method I am suggesting in this workshop is a simple, pencil-and-paper aid you can use right away to try out ideas and see how they will impact the user, to compare different designs, and to trouble-shoot bad spots in an existing dialog.

EXAMPLES OF GOOD AND BAD USER DIALOGS

As a way of demonstrating the importance of empathy toward the interactive system user, I will present several examples of poor design, chosen from actual NSA systems in current use. The examples will be disguised, to avoid needless embarrassment to designers, managers, and programmers associated with them. Features of several real systems may be lumped together into one artificial "system" for the sake of a dramatic illustration. The essentials of each feature will be retained, with surface details changed to conceal the source. "Rock-throwing sessions" are destructive to all concerned, and my intention is not to criticize any specific system or Agency element. In many cases, the particular unfriendly feature I have chosen to describe is only a small part of a system which is otherwise very helpful to users. With just a little forethought to avoid making needless problems for the user, these excellent systems could be performing far better. I will also present some counter-examples to illustrate good, "user-friendly" designs for contrast. Some of these are chosen from the same real-life Agency systems as the "bad" examples. (For reasons of space, the examples were not included in this paper, but were presented in my talk at the workshop only.) First, let's look at the basic shape of an interactive session - the structure that makes it essentially different from a batch program run.

THE STRUCTURE OF A DIALOG

I will illustrate these points with a convenient method of diagramming an interactive user dialog, which emphasizes the dynamic structure of the interaction. Fig. 1 shows a diagram of a simplified typical dialog. The circles are states of the user, and the arrows are exchanges of information between him and the system which move the user to a next state. Each step from one state to the next involves a user input, followed by a system response. In this analysis, we are interested in the USER's states; the system, too, has states, but we are seeing them entirely through the user's eyes at present, because it is the user's viewpoint we are trying to model and understand in this exercise. The action starts when the user sits down at the terminal and LOGS ON. When the system receives his log-on, it can either accept him, at state 1, and display a message, prompt, or menu, or else it can refuse him and give him (we hope) a clear message telling him what is wrong, at state 4. If his log-on is accepted, the system gives the user access to what I will call the TOP LEVEL of the dialog. Here he has a chance to select one of a set of major actions he can perform on the system. They can be commands he may type in, files he may call up, numbered choices from a menu, or function buttons he may press. When he selects one of these actions, the system will again respond by either giving him access to the subsystem he has requested (file, command, routine, package, etc.) or displaying a message warning him that he is unauthorized to use it, or has made an error in his input.

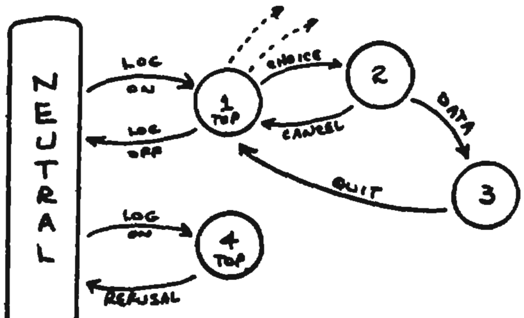


Fig. 1

The function the user has chosen may offer him still another set of choices, leading to another level of subsystems, or there may be a linear chain of actions and responses between user and system, involving no further choices of dialog paths, but continuing until the action is done (data entry, record retrieval, computation and display of a result, etc.), for instance, the step from state 2 to 3 in figure 1. After the user has viewed the display, or the system has completed work behind the scenes and given the user a message, the system may automatically return the user to a higher level and let him choose a new action at that level, or else it may ask him where he wants to go next. Eventually, the user will decide to quit work, or else the system will automatically terminate his session and he will be returned to the top level, where he will LOG OFF. At any point in the dialog, the user may suddenly see that something is wrong, or else he may have to break off his work unexpectedly, so he will need to ABORT or CANCEL the dialog and return to a higher level before work at the current level has terminated normally. In many existing systems, the top level is special, in that a user cannot ordinarily bypass it by an interrupt from inside the dialog. His interrupt will get him back to the top level, where he must LOG OFF to get out of the system entirely. This is because logging on and off are often handled by the operating system, while lower levels of dialog are handled by a specific routine or software package under the operating system. Some systems allow users to log off directly from one or more states within a subsystem without having to return first to the top. At each state, the system gives the user specific displays that must tell him what he needs to know to select the next transition. No matter what else may be going on behind the scenes in the host computer, disk files, mass memories, network connections, data links, etc., ALL THE USER'S DECISIONS MUST BE BASED ON WHAT HE SEES ON THE SCREEN RIGHT IN FRONT OF HIM HERE AND NOW. As the designers and programmers of the user interface, we have to find a way to tell him just what he needs to know right now, to make the best choice of his next action. Why make him guess at incomplete information, try to remember lists of commands, or recall data displayed on earlier screens? You are unnecessarily adding to his burden and detracting from his effectiveness in the primary task he is performing with the system.

