software age

MAY 1968

SOFTWARE MANAGEMENT THROUGH SPECIFICATION CONTROL. SEE PAGE 8

B8F3C 38117&OREC5341 T
CLYDE OREM
5341 TIMMONS AV
MEMPHIS TN 38117

THE MAGAZINE FOR PROGRAMMERS • MATHEMATICIANS • ANALYSTS • EDP MANAGERS

... or Improve it. That's what you'll be expected to do at Honeywell's EDP Technology Center. We're responsible for planning, designing and developing computer systems. To perform these responsibilities we need highly motivated programmers and analysts with a desire to prove and improve their state-of-the-art. To prove the importance which we attach to this task, you will find over 20 computers in our Technology Center — computers whose only function is to be at your disposal. Are you ready to improve your career potential?

PROGRAMMER/ANALYSTS

To be engaged in researching, designing and developing software packages which will maximize the efficiency of present and future generations of hardware. To demonstrate the imagination and creativity necessary to the original conception and development of complex software systems. Openings exist at the intermediate (2 years experience) through senior levels. We are specifically interested in:

PRIMARY SYSTEMS DEVELOPMENT:
The design of complete operating systems to act as interface between customer applications and computer systems in a real time environment.

COMMUNICATIONS SOFTWARE:
To monitor, control and interface communications networks and computer systems in a real time environment.

MASS STORAGE:
The design of software to control and utilize random access storage devices.

COMPILER TECHNIQUES:
Translation and simulation of existing programs and systems so that they may be more effectively used with new hardware and software.

SCIENTIFIC SOFTWARE PACKAGES:
Development of software for use in operations research, linear programming, numerical control, math modeling and simulation.

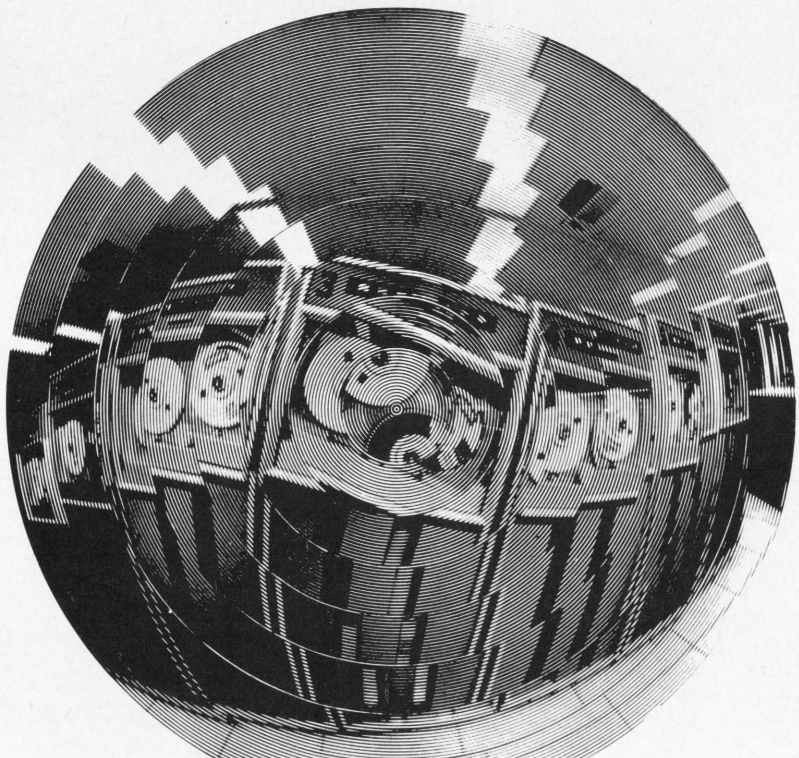
Please forward your resume to Mr. David McGowan.

HONEYWELL
200 Smith Street, Dept. SA-5 , Waltham, Massachusetts 02154

Opportunities exist in other Honeywell Divisions Send resumes to F. E. Laing, Honeywell, Minneapolis, Minnesota 55408. An Equal Opportunity Employer.

The Other Computer Company:
Honeywell

PROVE IT



software age

MAY, 1968 Vol. 2–No. 4

CONTENTS PAGE

Publisher ---- H. L. Rothra
Associate Publisher ---- David W. French
Advertising Manager ---- Norman Jacobs
Advertising Assistant ---- Betsy Pavkovich
Production Manager ---- William J. Murphy
Director of Circulation ---- Howard Rogers
Circulation Manager ---- Judith Arnopolin

Circulation of this issue more than 91,000

Software Age is published monthly
by Press–Tech, Inc.
(312) 869–1244

1020 Church Street, Evanston, Illinois 60201. Subscrip- free to qualified readers. Others, $10/yr. Individual copies, $1. Foreign subscriptions, $10/yr.

Main Sales Office: Norman Jacobs (Adv. Mgr.) or Norman Brodsky, 1020 Church Street, ***Evanston, Illinois*** 60201—Telephone (312) 869–1244. Gerald Green, 60 E. 42nd, ***New York, N. Y.***—Telephone (212) 697–5356. Richard Faust, 9800 S. Sepulveda Blvd., ***Los Angeles, California*** 90045—Telephone (213) 776–0100. David S. Wood, P. O. Box 189, ***Canton, Massachusetts*** 02021—Telephone (617) 542–1466. Richard D. Clemmer, 4900 Red Fox Drive, Annandale, Virginia 22003 ***(Washington, D. C.)***—Telephone (202) 461–9792. Richard Faust, ***Palo Alto, California***—Telephone (415) 327–8340.

CONTROLLED CIRCULATION POSTAGE PAID
AT MADISON, WISCONSIN

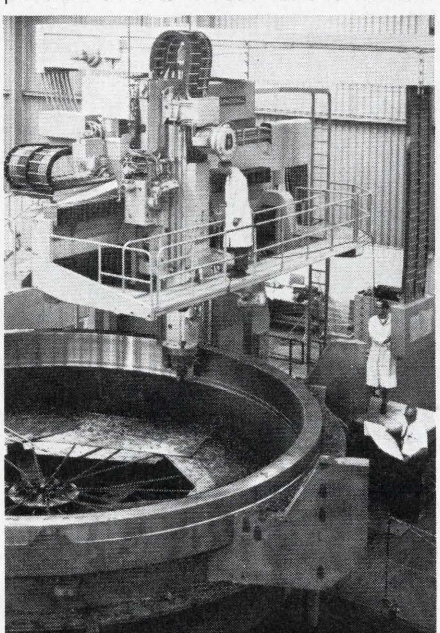
Manufacturing Systems and Process Engineers:

Work with advanced techniques at new Boeing fabrication facility.

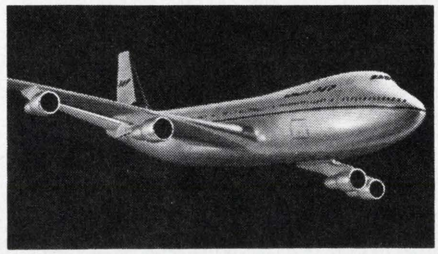
The Boeing Company's newly activated Central Fabrication facility near Seattle, a modern complex of specialized fabrication and processing shops and equipment, offers exceptional ground-floor career opportunities to manufacturing systems and process engineers.

The company is extensively exploring highly advanced, integrated manufacturing operations concepts involving (1) computerized control of manufacturing processes and (2) integrated planning and control systems.

Boeing has initially invested $120 million in this new facility. A large portion of this investment is in new machine tools, many with numerical control capability. The entire complex comprises 1.6 million square feet of covered area and includes four major factory buildings: an airplane wing spar and skin mill facility, a heavy machining facility, a tool fabrication facility, and a manufacturing process facility for metal bonding, plastic and fiberglass duct work.



World's largest numerically controlled vertical boring machine is one of many modern pieces of equipment at Boeing's Central Fabrication facility.



Boeing 747 superjet is typical of advanced jetliner programs supported by new fabrication facility near Seattle. The 747 is scheduled for rollout this year.

The Central Fabrication facility supports all Boeing programs in the Seattle area. These include current, advanced jetliner programs such as the 747 superjet, the short-range 737 twinjet, and the U.S. supersonic transport, as well as a number of in-production and developmental space and missile programs.

Immediate openings exist for engineers at many levels of experience and responsibility in the following functional areas:

MANUFACTURING SYSTEMS ENGINEERING

Production Automation Systems — Process control and operations, numerical control machine operations, computer-aided tool design, monitor and display systems.

Production Logistics Systems — Planning and scheduling, tool and production control.

Management Information Systems — Resource planning and control, simulation and model development, management reports and communications, cost schedule and technical performance.

MANUFACTURING PROCESS ENGINEERING

Flow Processes — Metal bonding, plastics and fiberglass, material handling.

Machining Processes — Mechanical and electronic controls, metal removal, forming and finishing, production automation.

Control Processes — Communications, computer applications, display and recording equipment, controls, models and simulators.

Qualifications: (1) three years related experience, (2) B.S. in engineering, mathematics, or the physical sciences, (3) Master's degree in business administration, or M.S. in engineering or mathematics.

You'll enjoy living in the Seattle-Puget Sound region where you can take advantage of year-round Western outdoor activity in the heart of the Pacific Northwest. From sea level to ski level is less than an hour's drive. The 172 San Juan Islands of Puget Sound are only two hours away. Enjoy sailing, fishing, hunting, golfing, clam-digging, beach-combing or mountain climbing at your pleasure.

Fill in and mail the coupon today. A Boeing representative will get in touch with you.

BOEING
COMMERCIAL AIRPLANE DIVISION

Please contact me with details about a career at Boeing's new Central Fabrication facility.

Name________________

Address________________

Phone (incl. area code)____________

City________________

State________ Zip________

I'm usually home between ______ and

______ (hrs.) on ________ (days)

Degree/Specialty____________

Mail to: Mr. William Denman, The Boeing Company, Commercial Airplane Division, P.O. Box 707-BBB, Renton, Washington 98055.

new installations

Systems Engineering Laboratories, Inc., Fort Lauderdale has ordered magnetic core memory stacks, valued at more than $750,000, from Ampex Corp. Ranging in size from 100,000 to 200,000 bits, they will be used in various data processing systems.

■ ■ ■

Project Genie, University of California, Berkeley, has ordered from Scientific Control Corporation a 6700 Central Processing Unit. Project Genie, under the sponsorship of the Advanced Research Projects Agency, Department of Defense, does much work in the field of time-sharing computer systems. The new system at Berkeley will consist of the SCC 6700 Central Processing Unit and memory produced by other suppliers.

■ ■ ■

American Optical, major supplier of optical systems for U. S. space and defense programs, installed the new IBM 1130 computer to assist engineers in designing complex lenses and other optical systems that can photograph the earth's surface from speeding aircraft or probe millions of miles into space. Programmed into the computer is the desired operating characteristics of a proposed new system and received back is the most efficient configuration.

■ ■ ■

St. Regis Paper Co. has installed a high-speed data transmission system by Digitronics Corp. The system links the company's EDP Center in New York City with remote sales and manufacturing locations.

■ ■ ■

Allied Van Lines, Inc., long distance movers, has ordered Communitype Corp. data communication equipment for proposed coast-to-coast management information network. The system is said to receive digital data at a speed of 1,000 words per minute. Compatible with most standard computers, it incorporates an IBM Selectric typewriter used for input and print-out operations.

■ ■ ■

American Collectors Association, Inc., an international trade association comprised of more than 25,000 independent agencies, and *Tri-Data, Inc.*, a computer consulting firm and service bureau, have installed GE-405 computer systems.

The GE-405 is the smallest member of General Electric's medium-scale GE-400 series of information systems. It includes a central processor with 8,000 words of memory, five magnetic tape units, a card reader, and a printer.

■ ■ ■

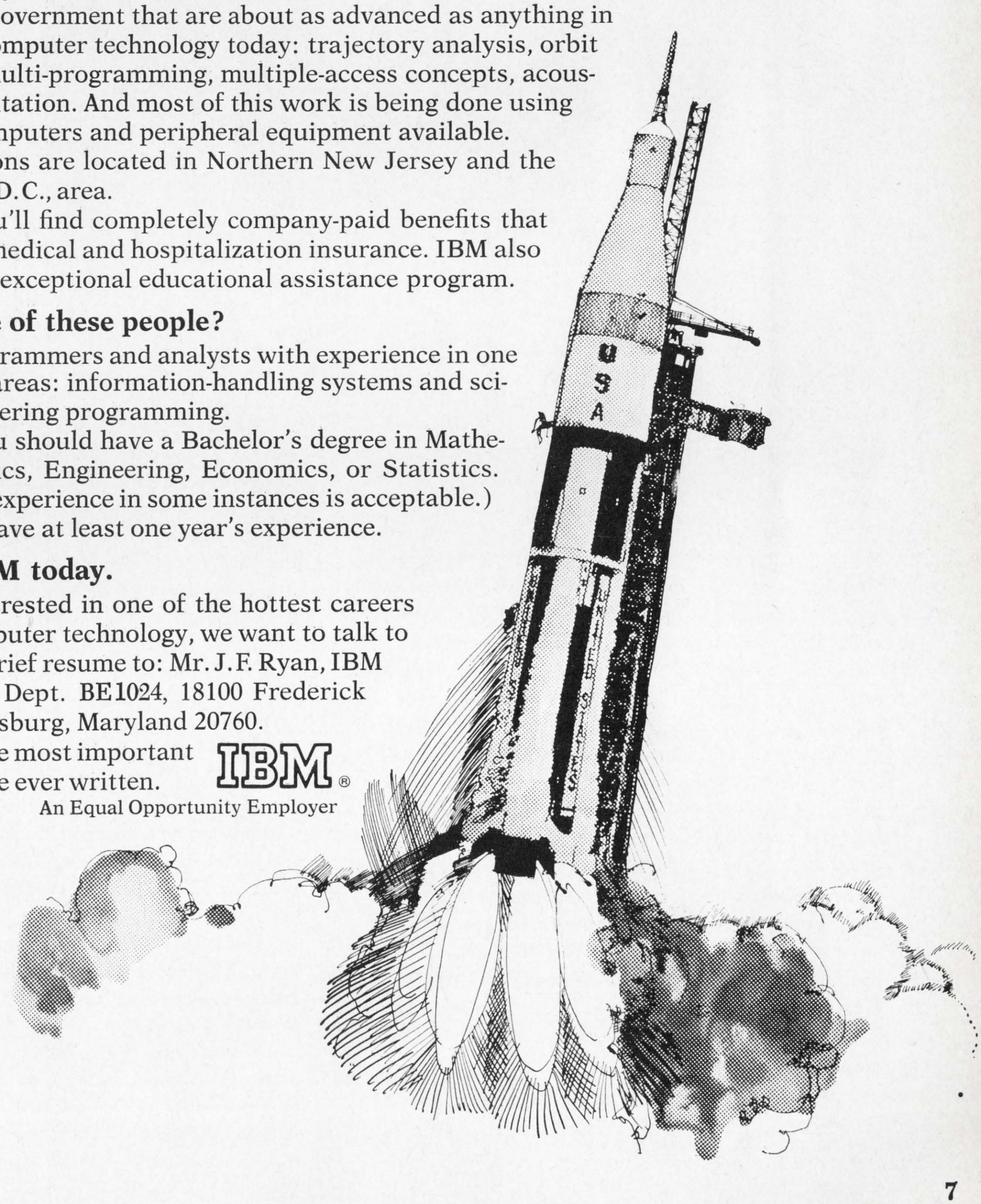
Dial-Data, Inc., Newton, Mass., has ordered a scientific data systems time sharing computer valued at $1.1 million. The computer will be installed in the Washington, D. C. area to serve the entire metropolitan district and provide peak load and backup capability for the SDS 940 computer which has been in operation since 1967 in Massachusetts. Both the SDS 940 centers offer conversational interactive computer access to as many as 32 simultaneous users located at remote terminals installed in their own offices. The users connect the computer via regular telephone lines using teletype keyboard terminals.