The table below is a state-transition matrix. It provides a useful way of summarizing information concerning pairs of states, or data associated with each transition allowed by the dialog in figure 1. This particular table shows user inputs (U) and system responses (S) for each transition. An X indicates that the transition is not allowed by the dialog. Such a table is a convenient method of reviewing all the possibilities and planning or analyzing an interactive dialog.

CURRENT STATE	NEXT STATE NEUTRAL	1	2	3	4
NEUTRAL	X	U:log on S:top menu	X	X	U:log on S:refusal
1	U:log off S:feedback	X	U:choice S:prompt	X	X
2	X	U:cancel S:top menu	X	U:data S:display	X
3	X	U:quit S:top menu	X	X	X
4	U:-- S:message	X	X	X	X

Figure 2 shows a diagram of a dialog with a user-interrupt transition skipping from State 3 directly back to the top level, bypassing an intervening State 2, and skipping State 4, which ordinarily would have come next.

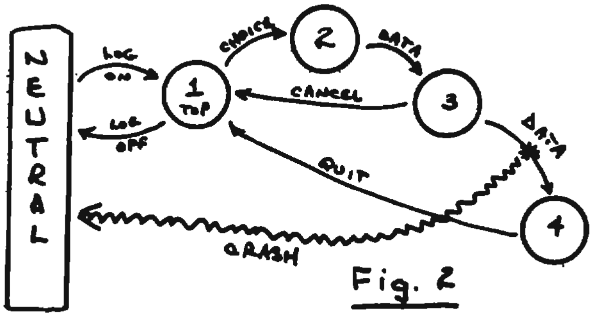


Fig. 2

For instance, suppose that a user has logged on, and at State 1 has asked for a file-update package. State 2, inside the file-update part of the dialog, has given him a data-entry format to fill in on his screen (State 3). While entering data, the user suddenly realizes that he should first have retrieved a record to check its contents before making the updates. He enters a command canceling the data entry screen, removing the effects of any data he may have entered already, and returning him directly to the top level, where he may request the retrieval subsystem and make his query. It is useful for the designer to make a state-transition table including all the "cancel" or "user interrupt" transitions he will allow, and listing the items of data or program variables that must be reset to clean up the loose ends at each point. This is also a good method of deciding where we can reasonably permit the user to cancel, without creating too much chaos in the data base or program variables. In general, it is more "user-friendly" to allow user-interrupts at as many points in the dialog as can reasonably be managed.

Before he logs on, and after he logs off, the user is in a special "neutral state", where he is not directly affected by anything the system does. In this state, he is not engaged in any dialog with the system. Once he logs on, and from then until he logs off, he is more or less closely coupled with the system, and his actions are directly affected by the give-and-take of the dialog. At each of his states within the dialog, he expects to see certain data, within a certain range of time-frames. He has specific uncertainties which must be resolved completely by the display on the screen, if he is to be able to continue working effectively. The system designer must be aware of this mental context built up by the previous steps of the interaction, and provide what the user needs to know right where he needs it. The diagram can help us by factoring out each transition and letting us consider just what path or paths have led the user to a given state. A state-transition table can help us organize and review all the items of information the user has supplied to the system and expects to receive from the system at each state.

Figure 2 shows another situation, which can happen in this all-too-imperfect world. Imagine that the user has worked his way to State 3 again. Suppose, for instance, that he has called up the retrieval subsystem (State 2), entered a query (State 3), and is waiting for the system to display the retrieval on his screen. Suddenly, he finds himself not at the top level, but all the way back at the neutral state, out of contact with the system. The dialog has been stopped dead, the screen is unresponsive, and when he pushes a key, nothing happens. The system has CRASHED in midstream, perhaps leaving a multitude of messy loose-ends hanging behind the scenes.

The way this experience feels to the interactive system user must be lived through to be truly understood. The best way to describe it is to say that one minute the user is closely involved in a lively give-and-take with a responsive, talkative entity, and suddenly everything has died on him. It is a very peculiar and frustrating feeling, a little like running into a wall. Whenever this happens to a user, he will have some very strong uncertainties that must be resolved somehow by messages or pre-arranged procedures, so that he can put his work back together and get going again with a minimum of lost motion after the system comes back up. In addition, THERE IS A STRONG EMOTIONAL RESPONSE of frustration and alarm, especially if the user knows that a lot of his work will be lost: not just the last action he performed, but perhaps hours or even days of earlier work as well. A state-transition table including "crash" transitions back to the neutral state can help us review the possibilities and plan what loose ends need to be cleaned up and what recovery features need to be provided to the user.

The diagrams can be a handy aid for "doodling" while you are planning an interaction. If you are a designer, they can help you to see the structure of a user interface from the user's viewpoint. They can help to make clear what are the successive choices a given design puts before the user, and what data he needs at each point to decide where to go next. If you are a user, you might find it an interesting exercise to try diagramming all or a portion of a user dialog for a system you use, especially a part of it that often gives you trouble. Experiment with various kinds of matrices and tables of data associated with state-pairs, for instance user performance times, system response times, user inputs and system messages, etc. The examples in these handouts will provide some illustrations. The references at the end of this paper list several sources where state diagrams for interactive dialogs are discussed. These references were brought to my attention by Joan McDonald, R8.

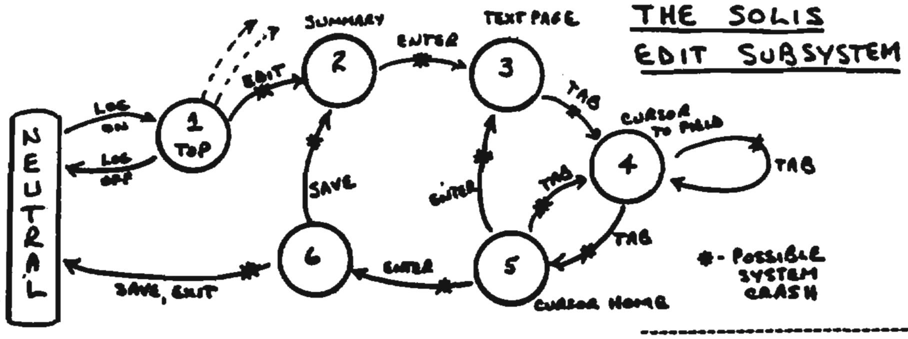
SOME HANDY RULES OF THUMB

Below are some guidelines that express the "moral" of the illustrations I presented in the workshop.

1. Don't make the user give you redundant information; get it from him once and use it efficiently behind the scenes. (E.g., if you ask him for his name and can look it up, you shouldn't need to ask for his initials and his social security number too. If you have a password, you shouldn't need anything else to identify him.)

2. Don't leave the user looking at a blank screen after he has input a command or data. Give him a message ("Document number not found", "retrieval ended", "retrieval failed") or at least a prompt to tell him what level of dialog or state he is in so he has an idea what to do next, especially if the command he just input has failed.

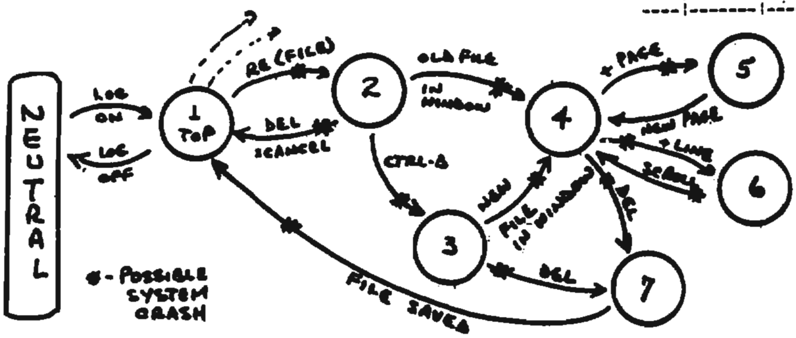
3. Don't build data entry formats or representations into your dialog which are counter to normal usage. You are laying the foundation for persistent user errors that will waste far more time in error checking and recovery code than the extra trouble to