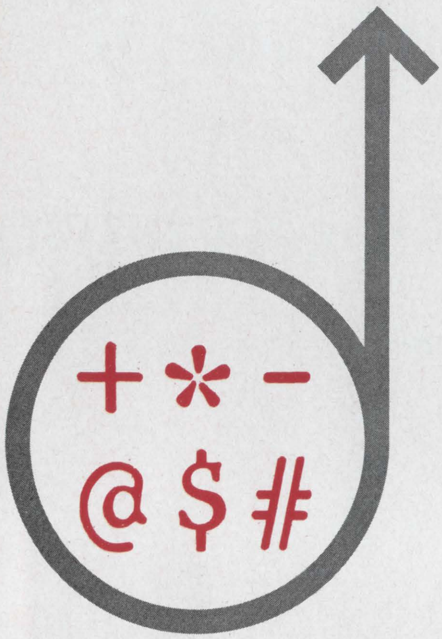
■ ■ ■

The U. S. Bureau of the Census, which installed the world's first data processing computer, has increased its computing capabilities with a new UNIVAC 1108 System. The 1108 is being used by the Bureau for statistical processing.

■ ■ ■

Davis Brothers, Inc., the nation's fourth-largest wholesale drug distributor, has ordered a three-station computer network from Honeywell EDP. The company will keep inventory records on 30,000 items for customers using a Honeywell-developed software system called PROFIT. ■


USA



SOFTWARE MANAGEMENT THROUGH SPECIFICATION CONTROL

By F. Liguori
Sen. Proj. Member, Technical Staff
RCA, Burlington, Massachusetts

About the Author

Mr. Liguori received his BSEE degree from Tufts University in 1957 and an MBA from Hofstra College in 1960. He received his formal computer education at the graduate school of Adelphi College. Since joining RCA in 1962, he has been involved in test designing and programming for various general-purpose automatic test systems, including the Multipurpose Test Equipment (MTE) and the Land Combat Support System (LCSS) for the U. S. Army Missile Command and the Depot-Installed Maintenance Automatic Test Equipment (DIMATE) for the U. S. Army Electronics Command, and a newly installed commercial automatic test system. For the past three years he has been a principle contributor to Independent Research and Development studies on advanced programming techniques for the RCA Automatic Test Equipment product line.

Contrasting software with hardware is a convenient way of describing software to non-programmers. This emphasis on differences has led to a common misconception in industry that the two products are inherently different and thus, their development must be handled differently. Close analysis of both tasks reveals often overlooked similarities. In fact, the two products are so similar in content that the commonly used formal techniques for managing hardware developments apply equally well to software development.

This article offers the rationale behind the thesis that hardware and software development tasks are in fact quite similar. It then discusses the applicability of specification control to software development, emphasizing how each advantage of specification control applies equally well or more so to software development tasks.

The conclusion is that software management is not inherently different from hardware management. The same tools of control may be (and in fact have been) successfully used for software development tasks. It is, therefore appropriate, and necessary for project managers to provide more definitive direction of software development tasks through detailed specifications without requiring personal proficiency in software design.

■ Examination of documents commonly used in industry to control system design and engineering reveals a lack of understanding of the nature of software and methods of software development control. This is particularly evident in Requests for Quotations (RFQ's) involving computerized systems requiring both hardware and software development. Typically, such RFQ's consist of 50 pages or more of very detailed technical requirements. They include not only the necessary performance characteristics, but often even dictate the basic method of implementation of the system to be designed. Of these 50 or more pages, often as few as two or three pages and certainly less than ten percent of the document is devoted to software specification. Yet, software for the system may cost as much or more than the hardware. This over emphasis on hardware is indicative of an unique pre-occupation with hardware characteristics and a lack of understanding of software techniques and requirements. Software characteristics, which by and large dictate operational characteristics of the completed system, are left vague and undefined.

Work statements and design specifications, which are outgrowths of the RFQ, further reflect the vagueness with which software is controlled. The result is that, unless the contractor selected happens to be unusually competent in software development, this aspect of the system is left to be worked out during the hardware development phase when it is impractical to make the optimum hardware/software configuration tradeoffs.

The Automatic Test Equipment product line at RCA has traditionally involved a complex interplay of hardware and software. Experience has proven the value of detailed software work statements and design specifications. This paper develops the rationale behind the similarity between hardware and software development and management tasks and discusses the advantages realized by close control of software tasks through definitive specifications.

Similarity of Software and Hardware Systems

Hardware and software are often contrasted to explain the special problems of software with respect to the familiar base of hardware development. And, indeed there are many aspects of software development that make it different from hardware. Hardware is developed in discrete stages (paper design, breadboard, engineering prototype, etc.) that progressively constrain system changes. Software stage boundaries do not seem to progressively constrain software system changes. The obvious reason is that software appears inherently more flexible (therefore changeable) because it is structured with "paper" rather than fabricated hardware. Ironically, it is this "advantage" of software that is really its nemesis. Nothing can hamper final designs more than susceptibility to change, and nothing is more fatal to schedules than design changes.

A second characteristic of software that makes it more elusive to schedules is its camouflaged nature. Unlike hardware developments which go through visible stages of development, preliminary software and debugged software are of the same general appearance. Even when it is apparent that a task is still in the process of completion, it is much more difficult to determine percentage of completion of software as compared to hardware developments in progress.

Despite the obvious differences in hardware and software it is the thesis of this paper that software and hardware systems have greater kinship than disparity. Furthermore, mismanagement of software systems development stems from overemphasis of differences and failure to recognize and capitalize on similarity. Once this thesis is accepted, it follows that the methods of hardware systems development can be successfully applied to software systems development. This causes two seemingly divergent disciplines to converge in most system applications. Hence, it allows well developed techniques to be applied in a field much less understood by management people[1] who control the integrated system development at the critical point—where software and hardware have a common manager.

The Basis of Similarity

To see the commonality between software systems and hardware systems one must examine not the form of the finished product but rather the resources required to develop each system. In either development, basic resources are time, manpower, and materials, all of which have a common denominator in dollars. These resources are common to all human endeavor so it is necessary to further investigate the types and quantities involved as well as their interrelationships to prove or disprove substantial correspondence between hardware and software development tasks.

When dealing with a system under development, whether it be hardware, software or a combination, schedules typically involve a year or more. Usually, there is no substantial difference in period of performance for hardware and software system development when comparing tasks of approximately equal dollar volume.

In any system development, manpower (particularly engineering labor) is by far largest single cost. Cost analyses of past research and development contracts show that manpower typically accounts for 75 percent of total cost. Variations are geared to the ratio between purchased and fabricated items in the system. Where all subsystems are specially developed, the labor approaches 90 percent of the task, whereas systems configured largely from off-the-shelf units yield much lower labor percentages. If one could determine the percentage of purchased item costs, these items too would be seen to be largely labor burdened except where very high production rates exist. In software development, also, labor generally runs 75 percent and it can run as high as 90 percent of the task. Even in control system programming where certain adaptor hardware elements are attributed to "software" costs, about 80 percent of the cost is expended in labor, largely engineering.

From this it is apparent that, in either hardware or software systems development, engineering (or other professional) labor comprises the bulk of the cost. Furthermore, the differences between software and hardware system development labor rates are negligible. The professional engineer or scientist plays the major role in both.

Material costs are naturally higher for hardware system development, being in the order of 25 percent, as compared to about 5 to 10 percent for software materials. In software efforts, however, there are generally equipment rental costs involved to debug programs on a computer. These typically are 10–15 percent of total costs. Rentals are often treated as "material" costs, at least in purchasing procedures. With rentals so classified, the ratio of materials to total costs also becomes quite comparable for hardware and software systems.

In conclusion, all factors of production are close to equal for either hardware or software developments involving equal dollar volume. It is also apparent that the largest single cost by far is professional labor. Hence, the most precious ingredient of any system is the "intelligence" expended in its design and construction. Since any effort can be reduced to management of factors of production, management of software and hardware development tasks can be handled essentially in the same manner.

Method of Control

The instruments of control in any system development are managers and procedures. The managers of hardware systems and software systems are almost always technically or scientifically disciplined professionals, at least at the project management level. Salary levels and prestige are approximately the same for projects of equal dollar volume since they each involve approximately the same number of personnel in the same salary ranges. Hence, the only variable between hardware and software management is the set of procedures.

The procedures for hardware management are by far more standardized and apparently more effective than for software.[2] In fact, there appears to be no standard approach to large scale software developments. Although most software managers have learned to rely heavily on documentation for control, the manner of task specification, quality control, product standardization and change control is quite loose in many organizations. With an absence of formal procedures, success or failure in software tasks depends much more heavily on close personal supervision and the quality of such supervision.

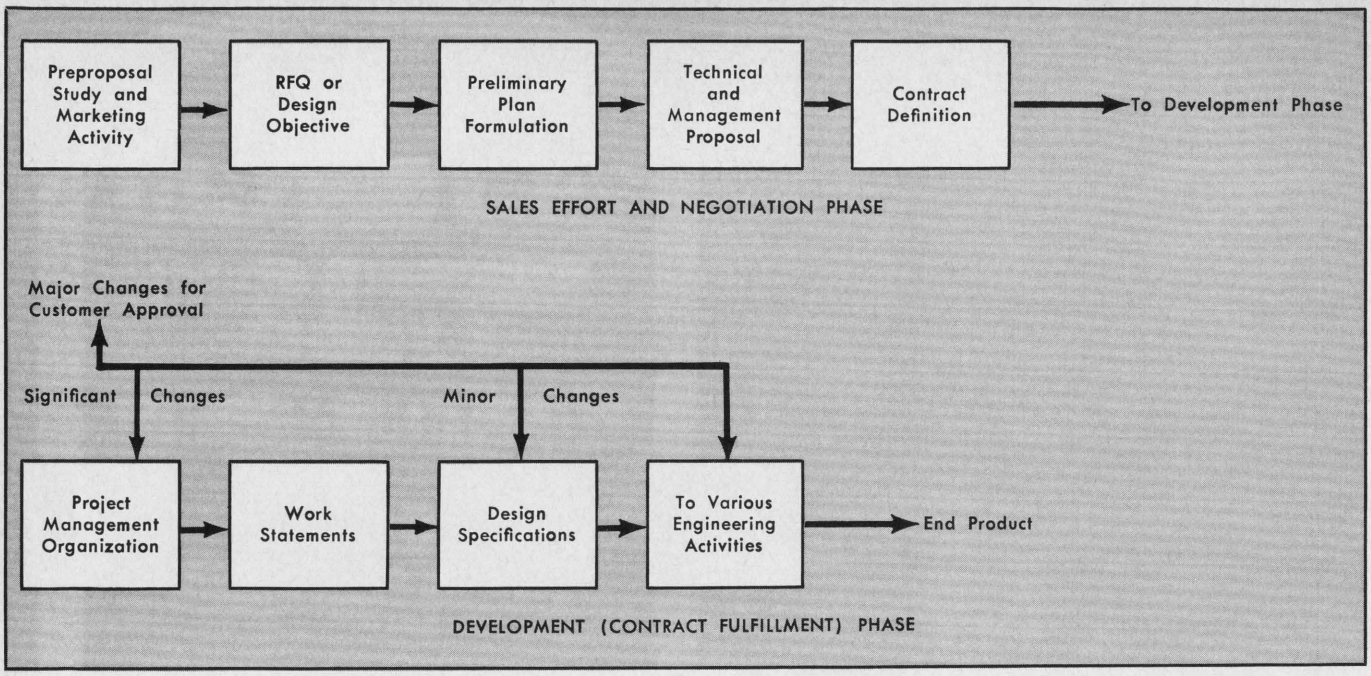


Figure 1. Documentary Control of the Engineering Design Process

While supervisory talent is a required commodity in any endeavor, heavy dependence on such a rare commodity is risky. It is not without justification that American Industry places as little dependence on personalities as possible, preferring to rely heavily on policy and procedure manuals. Software management should follow this example and shift some of the control from personal management to procedural management.