THE SOLIS EDIT SUBSYSTEM

Fig. 3

CUR-	NEU-			NEXT			
RENT	TRAL	1	2	3	4	5	6
NEU-TRAL	X	U:1.on S:top T:2min	X	X	X	X	X
1	U:1.off S:ackn. T:6-10s	X	U:edit S:dsply T:6-10s	X	X	X	X
2	U:-- S:crash	X	X	U:enter S:text T:6-10s	X	X	X
3	U:-- S:crash	X	X	X	U:tab S:field T:0	X	X
4	U:-- S:crash	X	X	X	U:tab S:field T:0	U:tab S:home T:0	X
5	U:-- S:crash	X	X	U:enter S:text T:6-10s	U:tab S:field T:0	X	U:enter S:prompt T:6-10s
6	U:exit S:top T:6-10s	X	U:save S:dsply T:6-10s	X	X	X	X



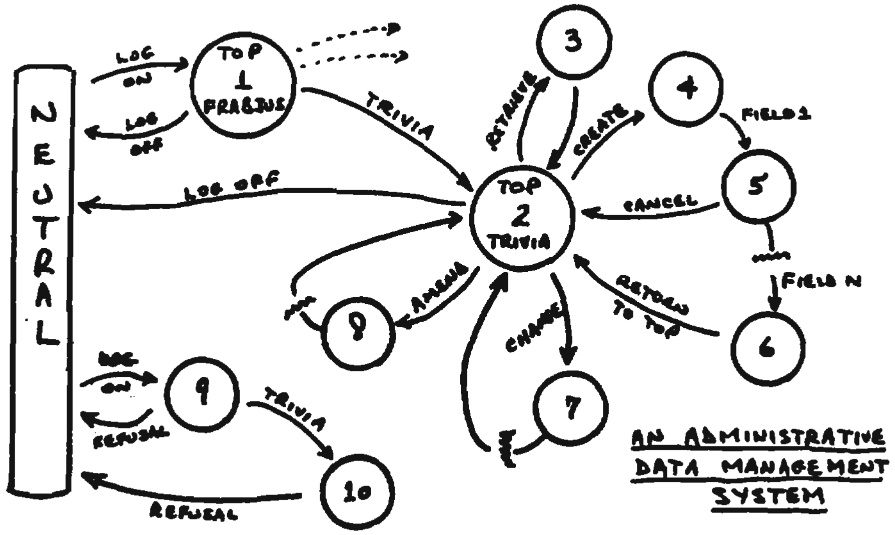
RAND EDITOR

Fig. 4

CUR-	NEU-			NEXT				
RENT	TRAL	1	2	3	4	5	5	7
NEU-TRAL	X	U:1.on S:top%	X	X	X	X	X	X
1	U:1.off S:messg	X	U:re(f) S:wndow	X	X	X	X	X
2	U:-- S:crash	U:del S:messg	X	U:ctl-B S:new f	U:old f S:dsply	X	X	X
3	U:-- S:crash	X	X	X	U:new f S:dsply	X	X	U:del S:save f
4	U:-- S:crash	X	X	X	X	U:+page S:new p	U:+line S:scroll	U:del S:save f
5	U:-- S:crash	X	X	X	U:data S:dsply	X	X	X
6	U:-- S:crash	X	X	X	U:data S:dsply	X	X	X
7	U:-- S:crash	U:-- S:top%	X	X	X	X	X	X

```
CRASH: states 4,5,6, keystroke recovery file
       states 1,2,3, user recovery not needed
       state 7: after hit "del", recovery not possible
```

Examples from Other Systems



AN ADMINISTRATIVE DATA MANAGEMENT SYSTEM

CUR-	NEU-			NEXT							
RENT	TRAL	1	2	3	4	5	6	7	8	9	10
NEU- TRAL	X	U:l.on S:top	X	X	X	X	X	X	X	U:log on S:refuse	X
1	U:l.off S:messg	X	U:trivia S:prmpt	X	X	X	X	X	X	X	X
2	U:-- S:crash	X	X	U:retrv S:dsply	U:create S:prmpt	X	X	U:change S:prompt	U:amend S:prompt	X	X
3	U:-- S:crash	X	U:exit S:prmpt	X	X	X	X	X	X	X	X
4	U:-- S:crash	X	X	X	X	U:data S:prmpt	X	X	X	X	X
5	U:-- S:crash	X	U:cancl S:prmpt	X	X	X	U:data S:prompt	X	X	X	X
6	U:-- S:crash	X	U:exit S:prmpt	X	X	X	X	X	X	X	X
7	U:-- S:crash	X	U:exit S:prmpt	X	X	X	X	.X	X	X	X
8	U:-- S:crash	X	X	X	X	X	X	X	X	X	X
9	U:-- S:messg	X	X	X	X	X	X	X	X	X	U:trivia S:refuse
10	U:-- S:messg	X	X	X	X	X	X	X	X	X	X

provide a natural format in the first place! If a field is to contain dollars and cents, permit the user to insert a decimal point, and edit it out later if you must. If the data structure behind the scenes requires an odd-ball representation of dates, times, etc., let the user enter them in a natural, easy-to-remember form and do the needed conversion in the software. When you display them to him, convert them behind the scenes back to the form he is used to.

4. Don't let different parts of the same system use different formats, procedures, or representations for things that are the same to the user. Use the same method of error warning and correction throughout all the data entry routines of a system. Use the same convention for "default" or "null" data entries and command selections. If one subsystem provides a list of options "1", "2", "3" and asks the user to pick one, don't use a similar display with numbered items to mean something else in another subsystem. If some subsystems tell the user "PROCESSING COMPLETE" at the end of their actions, as a way of letting him know he is back at the top level, all subsystems should do so, rather than some of them leaving the user looking at a blank screen. If you display a review after data entry in a form, always give the user the same way of indicating "correct" or "wrong" and making corrections, throughout all data-entry routines of a single system.

5. When you send an error or warning message to the user, tell him clearly what is wrong, where the error is, and what he can or must do next. Don't just say "invalid code"; give him a list of what the valid codes are, or provide a "help" or "?" command that will display them to him without interrupting the interaction. Don't just say "index out of range"! Tell him the name of the variable, and what was in it ("I=0", "X=9999999"). If the message indicates that something is wrong which the user can't fix, give him a phone number to call for help, and keep the message up to date so the number is right.

6. If certain data to be input by the user must function as key elements in the data structure or the procedures on which the task is based, let him input them at the beginning, check them thoroughly right away, and give him a chance to correct them if needed. Don't wait until he has entered several pages of a form before you tell him "WRONG SSN - FATAL ERROR" and make him re-enter all the other data.

7. If there are requirements for contents or formats in data entry fields, check for them right away and let the user correct errors for each field as he enters it. Don't give him a review of several fields at once, one of which may be incorrect, unless you also provide a forms-entry interface with tabbing from field to field and protected field boundaries.

8. Don't use different labels or abbreviations for field names, commands, or other key words in different parts of the same dialog or system. If a field is called "AMOUNT" in one display, call that field the same thing in every display or message that refers to it (not AMT one place, MONEY someplace else, "FUNDS" someplace else). Don't choose labels or abbreviations that look alike or are confusing. While an experienced user may be used to some of these, you are making it needlessly hard for a new user to learn and remember them. Experienced users have a way of leaving, and all users have to start out as new users sometime.

9. If there is information at the top of the screen the user needs to see, be sure it doesn't get scrolled off before he gets to use it. If you aren't sure it will still be there when he needs it, display it again; don't count on its being visible now just because you displayed it a few steps earlier.

REFERENCES

Foley, J.D., Notes and Lectures in a course taught at G.W. University, Fall 1980

Foley, J.D, and Wallace, V.L., "The Art of Natural Graphic Man-Machine Conversation", Proceedings of IEEE, Vol. 62, No. 4, April 1974, pp. 462-471.

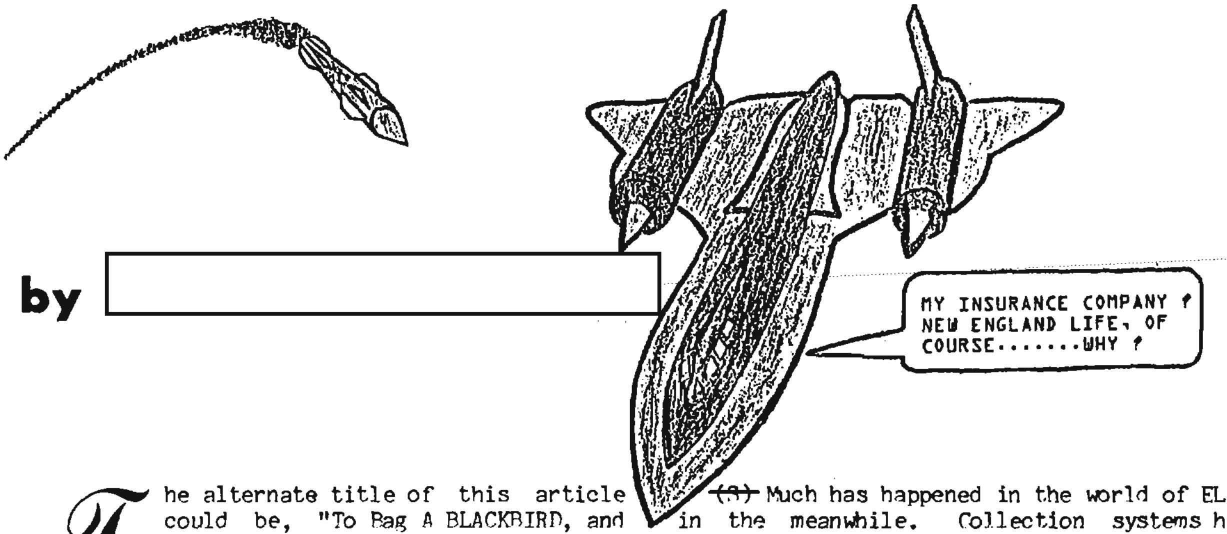
Giloi, W.K., Interactive Computer Graphics, Prentice-Hall Inc., 1978, pp. 205-207.