Management Control Tools

PERT and similar management control procedures are commonly used today. Many of the larger concerns are utilizing computers to implement these control procedures, thereby greatly facilitating correlation and dissemination of control data. Software as well as hardware elements of a system are included as milestones in the overall schedules. This has allowed broad control of tasks but the guidance for meeting individual milestones must come from more detailed technical direction. In hardware development, the process is well defined by formal documentation. A typical development process under document control is shown in Figure 1.

This process is a familiar one and need not be explained in detail. Two points are worth noting, however.

(1) Normal process flow is from customer requirement to end product with changes as a counter force. The more significant the change, the greater the impact and, hence, the further against the productive stream it flows.

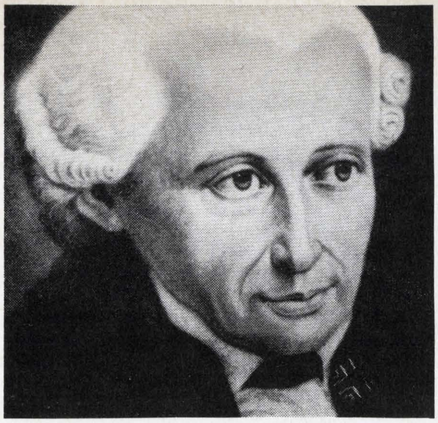
(2) The major point of contact between the management functions and the engineering effort is the work statement which references the design specification. The specification, therefore, provides the major technical direction from objective to accomplishment.

Advantages of Specification Control Techniques

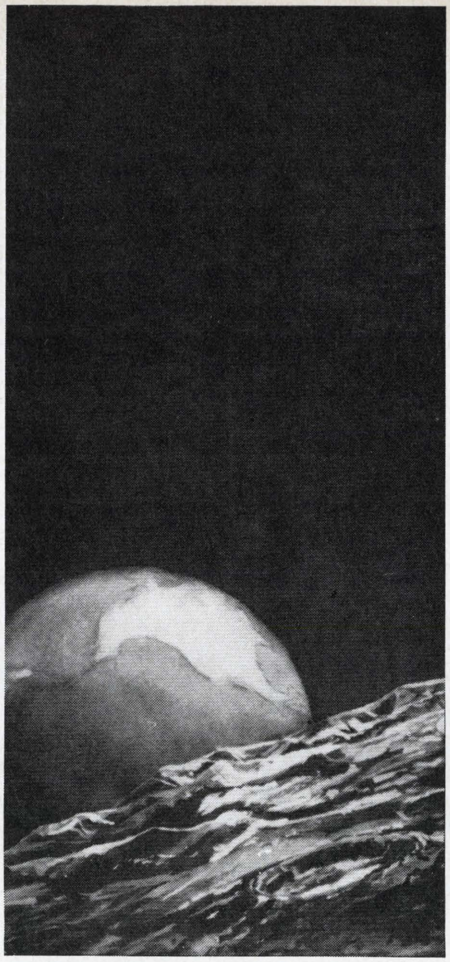
Borrowing from the hardware management discipline, the specification should be the basic control device in software development efforts. A brief review of advantages offered by specification control of hardware is worthwhile, for each advantage applies equally or more so in software management. A specification, when properly and timely executed, provides:

1. A clear, indisputable definition of the task requirements in terms of design objectives.
2. A legal base for negotiation.
3. A readily referenced source of guidance.
4. A logical means of subdividing tasks and interface handling.
5. A standard of measurement for progress review.
6. A standard of measurement for finished product performance.
7. A source of direction independent of personalities.
8. A means of standardization control.
9. A vehicle with which to build new designs based on past projects.
10. A means for maximum span of control by management.
11. A handy tool for orientation of personnel assigned to the project after formal orientation.
12. A basis of reference and control of inevitable changes.

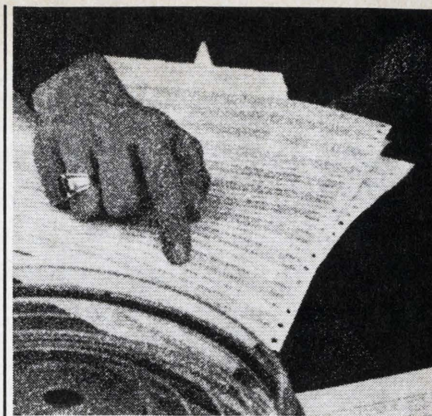
The advantages listed for specification control over less formal, personal control of hard-



If you're the kind of systems programmer who has a mind that's deep enough for Kant,



broad enough for science fiction,



and sufficiently precise to enjoy the esoteric language of the computer,

look into RCA. You're our kind of man.

You probably think far ahead of your time.

You want to take software out of its infancy. Into the 4th, 5th and 6th generations.

You want a chance to influence hardware design, instead of the other way around.

You want a variety of projects with definite objectives, instead of an endless task.

You want challenging work and inspiring rewards.

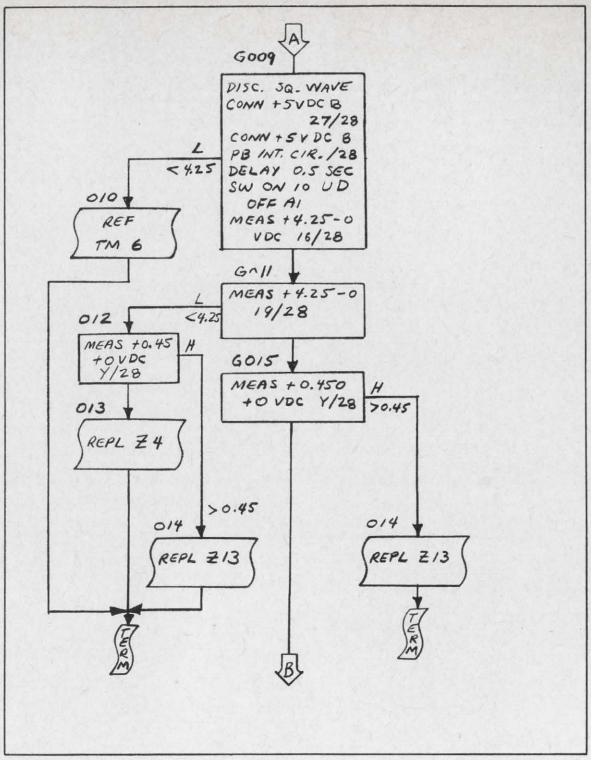
If this is what you're looking for, find it at RCA.

Write to us if you've had experience in language processors, operating systems, utility systems or communications systems.

We also have openings in Sales, Field Systems Support, and Product Planning and Engineering.

Contact Mr. J. C. Riener, Dept. SW-1, RCA Information Systems Division, Bldg. 202-1, Cherry Hill, New Jersey 08101. We are an equal opportunity employer.

RCA



Partial Flow Chart was used to help generate part of a computer listing.

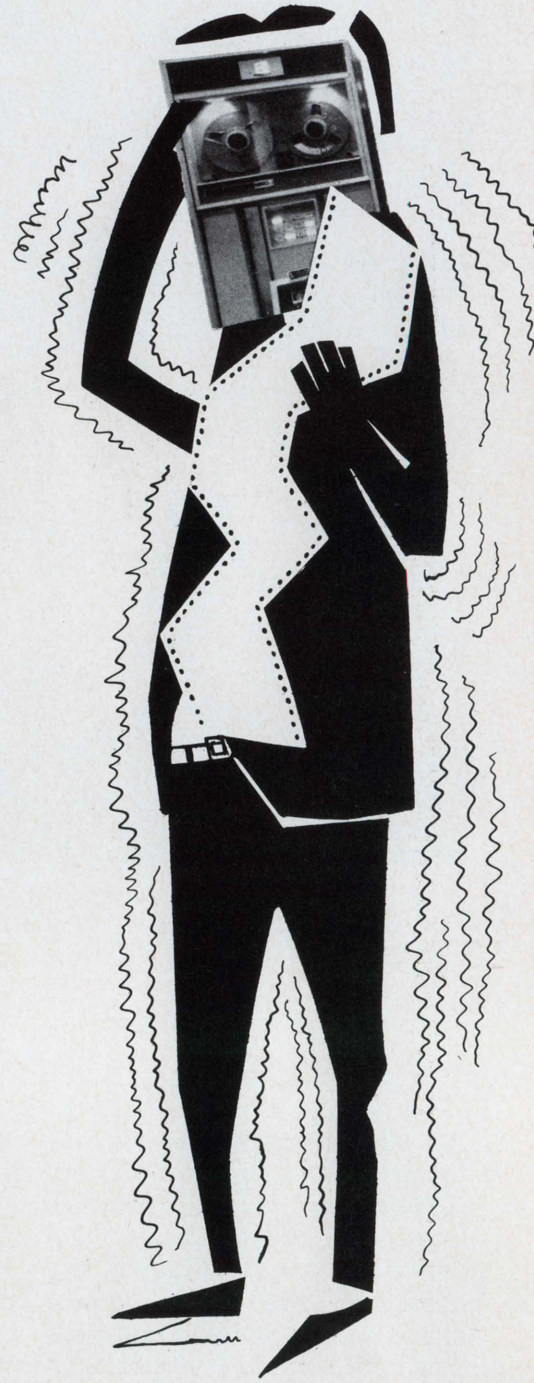
ware designs are fairly self evident. Rather than offering trivial explanations of these advantages, it is more worthwhile considering how some of the advantages listed have even more significant impacts on software design management.

Requirements for task definitions and a legal base for negotiation obviously apply to any endeavor and software takes no exception to these basic needs.

An item often lacking in software efforts is a readily referenced source of design guidance. In hardware efforts, system, subsystem and often standard circuit and module specifications set down both the performance objectives of the unit in question and even the methods of circuit implementation. In software efforts, no such tools are provided. Generally, a single work statement, vaguely worded, must serve as sole guidance to all tasks and subtasks. For a programmer doing a small segment or routine, this "big picture" objective offers little guidance. He must depend on his task coordinator to tell him what the precision requirements, interface with the main program, execution speed requirements and memory allocation must be in addition to the specific problem to be solved. Experience has shown it to be ineffective to rely on verbal direction given by subtask coordinators. With such an approach discontinuity generally arises among the program elements. Eventually such discontinuities are resolved and the program debugged, but an unnecessarily large part of the effort would have to be devoted to cut-and-dry techniques. Worse still, some subordinate functions of the program may be entirely

(Continued on page 32)

MANAGEMENT'S Rx PRESCRIPTION FOR 'COMPUTERITIS'



J. A. Salvail

Jack A. Salvail has been a career civil service employee with the Air Force and Navy for more than 20 years. He now serves as Special Assistant to the Technical Director of the Naval Command Systems Support Activity. He was project leader for *Management Data System, Planning Guide for Computer Program Development* and the *Selection of Technical Computer Personnel,* NAVCOSSACT projects.

■ Today the Computer Science and Management effort is plagued with a most serious illness. This illness is the inability to communicate. It has affected not only men working directly with the computer, but all men connected with the field of Computer Science and Management —technicians developing the system, salesmen, and those responsible for directing their activities and the overall business operations they effect.

This situation is encouraged by the variety of definitions individual computer technicians attribute to the functions of programming or analysis. Invariably, no two answers are identical, and in some instances a new title will be added to define the same function. For example, instruction is interchanged with coding as one function of programming. One never ceases to be amazed at this new, fast-growing vocabulary and its authors who have exceeded the time-worn medical profession in developing a language unto themselves.

Confusion within the computer profession has likewise contributed to confusion in dealing with outsiders. The lack of effective communication with users has been particularly harmful in creating mistaken impressions of the abilities of the computer, and in hampering understanding between computer professionals and users during system development.

The conservative Government Bureau of Budget has issued an automatic data processing glossary to assist in organizing the confusion of definitions in Government. Private industry is also taking steps in this direction, but much is yet to be accomplished. Progress has also been made in the managing and monitoring of system development, so that users can be more aware of the procedures involved. The following practical management prescriptions, representing much of the recent progress, are recommended to assist management as well as technicians in the care and treatment of "computeritis".

Prescription #1 is a word to the wise that hasty decisions to "computerize" are always costly, usually disappointing, frequently embarrassing and sometimes disastrous. Experience indicates that some officials may have been unduly influenced into making premature decisions in this area. The feeling that you don't comprehend "why"; that you're either "obsolete" or "stupid"; that the "halo effect" surrounding the concept of computer system operations and the impression that possession of an ADP system is the status symbol of progressive management; and that "everybody's doing it" are all factors that influence the making of premature decisions. The decision to "computerize" some data operations in your organization is NOT an easy one, and it should not be made without careful, objective and systematic deliberation.

We cannot question the revolutionizing influence of computers on information handling today, nor their tremendous, almost unlimited potential. However, automated data

processing is not a simple panacea for all data processing ailments.

- NOT ALL management problems are automatically banished by simply plugging in "black boxes".
- NOT ALL manual operations are necessarily slower, less efficient, and more expensive than ADP.
- NOT ALL organizations *need* ADP in order to accomplish their mission.
- NOT ALL data and record keeping operations are adaptable to ADP.
- NOT ALL organizations have enough ADP requirements to justify a computer system.
- NOT ALL organizations can *afford* ADP.

The whole process of effecting a computer application—from the initial decision to automate, through feasibility and application studies, equipment selection, systems design, programming, staff training, site preparation, conversion procedures, and testing to full ADP operation—usually requires several years, tens of thousands of man-hours, and hundreds of thousands (sometimes millions) of dollars in initial activating cost alone.

Most businesses of any consequence have several basic computer applications: payroll, fiscal accounting, personnel information, inventory, and taxes. Even with these, the purchase of a computer and, more important, the hiring of a staff to run and maintain the systems is very costly. Unless a business can utilize the computer at least 12 hours a day, the saving resulting from the installation is frequently disappointing.