Newman, W.M., "A System for Interactive Graphical Programming", AFIPS Proceedings of Spring Joint Computer Conference, 1968, pp. 47-54

Ramsay, H.R., and Atwood, M.E., "Human Factors in Computer Systems: A Review of the Literature", Science Applications Inc. Technical Report SAI-79-111-DEN, 21 September 1979.

Smith, S. L., "Requirements Definition and Design Guidelines for the Man-Machine Interface in C3 System Acquisition", MITRE Technical Report M80-10, 15 April 1980.

OPELINT is Alive and Well in B Group


MY INSURANCE COMPANY ? NEW ENGLAND LIFE, OF COURSE.......WHY ?

by [redacted]

P.L. 86-36

The alternate title of this article could be, "To Bag A BLACKBIRD, and Other SIGINT Tales," and it might start with a conversation something like this: ~~(C)~~

"Hello, Ralph? You know that SAM site that isn't there?"
"Yes, Jim, what about it?"
"It just launched two SAM's at the SR-71."
"Oh, no!"

[redacted]

~~(S)~~ Five years ago, [redacted] excellent article, "Yes, Don, There Is An ELINT!" (CRYPTOLOG, August 1976), brought ELINT out into the bright light of day for many. His TECHELINT oriented piece invited a companion article from an OPELINT-er in A, B, or G Group. No one has yet responded to the challenge, so this article will hopefully begin to fill that void.

EO 1.4.(c)
P.L. 86-36

P.L. 86-36

~~(S)~~ Much has happened in the world of ELINT in the meanwhile. Collection systems have proliferated, measurement capabilities have improved, and in many areas of DDO, OPELINT and COMINT analysts have joined forces against their targets. This inter-disciplinary mingling has spawned a new breed of SIGINT analysts who are equally comfortable on either side of the fence, and who constantly strive to operate on both sides, to produce the highest quality, most accurate SIGINT product available.

~~(S)~~ Much of the credit for this welcome evolutionary stage in the state of the art known as SIGINT production goes to those Agency and Community managers with foresight enough to appreciate the potential of ELINT, who have pushed to popularize and expand the ELINT fusion curriculum in the National Cryptologic School, and who have kept up the successful battle to remove the best ELINT collection from the compartment in which it resided for so long.

~~(C)~~ Managers who have made the choice to make ELINT work in their own SIGINT organizations have found that activism and encouragement are the keys to success. Without these ingredients, the curtain separating the two major components of SIGINT remains impenetrable, while analysts continue to "go with what they know," rather than trying something new and perhaps strange.

[redacted]

~~(C)~~ The crux of the dilemma which followed consolidation was how to keep ELINT close to the entity analysts while still achieving the economies of scale offered by consolidation, for OPELINT, worked away from the COMINT it supports, perpetuates the separation of the two.

[redacted]

[redacted]

~~(C)~~ As a result, we have a SIGINT analytic fusion effort which really has gotten the maximum mileage out of both COMINT and ELINT in producing a number of significant products and studies with far-reaching implications.

[redacted]

[redacted]

To Pag a "BLACKBIRD" (U)

[redacted]

EO 1.4.(c)
P.L. 86-36

[redacted]

EO 1.4.(c)
P.L. 86-36

EO 1.4.(c)
P.L. 86-36

[redacted]

Which Way Did They Go? (U)

~~(S)~~ The reconstruction of a recent SIGINT event scenario serves to indicate how valuable a resource ELINT can be in determining exactly what occurred.

[redacted]

[redacted]

Try It, You'll Like It! (U)

~~(C)~~ In sum, we are still on the uphill side of the learning curve with regard to the best uses of OPELINT, both on its own and in fusion. But the more we learn, the more we find we can do. And , oh yes, who performs these "exotic" analytic routines? Ordinary Traffic and Special Research Analysts, just like you and me, with a more than able assist from a one INSCOM-trained 98J ELINT Analyst.

~~(C)~~ We are all capable of using ELINT to its full potential, given minimal training and practice. Let's bring ELINT out of the closet and into the spotlight it deserves!

SOLUTION TO NSA-CROSTIC No. 35

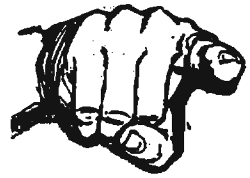
"Plain English," by [redacted] [redacted] CRYPTOLOG, May 1977, reprinted from the final issue of C-Liners.

"If we want all Agency personnel to speak and write plain English, perhaps we should first teach Agency personnel English. If we want Agency management to write concise, active, decisive memos, perhaps we should first teach Agency management to be concise, active, and decisive.
"Let us attack the problem, not just hide the symptom."

P.L. 86-36

EO 1.4.(c)
P.L. 86-36

What do YOU think?



Review:
"Wouldn't It be Nice if We Could Write Computer Programs in Ordinary English -- or Would It?"
I. D. Hill, The Computer Bulletin, June 1972, pp.306-312
by [redacted] P13

P.L. 86-36

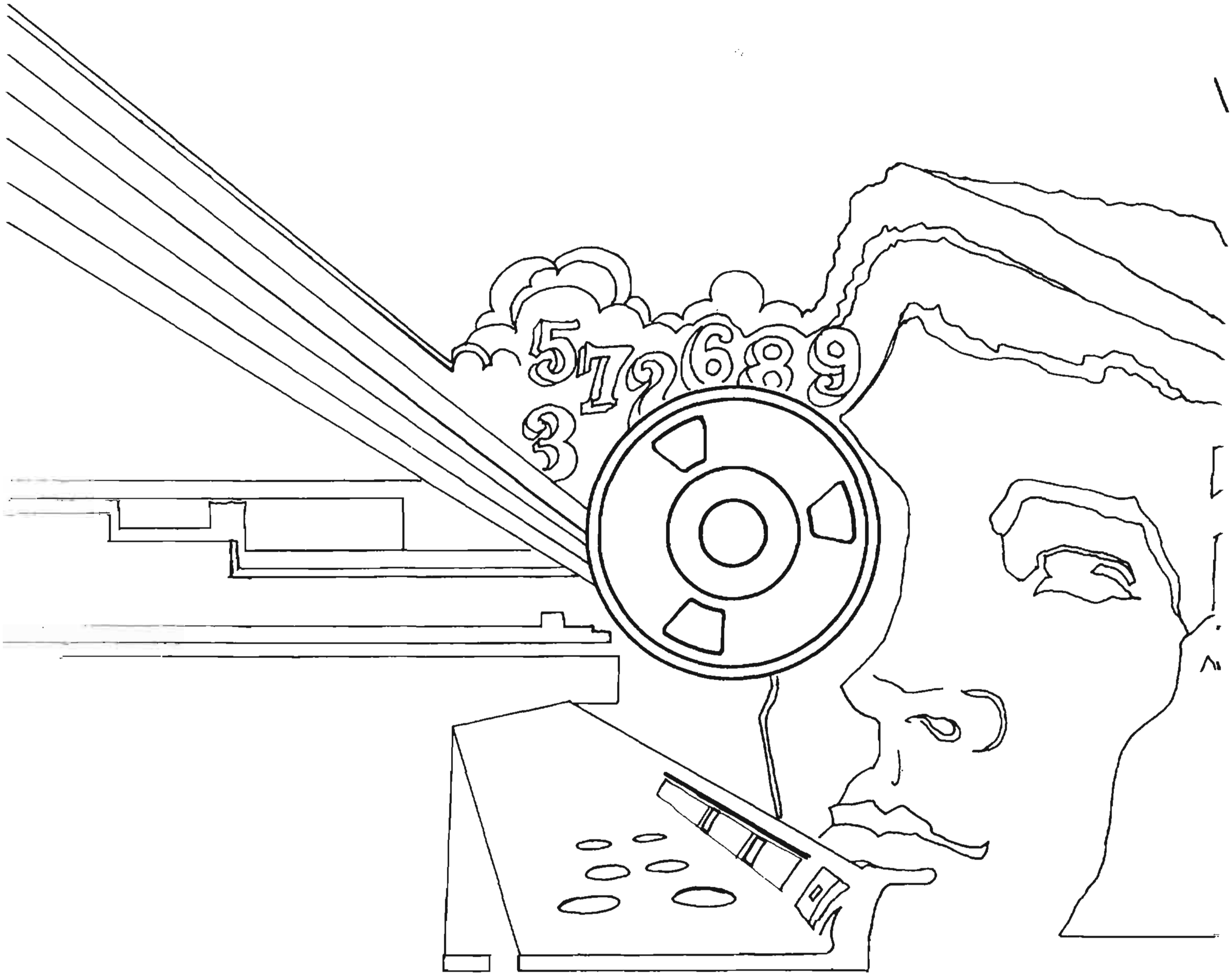
Although this paper was published nearly a decade ago, it is still right on target in an area of computer technology that is receiving even more concentrated attention today. There are many software designers who seem to think that "natural language" (by which they usually mean "everyday, conversational English") is the ideal medium for communication between humans and computers. The arguments for this idea are obvious: we all know English, and nobody has to spend a lot of time and money teaching it to us. Of course, teaching it to computers WILL require tremendous amounts of time and money, if it is ever really possible at all. The "natural language" enthusiasts are convinced that the vast benefits conferred by computer English will counterbalance the costs. Proponents of "natural language" also seem to assume that English is an ideal medium for humans to specify exact and detailed instructions to computers and other humans. Hill has presented some very good arguments against this assumption, and I believe all of us who are concerned with human-machine systems can profit from a careful consideration of the points he raises. If everyday English is ineffectual as a means of conveying exact instructions, it certainly isn't worth spending all that money and time to design English-like programming languages.