Prescription #2 is that businesses with few applications or a questionable need of a computer installation should initially obtain the services of a Computer Service Bureau. During this introductory period, the firm can establish the data base, permit the members of the organization to become conversant with the method of obtaining and submitting computerized information, and review and evaluate the format, content, use, and frequency of output generated by the automated changeover.

After the foregoing period, if additional computer time and inhouse system applications are clearly evident, the purchase or rental of a computer should be seriously considered.

Should you succumb to "computeritis", the communication bit will continue to be a serious problem. How does Management make its needs known and control a function that speaks strictly in a foreign language and for which there is no standard measure of productivity? One has no dozen, no pound, no inch or gage of production progress.

Prescription #3 is designed to scratch the surface of a new plateau and prescribe the following uniform titles and definitions of phases and progress for the orderly development of a complete automated system. Each phase constitutes and defines an orderly, sequential step-by-step requirement in automating a manual procedure into a computerized operation or developing a so-called complete software package. If these milestones were adopted universally by businessmen and required of the computer industry, a positive bench mark and common basis of reference of work and progress would be applied across the field of Computer Science. Incidentally, the Government, the largest user of computers, has generally followed this phasing in developing an ADP system.

Development Phases

1. **System Analysis**—The process of evaluating and determining methods and procedures for currently doing the operation proposed for automation.
2. **System Design**—Designing, developing, and describing the methods and procedures to automate the operation from its present status.
3. **Program Development**—The flow charts, program designs, file arrangements.
4. **Coding**—The Computer instruction and desk checking.
5. **Check out**—Testing to assure the instructions are accomplishing the programs and planned output.
6. **Documentation**—A functional description of the system, (Users Manual), a Programmers manual, and a Computer Operators manual.
7. **Data Collection and Conversion**—Converting existing rec-

ords to acceptable automated data elements and establishing the data base.

8. **Assistance and Training**—Guidance, instructions, and re-orientation of personnel and procedures associated with installation, maintenance, and use of newly automated system.

The adoption of these phases and their definitions facilitate a better understanding of what is being done and what is expected in the orderly, progressive development of an automated system.

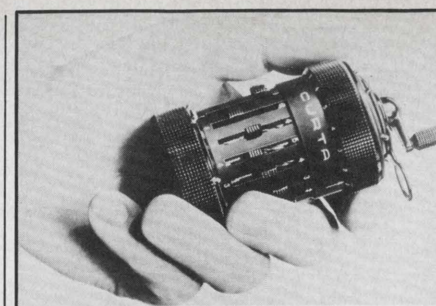
Presently, an average total software system of a single business operational phase may take 9 months to over a year to develop. Upon inquiry the system technician would now indicate to a management official that he is in "analysis" at various intervals over a 3 to 4 month period and subsequently in program development over the next 2 to 3 month period. This is very taxing to a manager's sense of humor and hardly helpful in defining progress, whereas the proposed phase description of progress or status of development would be acceptable and generally understandable to all concerned.

The phase system will also identify programming changes to existing systems or feasibility or similar system studies that may be initiated. Changes to an existing system are identifiable under the phases associated with programming as described in two documentation manuals (Programmer's manual and Operator's manual). A feasibility study is identified as the analysis phase and is documented by the management functional description. By applying the phase identification, both parties involved understand and identify what is expected and the relative degree of progress or delays. Accordingly, misunderstanding during development of an automated system is materially aleviated.

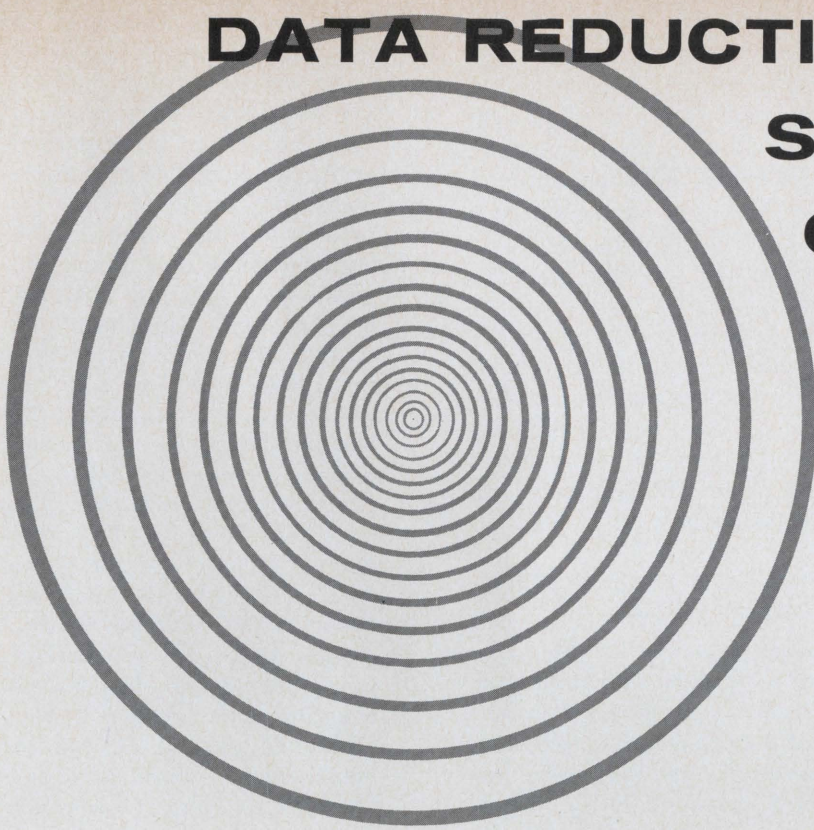
Prescription #4, and a must product of the analysis function at the conclusion of system analysis and system design phase, is the submittal of a "management's functional description" of the proposed system. This is one of the three publications in the documentation phase effort. It is essential that this document be written in a language understandable by Management. This document is the functional description and work flow of the changes in input, output, and current procedures of operations necessary to automate or computerize the operation. It includes information concerning form design, implementing methods, and procedures as well as any organization and personnel changes.

Prescription #5 points out that it is essential for Management and technicians to agree entirely on the foregoing management automation plan described in the function description and that no or few minor changes be made in the procedure after agreement and commencement of the programming phase. Any changes should be resolved prior to commencement of programming. Repeated changes during programming are very costly, can delay the delivery schedule, and have been known to cause programmers to grow beards and wear torn sneakers. It is wiser to make changes after the system has been totally developed.

Prescription #6 and the last word to the wise, but one of the most important: the best engineered system in the world is only as good as its data base, a fact frequently overlooked by managers or deemphasized by operating officials. Therefore, initially, put your chief executive in charge of establishing your data base. This could prevent refunding $6000.00 to an already overdue customer to ordering 20,000 typewriter ribbons for your secretary. ■



DATA REDUCTION THROUGH SAMPLING IN COMPUTER SYSTEMS



Donald V. Mathusz

Donald V. Mathusz, 37, is currently an Operations Research Analyst at the Annapolis Division of the Naval Ship Research Development Center, Annapolis, Md. He has directed or been a consultant on projects dealing with data systems design, simulations, statistics and reliability. He received a BIE degree from Syracuse University in 1960; the Masters in Management Engineering from Rensselaer Polytechnic Institute in 1963; and he is currently pursuing the Ph.D. at the University of Maryland.

INTRODUCTION

■ The populace in general is fascinated by the speed with which computers can digest data. Many systems people fall into the same pattern of thinking, and so—do not consider (seriously) techniques of data reduction. These techniques can reduce work at many levels, from the systems design stage through implementation problems, to actual alternative processing methods that could save substantial CPU time. In this paper, we will discuss sampling in a general fashion, as a data reducing technique.

Part of the orientation problem is that very little has been written about actual applications in the general business orientated data processing vernacular*. Mathematical statistics, upon which sampling depends for its theory, is only now coming into vogue in our business schools. For this reason, few business orientated systems people presently have any mathematical statistics background. (The Engineering/Sciences areas are a bit better off, with one statistical course in the general undergraduate curriculum.)

The body of the available literature, however, tends to the graduate level. This is a bit high for reference to a first try at understanding and possible applications by the ordinary analyst. A bibliography of references is included for those staunch enough to advance into the area. This bibliography is not expected to be complete, but Tukey's article examines the current state-of-the-art under the broad heading of data reduction, while at the other extreme, Moroney's well-known little book is about as fine an introduction to the subject matter as can be had. The Grant and the Cyert & Davidson books are of a higher, intellectual level, (but still very reasonable), while the others get a bit more involved.

One can hardly escape the fact that I have not gotten into the actual techniques of sampling itself. Here, I would agree with most statisticians that a little knowledge can be harmful. What I am attempting to do, is give the average systems analyst insight into when it might be very profitable to call in a consultant. Please note that you may well have people in-house who can be consultants, at least for situations that only call for simple random or possibly simple strata sampling. These people are often found in Statistical Quality Control, Reliability, Operations Research and Marketing Research . . . Hopefuly, they should also know when to call in outside consultants.

All this is to point out the subject matter is admittedly involved, but not impossibly so, and certainly not to the point that its lack of general use would have us believe. Next we will take up the basic criteria of how we can recognize when we have a possible sampling application, or a tiger by the tail.

CRITERIA FOR DATA SAMPLING: AUDATA OR STODATA:

A first easy decision to make is, are we processing data which must be accounted for in the "balanced book" audit trail—accounting sense? If the answer is yes, we have audit type data or audata (acronym). On the other hand in many cases, (espe-

* *A recent exception is Arkin, A., article in the Journal of Accounting, October, 1955, pp. 44–48.*

cially in management information systems), we only require average values of some sort. Average values are ready made for a possible sampling application, since they are by their own nature statistical variates. However, any type of output data not audata is a sampling candidate before or during the run. This data not audata, we will call stochastic data for generality, which implies that a direct sampling possibility exists in the processing of its stodata (acronym). Sampling possibilities exist at the various levels of system development involving audata, but not to the same extent.

SAMPLING-DATA REDUCTION BY DATA SYSTEM STATUS:

Sampling, by no means has to be confined to internal program processing. Its basic purpose is to reduce a mass of data to a few simple numbers descriptive of the mass data set, which incidentally is a goal characteristic of management information systems. This broader concept of data and processing reduction is the one we wish to emphasize here. It extends from system design feasibility studies to audits of implemented systems. We will next describe some broad classes of systems design and processing status in which sampling may be useful.

1.0 Systems Design Feasibility: In systems design it sometimes occurs that the range of possible or likely input values of a parameter are unknown at the design feasibility stage. Quite often, it is possible to arrive at good estimates of this range by sampling existing files, and calculating confidence intervals on the sampling mean of the parameter. This for example, could be critical in deciding the field size requirements in information retrieval modules of any system.

2.0 Design for Input Data Reduction: If all of the required data outputs are Stodata, there is a good case for reducing the input data by sampling before it ever reaches the keypunching stage. This approach reduces all work but some initial programming. (See note on bottom of Table 1.)

3.0 Implementation Sampling: In some large systems, important questions remain after the basic system design and programming have been accomplished, as to the extent of the implementation problem. One approach to this problem is to implement the operational programs on a sampling basis i.e., over sections of the files to controlled, defined alpha range of input data et al. This approach to debugging eliminates the massive interference to existing files and procedures that results from the usual early debugging requirement of running dual operating systems simultaneously throughout the whole new system debugging cycle.

4.0 Normal Processing—Data Tapping: During many business accounting type processing runs, additional data is desired for management information purposes in the form of averages per some set of categories. It may save substantial CPU time, to program an interrupt sampling scheme that would only process during the interrupt. The CPU trade-off is total marginal processing versus interrupt processing time. (Note: The possibilities inherent in multiprocessing mode of computer operation using this approach.)

4.1 Special (Data Tap) Processing of a Standard File: Management information systems may be able to tap existing tape or disk files to obtain data input where the required output is of the Stodata type, say a series of averages. Sampling then becomes a logical approach.

5.0 Normal Processing: During a normal run, it may be desirable to establish that the exact (audit trail) sequence of calculations is correct. Rather than force a series of dumps, some output (preferably a printer) could be programmed to print out the audit trail of the calculation in process, on a sampling interrupt basis. (The interrupt could be a sense switch type which would give the auditor a great satisfaction of flipping it himself and this could be done on a random time basis.)

TABLE I

	SYSTEM STATUS	SAMPLING OBJECTIVE	GENERAL EXAMPLE	SAMPLING SCHEMES	PROGRAMMING REQUIREMENTS
1.0	Systems Design Feasibility (Before Systems Design)	Estimate input parameter ranges, (from either AUDATA or STODATA)	Establish if the standard A, B, C inventory categories are practical and estimate the size of each	Simple Random to Strata-Cluster	Generally none unless 4.0 or 4.1 involved
2.0	Design for Input Data Reduction (During Systems Design)	Reduce input Data for CPU savings, (STODATA output only)	Management Information e.g., changes in average service call time or costs	Same	None for Systematic sampling* others may require some. See Yates 60.
3.0	Implementation Sampling, (After Design, Before Implementation)	Sample System Implementation Problems (AUDATA or STODATA Systems)	On master tape merges, determine mismatch percentage/on disk-determine percentage overflow in Index Sequential Mode	Tapes generally Strata-Cluster/ Disks generally simple random	Little; either to merge register on tapes, or random number inquiry status
4.0	Normal Processing —Data Tapping	Sample Stream of AUDATA or STODATA (for Secondary STODATA use)	On a payroll run, type of the average income per professional category or department	Generally Simple Random	Amount Dependent upon sampling scheme employed
4.1	Special (Data Tap) Processing of a Standard File	Sample existing file (may be either AUDATA or STODATA)	On a standard billing file, extract, average bill per mailing zone	Simple Random to Strata-Cluster	Same**
5.0	Normal Processing	Establish proof of auditable process (AUDATA only)	Select on a sample basis, data Audit Trails to be followed for Auditors benefit	Simple Random	Simple or could even be done by time samples via sense switch

** A machine is available which sorts out every "n"th document or IBM card, in effect systematic sampling, Pitney–Bowes "Tickometer".*

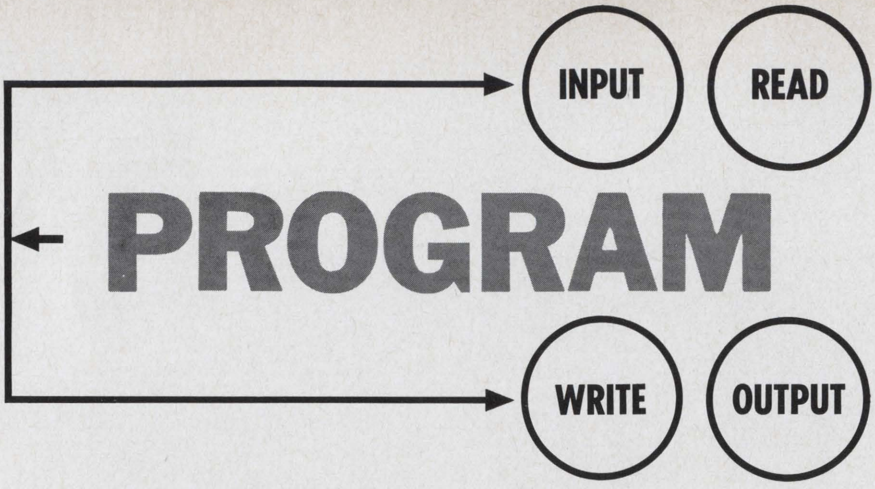
*** Simple random time dumping of an existing tape file using one of the standard utility sense switch controlled tape dump routines, should be considered for one time only estimates, as in conjunction with 1.0 above.*

TABULAR LIST OF APPLICATIONS BY SYSTEMS DESIGN STAGE:

The above list is meant to summarize the above arguments and serve as an interest stimulator in the possibilities that abound. I am certain it is capable of great expansion both by experience and good thinking. ■

BIBLIOGRAPHY

1. *Cochran, W. G.*, Sampling Techniques, Wiley & Sons, N. Y. 63
2. *Cyert, R. M.* & *Davidson, H. J.*, Statistical Sampling for Accounting Information, Prentice–Hall, N. J. 62
3. *Eisenhart C., Deming L.*, and *Martin C.*, Tables Describing Small-Sample Properties of the Mean, Median, Standard Deviation and Other Statistics Technical Note 191, National Bureau of Standards
4. *Grant, E. L.*, Statistical Quality Control, McGraw–Hill, N. Y. 52
5. *Hansen, Hurwitz, Madow,* Sample Survey Methods and Theory, Vol. I and II, Wiley, N. Y. 53
6. Military Standard, Sampling Procedures and Tables for Inspection by Attributes, MIL–STD–105C, 18 July 1961
7. *Moroney, M. J.*, Facts from Figures, Pelican Books, London 54
8. *Tukey, John W.*, "The Future of Data Analysis", The Annals of Mathematical Statistics, Vol. 33, No. 1—March 1962
9. *Yates, F.*, Sampling Methods for Censuses and Surveys, Griffin, London, 1960



PROGRAM DOCUMENTATION

THE VALUABLE BURDEN

Ned Chapin, Ph.D.
Data Processing Consultant
InfoSci Inc.

Dr. Ned Chapin received his Doctor of Philosophy in Business and Economics with a minor in Industrial Engineering, 1959, from Illinois Institute of Technology. In 1949 he earned a Master of Business Administration from University of Chicago. Dr. Chapin has written two books on computers and dozens of articles in various publications. He has taught systems, data processing, computer programming, production control and other business and engineering subjects in college. At present he is a data processing consultant.

Cost and Value

■ In this age of software, good software writers typically hate to do documentation. The challenge and stimulation that makes programing fun is missing from doing program documentation. Most software types regard program documentation as a chore—a necessary one perhaps, but nonetheless a distasteful one. Program documentation is part of the burdensome overhead price a software writer pays to keep his boss happy and to meet installation requirements.

Besides being a burden to produce, most program documentation is a burden to read. The average job of documentation done at most installations is incomplete, inconsistent from program to program, and in a format that is difficult to decipher. Essential parts are usually in no set order or arrangement. The result is that the programer must search through the program documentation materials in order to locate the information which he needs, if that information is present at all.

Any programer or analyst who has done program maintenance work, or who has been assigned to revise a program that even he himself wrote a year or more before, can testify to the value of complete, accurate, and comprehensive program documentation. Well done, program documentation enables the programer or analyst to grasp quickly the algorithms used in a program, to find the portions of the coding which implement the various portions of the algorithm, and to assess quickly the consequences of a change of input data or format, of the output data or format, or of the algorithm itself. Good program documentation enables the programer or analyst to strike quickly to the heart of program maintenance work and accomplish it efficiently.

The absence of good program documentation leads a programer or analyst to rethink the algorithm, to ferret out the details of the ways in which the input format and data affect the algorithm, and the operation of the program, and to sleuth relentlessly after the reasons why some unwanted data appear mysteriously in the output. Rethinking is hard work and very time consuming. The joy of original creation is rarely present. The organization gains no more from having a program corrected at a cost of many man hours of rethinking, than it does from having a program corrected quickly and efficiently through the assistance of good program documentation. The benefit to the organization is the same. But the cost to the organization can be drastically higher in the absence of good program documentation.

The advantages of program documentation are so well known that program documentation is given lip service in almost every computer installation. Rare indeed is the analyst, programer, programing supervisor, operations director, manager of data processing, or manager of systems who decries documentation. Most sing its praises, and note its defects—or even occasionally its near absence—in their installations.

A few voices can be heard that program documentation is unneeded for COBOL, FORTRAN, PL/1, ALGOL, and the programs written in like languages. The language, some claim, is self-documenting, and hence no further documentation is needed. This is a very narrow view of what documentation comprises. To see the full picture, let us review briefly the objectives of program documentation, the advantages, the levels, and the elements of program documentation.

Objectives

Objective #1 of program documentation is to provide a written record of what the program directs the computer to do and how the computer is to do it. Since the program controls the computer's operation, information about what the program directs provides information about the way the computer performs its work, but not necessarily about what that work is.

Objective #2 of program documentation is to provide a written record of what the computer accepts as input and what it produces as output. This information tells what work the computer does in a way that reveals how this work contributes to and is a part of the over-all system. This information describes what the raw materials are that the computer uses, and what the products are that the computer produces from them.

Objective #3 of program documentation is to provide a means of communicating about a computer program from one human being to another. Specifically, program documentation is *not* concerned with providing information to or from a computer or other machinery about a program. Program documentation is designed for human consumption only.

Objective #4 of program documentation is to provide information towards four specific ends: documentation and program status, man-machine interface, data specifications, and program logic. Appropriately defined, these cover the major topics.

Thus, program and documentation status inform about when the program was revised, by whom it was written, of which system it is a part, and how fast it executes. Who authorized or made changes and when they were made is also part of the status information.

The man-machine interface description informs about human performance needed to make the computer and peripheral equipment operate correctly. This describes what a person must do to enable adequate performance from the computer.

The data specification information pictures the data handled by the computer. The contents of records, the data formats, and the control data used are part of this information.

The program logic information delineates the way the program directs the computer to take the input data, transform it, and produce from it the output data.

Advantages and Documentation

Helps make a program belong to an organization. Programs are collections of symbols which in and of themselves typically have no obvious meaning. The meaning is known only to the creator of the symbols and not known to others in the organization. Program documentation restates these symbols which have private meanings to the programers who created them, into symbols that have public meaning broadly throughout the organization. Without documentation a program is rather like a dog who obeys only one master. But programs are created by individuals for the benefit and use of their organizations.

Helps the programer think clearly about program structure and choice of algorithm. When the programer knows that what he specifies in his program will be presented for public view, he usually feels more compelled to search for and utilize clearly thought out and efficient algorithms for accomplishing the work of the program. A well reasoned, adequate, and comprehensive approach stands inspection better than something put together on a hit-or-miss basis. With program documentation the result typically is better programing, faster operating time, and fewer bugs.

Helps protect the organization against forgetting. When the programer who wrote the program initially is assigned to other work, the forgetting process begins. Within a few months, the programer typically has forgotten major significant details, and within years most of the details about a program he has written. Yet with program documentation, the organization has available all of the facts in full detail at all times.

Helps reduce the costs of personnel turnover. When a person who created the program resigns, falls ill, goes on vacation, takes a leave of absence, or is promoted or fired, knowledge about the program often leaves with him. Program documentation keeps that knowledge about the program in the organization, and thus helps protect the organization against the cost and time of redoing work.

Assists in training new programers. When programers are added to a staff or assigned to new work, program documentation provides information about other programs in use in the organization and about how programing is done in the organization. As such, it saves valuable

supervisory and training time in getting new programers to be productive.

Assists in upgrading programers. The senior programers' "tricks of the trade," their techniques, and their successful approaches can be studied and copied by all programers through the medium of program documentation. This permits each programer to learn from all the others.

Assists in estimating the extensiveness of program changes. Program documentation lays out all the facts about each program that are relevant and significant points. It provides the basic data required for evaluating and making program changes. Each place in the program that needs to be changed and the way in which it needs to be changed to meet the new requirements can be more readily found with program documentation.

Helps decrease the cost of debugging. With all the facts about a program laid out before him, the programer can more easily detect, diagnose, and correct errors in the program. This assumes that program documentation is begun prior to the time that program testing is begun. If this be so, then the programer has been called upon to anticipate all the vital facts about the program. This very anticipation often catches mistakes.

Encourages the use of standard coding conventions. For example, the programer's assignment of names to data and to parts of the program are more likely to follow standard practice if the programer's work is open to public inspection in the organization via program documentation. Standard approaches in methods of calling subroutines, handling of input and output errors, initializing input areas, and the like, are more likely to be followed when program documentation is done consistently.

Encourages standardization in the description of computer operations. This is made possible in part by the use of standard formats and standard terminology in the documentation itself. This, in turn, encourages programers and analysts to think about problems in terms of standard building blocks, standard formats, and standard terminology in the organizations. This helps speed up and sharpen problem definition and tends to reduce programing time.

Aids in scheduling computer operation in an organization. How long a job will run, what factors affect the timing, and by how much, are more easily evaluated when program documentation exists in good form.

Partially documents the system. Since the system typically is built in major parts of computer operations, the documentation of the individual computer operations taken collectively constitutes a partial documentation of the system.

Levels of Documentation

Program documentation typically is useful in an organization at three levels: general overview, computer operations, and program and system maintenance. Each of these levels of documentation can be used by persons primarily concerned with other levels, as for example, a programer may use data at several levels for different purposes.

At the overview level, persons who use program documentation typically seek a general understanding of what the computer does and how it does it, what it accepts as input, and what it produces as output. The overview level provides the broad picture.

At the operation level, those who use program documentation seek direction on how to set up the computer and peripheral equipment to make a run. They seek detailed information about the data and equipment to be used, and options available, and the operator intervention alternatives, for example.

At the maintenance level, those who use program documentation seek typically general and detailed information about what the program directs the computer to do, how it directs it to do this, and upon what specific items of data the operations are to be performed. Full and complete coverage of all details is needed in a well organized form.

Major Documentation Elements

Revision Status. This provides an introduction to the rest of the program documentation because it provides information about the currency of the documentation and the position of the program in the system. As such, this element of program documentation provides information at the overview level about status.

System Charts. A system chart or system flowchart preferably in U.S.A. Standard format provides graphically the basic information about the input and output for the program. As such, this element of program documentation provides information at the overview level about status, format, and logic.

Program Description. A description in words, preferably non-technical, about what the program does, what it uses as input, what it produces as output, and how it produces the output from the input, serves as a useful accompaniment to the system chart. The program description provides information on the overview level about format and logic.

Operation Sheet. The operation sheet supplies detailed information about how to set up the computer for runs of the program, what data are to be used as input, how to call the program, how to set up each item of peripheral equipment used, and how to label the output. This element of program documentation provides information at the operation level about the man–machine interface.

Halt List. The major options open to the console and peripheral equipment operators are sometimes identified by computer halts. The halt list enumerates these specifically giving the reasons for each and the action alternatives available. As such, this element of program documentation provides information at the operation level about the man–machine interface.

Card Formats. For the computers that do not utilize punched cards as a primary medium of input and do not utilize the punched cards for control input to the operating system, information on card formats is irrelevant. Most computers and most operating systems in use today use punched cards either for data input or for the control input. The card formats provide the detailed information about the cards needed and the fields on each. As such, this element of program documentation provides information at the operation level about format, logic, and the man–machine interface.

Input–Output Formats. Information about the data formats in storage at the time of input or output is typically shown in the input and output format. As such, this element of program documentation typically provides information at the maintenance level about format and logic. For greater usefulness, input–output formats normally show only those fields which are utilized or active in the program.

Algorithm. Any of several devices can be appropriately used for the task of stating in condensed form the algorithm utilized in the program. In some circumstances, a mathematical symbolism may appear appropriate. In others, decision tables provide a convenient way of expressing the algorithm. Occasionally, other devices are useful. In any event, this element of program documentation provides information at the maintenance level about status and logic.

Summary Flow Diagram. The summary flow diagram or summary program flowchart serves as a directory or index to the detailed flow diagram. This diagram is most useful if it is prepared in U.S.A. Standard format, and uses the recommended notations. As such this element of program documentation provides information at the maintenance level about logic.

Detailed Flow Diagram. The level of detail in the detailed flow diagram or detailed program flowchart is normally determined by the level of detail in the coding of the program. Thus, the level of detail for a program written in a simple symbolic language is normally more detailed than is the level of detail for a program which was written in COBOL, FORTRAN, ALGOL, or PL/1. In either case, the diagram and the program list should cross-reference. The detailed flow diagram provides a graphic picture of the logic of the program. It is most useful is presented in U.S.A. Standard format using the recommended notations. As such, this element of program documentation provides information at the maintenance level about logic.