Hill's paper presents a very clever and amusing argument against the use of "natural" English for programming. In fact, in a half-serious way, it suggests that English isn't very good for any precise description or instructions intended to guide others' actions; instead, people should learn to use an algorithmic formal language, even in specifying procedures for other people, as well as in working with computers. Hill makes his points about the deficiencies of English very convincingly, in spite of the exaggerations implicit in the humorous view he presents. He makes some other very interesting points - for example, that the unreadableness and general difficulty of legal language, (which attempts to specify something precisely and unambiguously), result from the unsuitability of natural English for this use. He gives some amusing examples of people misunderstanding other peoples' descriptions, and some more serious ones. He uses an ALGOL-like block notation to show the precise meaning (or alternative possible meanings) of many examples quite effectively. Cooking recipes, knitting instructions, and musical notation provide further examples of somewhat better specialized subsets of English; these are incomplete, however, and still open to misunderstandings.

Hill makes the further claim that, even if a way could be found to enable computers truly to understand natural language inputs, there remains a much more basic objection. English just isn't the right language to THINK in while deciding "EXACTLY what is the right thing to do". We would sacrifice precision and power in the way we think about a problem or task, and also in the way we represent the task and possible solutions. We would also lose a main advantage of computers: their reliable obedience in doing exactly what we unambiguously instruct them to do, without a chance of misunderstanding. It wouldn't really be a help to have computer systems that responded, like people, "Oh! but I thought you meant so-and-so!" (especially if you, the human user never realized that the system was acting on a totally different, BUT SENSIBLE interpretation of your instructions until it was too late for correction!) Hill suggests that we start by using an algorithmic form for specifying legal/financial matters (e.g. taxes), many of which have to be programmed into computers, anyway.

While this idea of using a "programming" language in communicating with other people seemed bizarre to me at first glance, second thought made it more and more convincing. English phrases, with BEGIN, END, meaningful labels, judicious use of "go-to's", and careful use of parentheses to define scope, with some other conventions defining logical implication, conjunction, and disjunction might prove a highly useful tool. People could switch to an algorithmic description in this language whenever they felt they had not been understood, or use it whenever they anticipated difficulty. It could be taught in elementary schools, especially when terminals and interactive teaching networks become commonplace. I can imagine a highly amusing parlor game involving "charades" or skits, called "What's my program?" In this game, one team would compose a set of tricky "programs" for sets of actions to be performed by the other team; penalty points would be scored for failure to follow the specifications exactly.

What do you think?



~~THIS DOCUMENT CONTAINS CODEWORD MATERIAL~~