Sample Input–Output Forms. Sample input–output forms provide graphic examples to programers, analysts, and operators about the appearance of correct input and output data. As such, this element of program documentation provides information at the operation and sometimes the maintenance levels about format and the man–machine interface.

Program Listing. The most useful program listings are those that show both the source and object language, and a symbol table cross-reference. This is because the maintenance work sometimes involves making object level language changes without recompiling in order to compensate for deficiencies in logic, or in compilation, or to handle situations not adequately provided for by the source language. Easy cross-reference to the diagram is also very helpful, as are comments in the coding. This element of documentation provides information at the maintenance level about the logic.

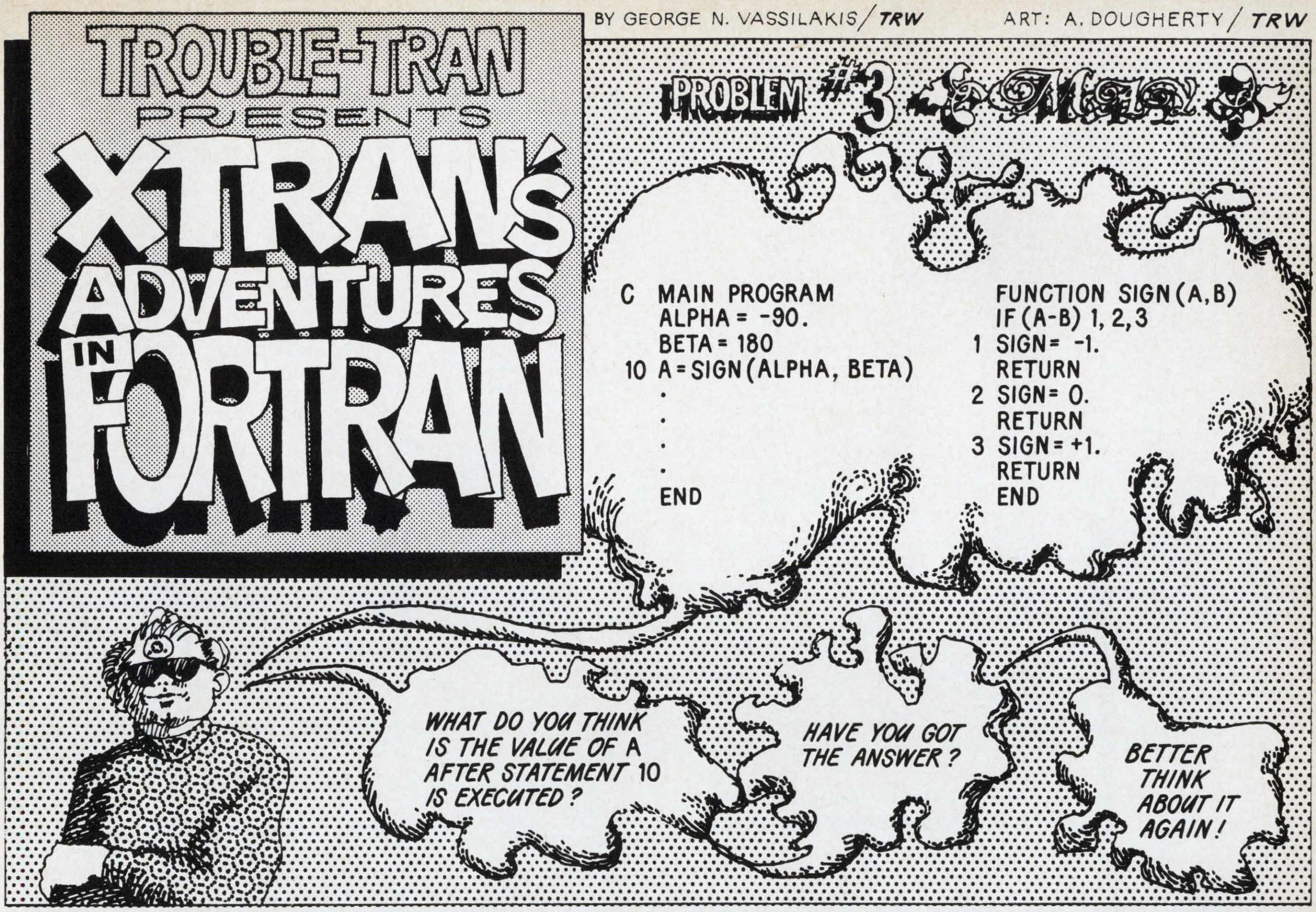
Conclusion

These twelve elements of program documentation imply a philosophy of computer operation. That philosophy is that the programer tells the computer operator how the computer is to be operated, and not vice-versa. By contrast, in some documentation schemes, it is the task of the console operator to build the documentation by stating how he succeeded in getting the computer to run the job successfully. He is provided with forms on which he is asked to report how he set up the job, what he did at each halt, and what the results were.

The documentation approach presented here is particularly applicable to those installations using operating systems, or those in which the programer and the operator are clearly separate persons. Under these conditions, the console operator can note on the console log sheet any unusual conditions that arose and can make a reference to the program documentation if needed. Thus, the console log sheet, rather than the program documentation, becomes the record of what the console operator did.

Under the approach presented here, program documentation is the task of the programer. It is a burden on him at the time of programing and debugging, since he must create it to reflect a programer's knowledge of and approach to programing. At its best, the approach goes back to the basic objectives of documentation. The record of what the program does and how it does it, and what the program accepts as input and what it produces as output must be complete enough and accurate enough to serve at the three levels noted. The intensiveness of use of these levels depends upon and varies from organization to organization. But, regardless of the intensiveness of use, program documentation is basically to communicate easily and intelligibly the important facts and relationships that distinguish a program. ■

BY GEORGE N. VASSILAKIS/TRW ART: A. DOUGHERTY/TRW



TROUBLE-TRAN

for fun and profit

Send your ANSWER to the problems posed here in each issue to:

TROUBLE-TRAN EDITOR

software age

1020 Church St., Evanston, Illinois 60201

You can also profit by submitting PROBLEMS for this feature. If your problem in FORTRAN programming is selected for use in this feature, you will receive $25.00

Contest Rules:

1. ASA standard FORTRAN IV is assumed.
2. The IBM 7094 FORTRAN IV compiler is used in verifying answers.

The correct answer bearing the earliest postmark will net the reader submitting it $25.00

The second correct answer with earliest postmark wins $15.00

XTRAN'S ANSWER TO LAST MONTH'S PROBLEM

PROBLEM # 2

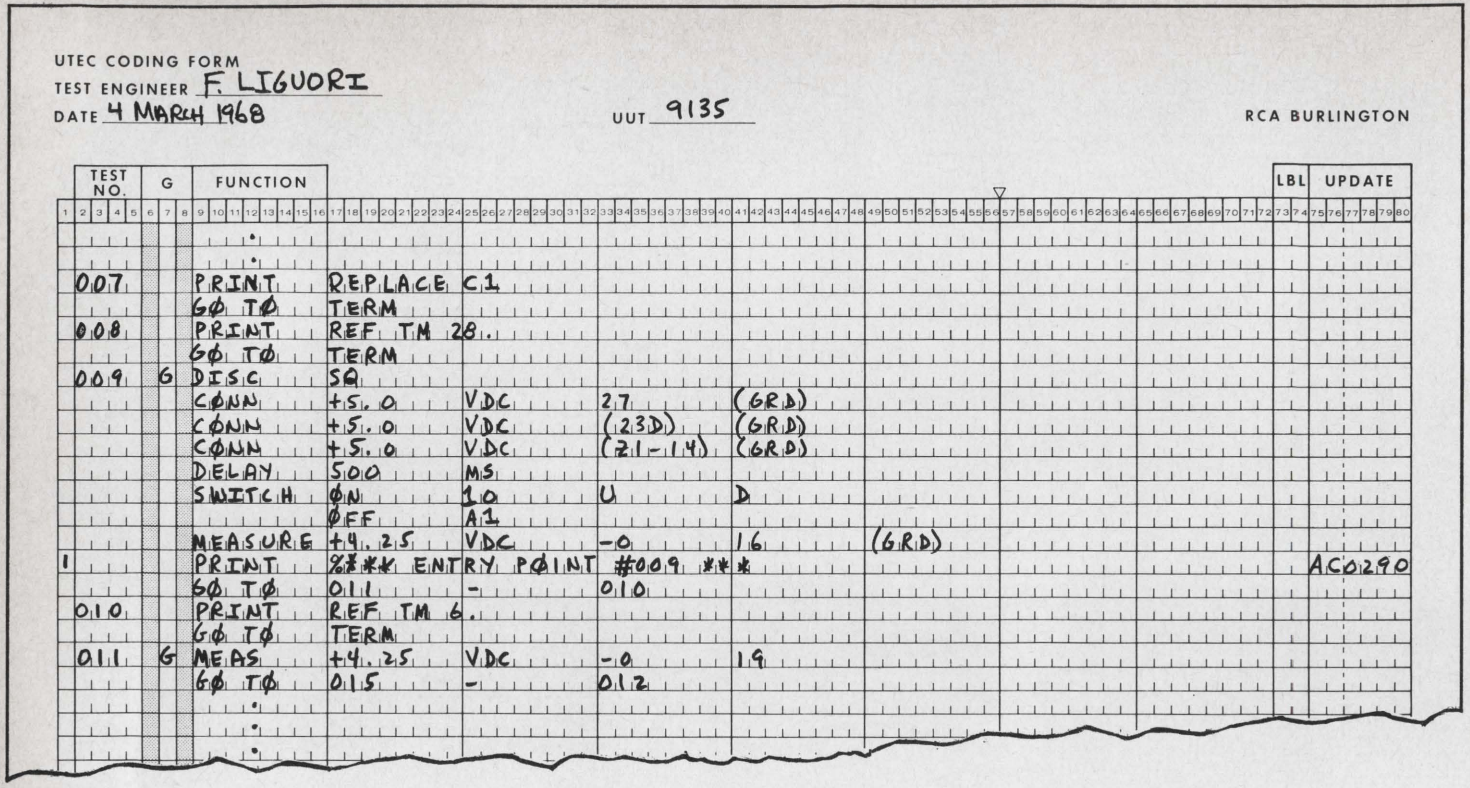
THE PURPOSE OF THIS PROBLEM WAS TO EMPHASIZE THE FACT THAT "UPON EXITING FROM A DO *BY SATISFYING THE* DO, *THE INDEX MUST BE REDEFINED BEFORE IT IS USED IN COMPUTATIONS." WHEN WE WRITE* DO 100 I = 1, 10 *THE INDEX VARIABLE* I *IS INITIALIZED WITH* 1, *AND THE* 2's *COMPLEMENT OF* I *IS PLACED IN AN INDEX REGISTER. LOOPING IS CONTROLLED BY THE INDEX REGISTER AND* NOT *THE INDEX VARIABLE* I.

NOW LET US LOOK AT THE PROBLEM ONCE MORE.

```
      DIMENSION X(100), Y(100)
  100 FORMAT (4E15.5)
      K=1
      DO 20 J = 1, 50
      KK = K + 1
      READ (2, 100) (X(I), Y(1), I = K, KK)
   20 K = I + 1
```

THE READ *STATEMENT INCLUDES AN IMPLIED* DO, *WHICH IS EQUIVALENT TO A* DO *STATEMENT. SINCE LOOPING IS CONTROLLED BY THE CORRESPONDING INDEX REGISTER AND NOT BY THE INDEX VARIABLE, THE VALUE OF* I *AFTER EACH* READ *IS EQUAL TO* K *AND NOT* KK.

LOOKING INTO ARRAY X, *WE SEE THAT ON THE FIRST* READ *WE READ INTO LOCATIONS* X(1), X(2). *ON THE SECOND* READ *WE READ INTO LOCATIONS* X(2) *AND* X(3). *AS YOU SEE, EVERY SECOND PAIR OF DATA IS LOST, WITH THE EXCEPTION OF THE LAST ONE. TO CORRECT CHANGE* : 20 K = K+2



UTEC CODING FORM

TEST ENGINEER F. LIGUORI

DATE 4 MARCH 1968 UUT 9135 RCA BURLINGTON

TEST NO.	G	FUNCTION						LBL	UPDATE
		.							
		.							
007		PRINT	REPLACE C1						
		GØ TØ	TERM						
008		PRINT	REF TM 28.						
		GØ TØ	TERM						
009	G	DISC	SQ						
		CØNN	+5.0	VDC	27	(GRD)			
		CØNN	+5.0	VDC	(23D)	(GRD)			
		CØNN	+5.0	VDC	(Z1-14)	(GRD)			
		DELAY	500	MS					
		SWITCH	ØN	10	U	D			
			ØFF	A1					
		MEASURE	+4.25	VDC	-0	16	(GRD)		
1		PRINT	%*** ENTRY PØINT #009 ***						AC0290
		GØ TØ	011	-	010				
010		PRINT	REF TM 6.						
		GØ TØ	TERM						
011	G	MEAS	+4.25	VDC	-0	19			
		GØ TØ	015	-	012				
		.							
		.							
		.							

The source language coding is used for generating the RCA Universal Test Equipment Compiler (UTEC) language program.

SOFTWARE MANAGEMENT

(Continued from page 14)

overlooked in the design process. These must then be "patched" into the program, often at decreased program execution efficiency in order to minimize the interaction on the completed sections of the program. By reducing each subprogram task to a separate, though subordinate specification, all requirements may be defined in detail at the outset.

Task subdivisions in hardware design tasks usually fall into neatly defined physical unit boundaries. This is seldom the case in software efforts. Software can be broken down only into a small number of separate programs. One may be for system pre-acceptance testing, another for operational control and a third for maintenance. Beyond such major divisions, there is little natural subdivision. When no definitive specifications are imposed on the task coordinate, the problem is often handled by assigning a major task to a task force of several programmers, all of which have overlapping responsibility. In such circumstances the more ambitious member of the team (not necessarily the task coordinator or even the most capable programmer) must bear the brunt of the task. By substituting indigenous leadership for documented directives a good deal of inefficiency often results. Some work, incompatible with the final approach, is necessarily scrapped while other parts are reworked. Specification control provides an explicit means for subtask definition, ensuring compatibility by facing the problem at the outset rather than when well along the design path.

Without a detailed work statement and subordinate specifications, attempting to ascertain percentage of completion of a software task is generally a guessing game. This problem arises from the lack of visual evidence of accomplishment in partially completed programs. Generally there is far more paper work observable part way down the road to completion than when the program is finished. This is due to the overlapping coverage of subtask programmers, extensive rework and alternate coding generated to cover alternative approaches. All of which point to a lack of definitive direction in the detailed design stage. In hardware tasks, pieces fit together more naturally. Hence, the end of one phase and start of another is more definite, thereby leading to a simpler, more accurate assessment of percentage of completion. If software were generated in conformance to a network of subordinate specifications, fulfillment of such specifications would provide an accurate measure of accomplishment.

Finished product performance is judged against an overall requirement specification and is thus fairly easily ascertained. At that level, software performance can be determined just as readily as hardware performance if not more so. While it is true that some software bugs may exist for years before detection, these are minor deviations to desired performance. Once debugged and operational, programs simply do not fail; so once properly demonstrated, they perform without error forever.

The need for de-personalization of tasks in scientific and engineering endeavors is self-evident. Programming should also be approached just as objectively as hardware designs and specifications provide objectivity, if nothing more.

Standardization in hardware is a recognized necessity for many reasons, not the least of which is the interplay of subunits. Software tasks involving many individuals have similar problems of interplay. The fact that this interplay is less evident makes it more of a problem. Standardization would go a long way toward facilitating this interplay. Specifications, with their use of applicable document references, provide a convenient means of imposing basic standards on the overall requirements of a peculiar task.

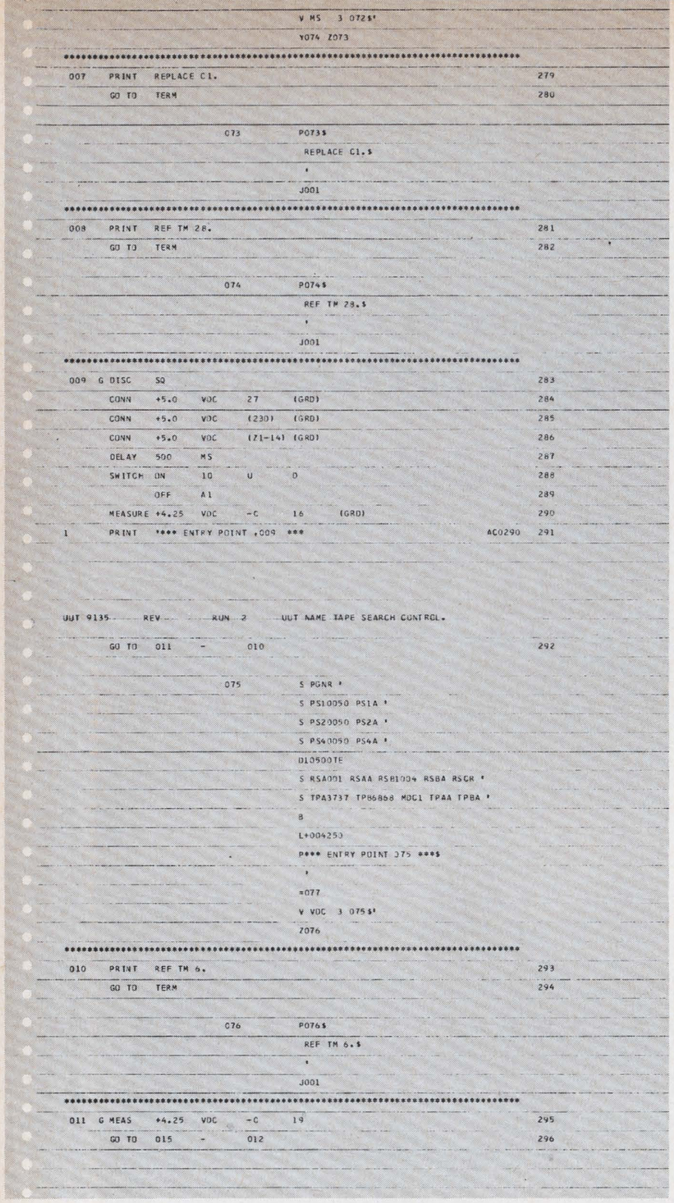
Any design well controlled by documentation provides a sound basis for related development efforts to come. To appreciate the design considerations of a program, a good deal more than a finished program is required. With specifications covering each facet of software development, later projects can see exactly the logic, objectives and accomplishments of a past software effort.

In any effort, a manager's span of control is effectively increased by relegating definable direction to written procedures and reserving his personal attention to problem areas as they develop. Most of the software details are directly related to the ultimate task requirement and thus may be documented at the start of a project. Specifications provide such a vehicle of documented control.

Specifications, as definitive documents, provide historical information on past projects. This was discussed as it pertains to follow-on work. Specifications also provide a tool for orientation and training of personnel. This is an equally valuable tool for software as it is in hardware development.

The final consideration for specification control deals with design changes. This has been deferred to last not because it is least in importance but because it is perhaps most worthy of consideration and comment. Nothing can be more detrimental to schedules, costs and even the success of a task than uncontrolled changes. This has been recognized in hardware design efforts. Formal and often elaborate systems for design changes exist at every phase of system design. It becomes increasingly difficult and expensive to make changes the further along the process a product has progressed. Yet, no matter how undesirable, changes are inevitable. They must be contended with but they cannot be allowed to get out of control. Hence, in software, as in hardware designs, only mandatory changes should be permitted and budgets must be realigned periodically to adjust to work load effects of approved changes.

Somehow widespread miscomprehension exists as to the effects of changes on software efforts. Because programs are essentially "paper designs" there is a tendency to feel that nothing is firm. Perhaps it is simpler to change a few coded



```
                              V MS   3 072$'
                              Y074 Z073
*************************************************************************
007    PRINT   REPLACE C1.                                          279
       GO TO   TERM                                                 280

                    073       P073$
                               REPLACE C1.$
                               '
                              J001
*************************************************************************
008    PRINT   REF TM 28.                                           281
       GO TO   TERM                                                 282

                    074       P074$
                               REF TM 28.$
                               '
                              J001
*************************************************************************
009  G DISC    SQ                                                   283
       CONN    +5.0    VDC     27      (GRD)                        284
       CONN    +5.0    VDC     (230)   (GRD)                        285
       CONN    +5.0    VDC     (Z1-14) (GRD)                        286
       DELAY   500     MS                                           287
       SWITCH  ON      10      U       D                            288
               OFF     A1                                           289
       MEASURE +4.25   VDC     -C      16      (GRD)                290
1      PRINT   '*** ENTRY POINT ,009  ***                   AC0290  291



UUT 9135     REV        RUN  2     UUT NAME TAPE SEARCH CONTROL.
       GO TO   011     -       010                                  292

                    075       S PGNR '
                              S PS10050 PS1A '
                              S PS20050 PS2A '
                              S PS40050 PS4A '
                              D10500TE
                              S RSA001 RSAA RSB1004 RSBA RSCR '
                              S TPA3737 TPB6868 MDC1 TPAA TPBA '
                              B
                              L+004250
                              P*** ENTRY POINT 075 ***$
                               '
                              =077
                              V VDC  3 075$'
                              Z076
*************************************************************************
010    PRINT   REF TM 6.                                            293
       GO TO   TERM                                                 294

                    076       P076$
                               REF TM 6.$
                               '
                              J001
*************************************************************************
011  G MEAS    +4.25   VDC     -C      19                           295
       GO TO   015     -       012                                  296
```

A typical sheet from a computer listing of a test program has been prepared for the Land Combat Support System. Source language (left side of sheet) is that of the RCA Universal Test Equipment Compiler and object language (right side of sheet) is machine language for the RCA Land Combat Support System (LCSS).

statements than it is to replace elements in a unit of hardware. But if the program modified costs ten thousand dollars of labor to develop, a five percent change can reasonably cost 500 dollars. Changed programs represent destroyed intelligence which in today's professional labor market is far from cheap. The effect of change on software can be just as costly as hardware changes even though physical evidence of disruption is missing. Stringent methods of change control must be applied to software efforts if schedules and budgets are to be maintained. With specification control of software tasks, changes must be subject to formal control. Availability of this control alone would repay all of the effort expended in setting up a family of specifications to direct and control a complex programming effort.

In summary, this paper has attempted to show the similarity between hardware and software design tasks. This leads to the conclusion that specification control, the method proven effective in hardware management, applies also to software tasks. In fact, in many ways this means of control is more valuable to software control. The existing problems of software development discussed herein are not intended to be critical of programmers. On the contrary, it is somewhat amazing that many programmers in industry are able to achieve goals at all when all of the inherent disadvantages of dealing with elusive software are considered. If anyone is to be chided at all, it is the systems manager who does not give sufficient attention to software at the outset of a system design specification task. The importance of software in computerized systems is self evident. It is time software management methods reflected their due measure of importance throughout each phase of software development. ■

REFERENCES

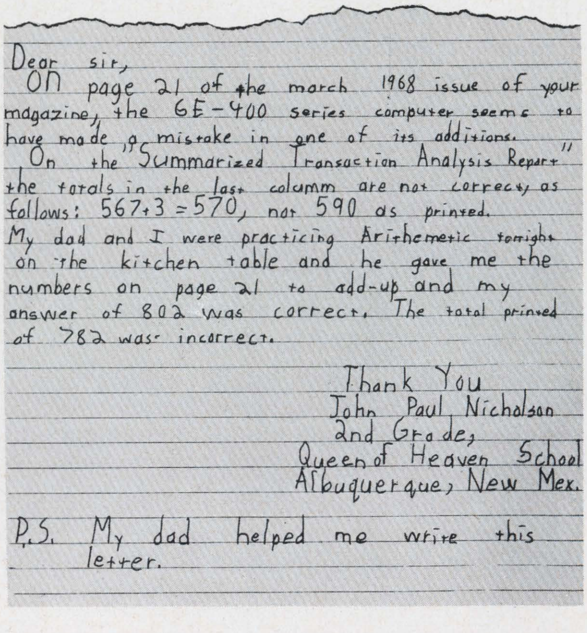
(1) B. L. Ryle, *"An Engineering Approach to Configuration Management"*, Proceedings of the 1966 Automatic Support System Symposium, St. Louis, Missouri, Section of IEEE.

(2) Project SETE, N.Y.U., *"Problems and Pitfalls in Automatic Test Computer Programming"*, May 1965.

(3) Bellcomm, Inc., *"Management Procedures In Computer Programming for Apollo"*, Interim Report for NASA, TR–64–222–1, 30 November 1964.

EVEN A SECOND GRADER KNOWS THAT!

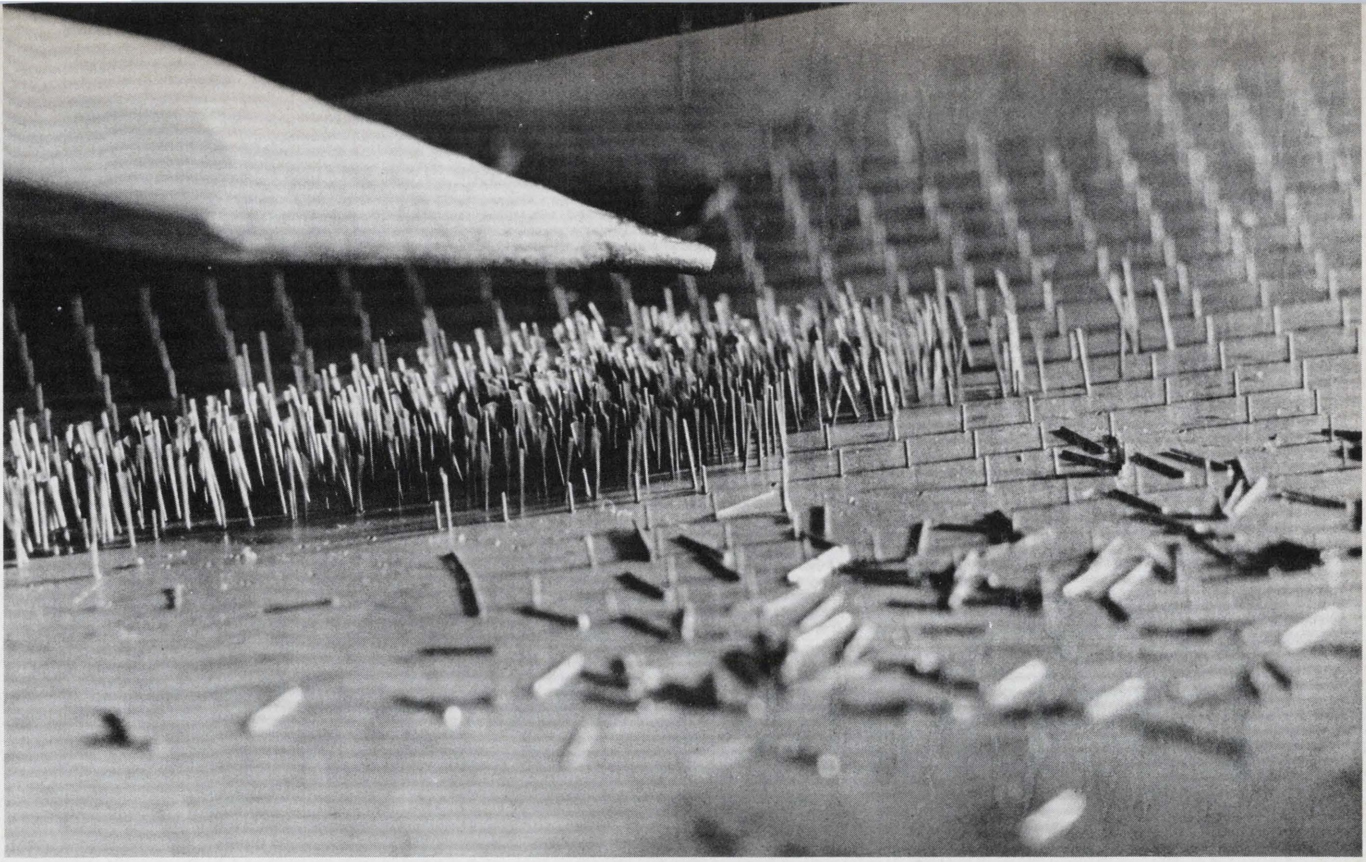
Did you notice something wrong with the printout, "Summarized Transaction Analysis Report" on page 21 of March SOFTWARE AGE? John Paul Nicholson did. And he is in 2nd Grade, Queen of Heaven School, Albuquerque, New Mexico.



Dear sir,
On page 21 of the march 1968 issue of your magazine, the GE-400 series computer seems to have made a mistake in one of its additions.
On the "Summarized Transaction Analysis Report" the totals in the last column are not correct, as follows: 567+3 = 570, not 590 as printed.
My dad and I were practicing Arithemetic tonight on the kitchen table and he gave me the numbers on page 21 to add-up and my answer of 802 was correct. The total printed of 782 was incorrect.

Thank You
John Paul Nicholson
2nd Grade,
Queen of Heaven School
Albuquerque, New Mex.

P.S. My dad helped me write this letter.



These tiny metal rods are the heart of a new computer memory. The rods, only a tenth of an inch long, replace conventional magnetic cores. They literally "dance" into centers of minute solenoids when placed in a pulsating magnetic field.

SOFTWARE of the CENTURY

William P. Keating
Director of Basic Software
The National Cash Register Co.

■ The family idea dominated the software development program of the Century Series of Computers.

User growth and usage patterns dictated the sensibility of a plan for complete upward compatibility of both hardware and software through an entire family of systems.

Three goals were set:

1. The total system marketed must be customer oriented. It must be designed to solve the real world, functional problems of prospective user organizations. Customers must be able to install and program the Century Series with a minimum of effort and expense.
2. The entire effort must be on time. The hardware and software must both be available to the initial user of the Century Series.
3. Modularity, to the highest degree possible, was mandatory for both hardware and software.

To accomplish these objectives, NCR started with a study designed specifically to isolate and define the functional requirements which would be placed on software by a user's object program at the time of execution. Once we were sure that we were providing those requirements necessary to simplify the programming burden, attention then turned to optimizing the space, time, and cost variables associated with including these functions within our system. Our design considerations point out several categories of software, each having their own implementation requirements. These categories are:

1. General software facilities, which are program dependent, continuously required and frequently called upon. Involved here are such facilities as data and file referencing. To meet this need, a series of variably generated sub-routines was incorporated in the system software. These generator routines determine what type of peripheral, file device, and internal logic is required, and the necessary commands are generated to fulfill the program at execution time. Such tailoring allows considerable savings both in memory and running time requirements of user systems, when compared with fixed sub-routine or large executive designs.

2. General facilities, which must be available at all times, which are used frequently, but are not program dependent. The executive includes capabilities such as interrupt handling, simultaneity control for all

peripherals, and software overlay calling routines.

3. General capabilities required throughout the running of the program but used only occasionally, such as peripheral unit initiations, file closing routines, error procedures, and logging. Communication to this logic is handled by the resident executive mentioned before. The executive will select and read into memory these infrequently used sub-routines, which provide full operating control of peripherals.

This concept of using supplementary disc storage provides the vital link in assuring upward compatibility between the smallest and largest members of the Century Series.

4. General procedures, which are required before and after a program run, to assure successful initiation, completion, and work flow. This is satisfied within the new software package by a job scheduling monitor system. This controls all program to program flow. It is responsible for program loading and system setup. The sequence of programs to be performed can be maintained on disc and a high degree of selection flexibility controlled by dates or data is possible within the system.

New NCR Century Series computers are based on design advances in ultra-high-speed thin-film memories, monolithic integrated circuitry and automated production techniques.

The guaranteed use of the disc on all family members also allows the creation of a daily magnetic log for all operations, and performs other necessary housekeeping functions.

This log records the names of all user programs which have been run, all errors detected, and all system malfunctions. Among other things, it serves as an invaluable diagnostic tool for maintenance personnel.

The composite of the above features represent an advanced approach to operational problems. These capabilities help measurably in achieving the basic goal of making the Century Series easier for customers to use and operate.

After providing for the execution of the object user program, the next phase in the new software philosophy called for basic needs of user program compilation and testing. This cycle was broken into three segments:

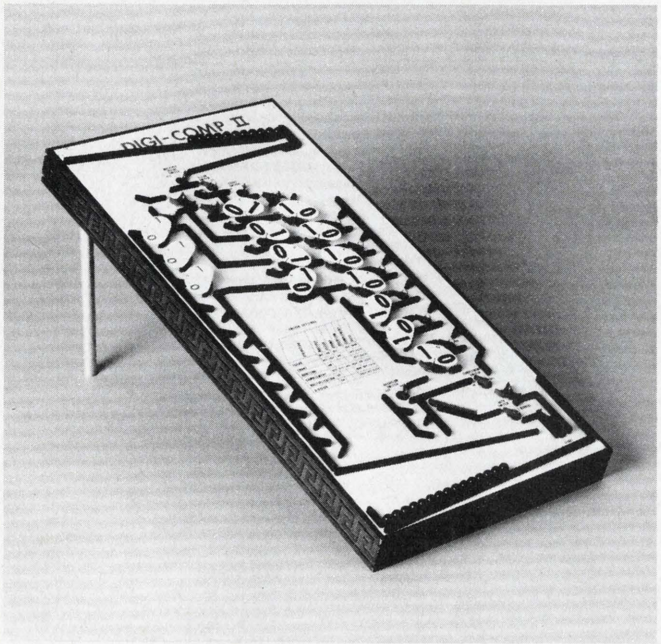
1. Functions to be performed prior to compilation.
2. Actual compilation.
3. Post-compilation requirements.

These requirements are language independent in that they must be accounted for no matter what programming language is used. The Century system programmer will be provided a series of software routines and compilers to satisfy the requirements cited above.

Now . . .

DIGICOMP II - the "second generation" of the famous DigiComp,

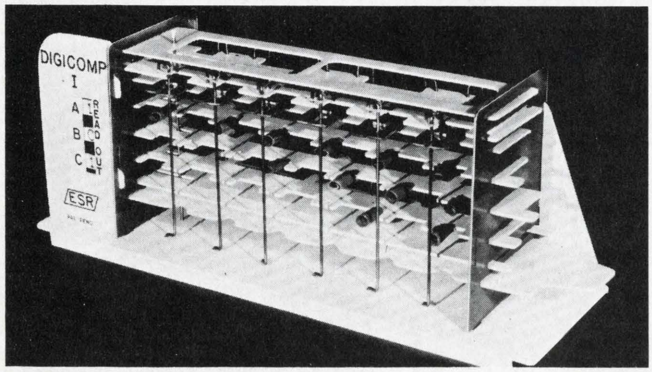
enormously more "sophisticated" as additional "switches" and "core positions" permit many more decisions and "branching" possibilities.



Like its predecessor the DigiComp-I (see below) it is entirely mechanical in operation but affords the functions common to electronic computers in a form far more comprehensible. The best way ever to learn (or to teach) the principles of modern data processing.

$16.95 Postpaid

DIGICOMP-I COMPUTER

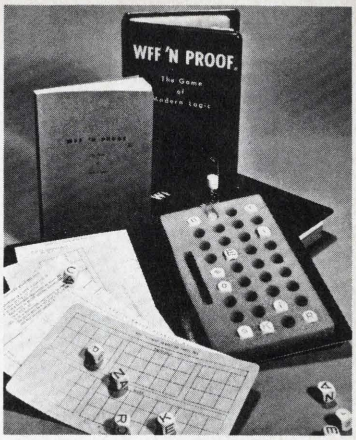


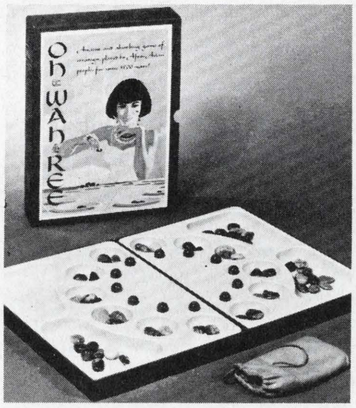
Ingenious mechanical equivalent of an electronic digital computer. Can be "programmed" to perform, individually, every operation of a digital computer. Best thing ever for teaching binary math, how a computer operates, how and why problems must be broken down for machine solution. In kit form—1 hr. to assemble. $5.98 Postpaid

WFF 'N PROOF

Developed in the course of the ALL Project (Accelerared Learning of Logic) Wff'nProof incorporates 21 games of progressive difficulty; has been used to teach propositional calculus to elementary school children but challenges the intellect of professional logicians. Complete with 224-page manual which is an introduction to the world of modern logic; filled with valuable tips to clear thinking.

$6.25 Postpaid





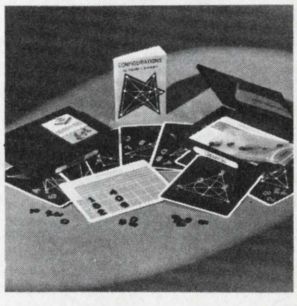
OH WAH REE

The ancient game of mathematical strategy which has intrigued millions for more than 30 centuries. A game at once so simple in its basics that a pre-schooler can enjoy it yet so complex in its possibilities that its fascination multiplies with the mathematical grasp of the player.

$7.95 Postpaid

CONFIGURATIONS

Based on concepts from the geometry of incidence, CONFIGERATIONS is a series of intriguing mathematical and geometric puzzles that will challenge and delight those who enjoy careful reasoning. It is authored by Professor Harold L. Dorwart, Chairman of the Mathematics Department, Trinity College, Hartford.



$4.95 Postpaid

TWIXT

Another famous "bookshelf" game, package matches that of Oh-Wah-Ree. A stimulating game with a fascinating chess-like strategy of move and countermove. Two or four players.

$7.95 Postpaid

SCIENCE/SYSTEMS P. O. Box 1176 Evanston, Ill. 60204 **SA-5**

Please send me at the address below:

. . . .

. . . .

. . . .

. . . .

Save $1 by ordering any 3 items or $2 on any order for 4 or more of the above

(Games are shipped from separate warehouses)

(name)

(address)

(city) (state) (zip code)

Illinois residents add 5% OES

Prior to compilation, all programs must be put into proper form, format, and sequence for processing. The new NCR software handles these requirements with a package known as SPUR (Source Program Utility Routines). These routines handle all of the sequencing and correction of source programs. Maintenance of source program libraries and the ability to include commonly used routines or data definitions from other programs substantially reduce the raw input requirements of the programming. This program handles all Century languages.

The language and subsequent compiler selection of the Century Series began with a definitive study aimed at classifying the types of users we were most likely to encounter. This evaluation took into account that a whole new family of computers was involved and that the prospective users would range for the neophyte up through the experienced sophisticates. Based on these initially established requirements, the NCR software group elected to provide the following programming languages to users of the new computer systems.

For scientific programmers, two versions of FORTRAN are available. This approach makes scientific programming feasible at every level of the Century Series. For minimal systems (16K memory, 2 disc packs) FORTRAN II is provided. On larger systems, FORTRAN IV serves as the scientific language.

Recognizing and providing for the needs of business programmers who require a standardized language created the need for a COBOL Compiler. Different levels of COBOL are offered with the Century Series. A basic level provides a compiler usable on the minimum 16K configuration. Customers installing medium and larger systems have COBOL Compilers which are scaled to the capabilities of their equipment and conform to the latest USASI draft standards.

An additional effort has been made to meet the requirements of both the neophyte who wishes to program utilizing higher-level, easier to use, generator and macro techniques or the highly sophisticated user or software writer who needs an assembly level language which makes it possible for him to code programs in one-for-one correspondence with machine functions. In supporting the Century family, we have met these requirements. Specifically, through the development of a single, multilevel language, NEAT/3. NEAT/3 provides a vocabulary ranging form instructions similar to basic COBOL on up to powerful reoccuring data processing functions.

As the basis for development of this segment of the NEAT/3 Compiler, an analysis was made of large numbers of actual user programs to classify the functional tasks required of informational system compilers. These, it was found, fall into two broad categories:

1. Basic tasks. Typical is ADD A to B.
2. Tasks incorporating many parameters and wide scope. This type of command corresponds in capability to Report Writing. NEAT/3 includes nine separate functions of this degree of complexity in its initial version. Furthermore, the vocabulary is open ended, providing for continuing and expansion. Examples of functions available include file updating, report writing, and several others. You have seen in the slides shown earlier the type of questionnaire the programmer answers in requesting these tasks.

Although necessarily brief, it is hoped that the description of the compilers offered with the Century Series will answer the needs of the already initiated scientific and business users with the COBOL and FORTRAN Compilers and also provides a direct appeal to prospective users looking for an easy and/or fast way to write programs through the use of NEAT/3.

To fulfill their responsibility to users, the programming language facilities provided with the Century offer the capacity to debug, as well as to code, symbolically. This symbolic debugging capability is offered with both NEAT/3 and COBOL.

Furthermore, to be of real value, it is felt that the option of symbolic debugging should be available to the programmer at any time. Under the new NCR programming language concept, for example, the programmer is not restricted to stipulating his chosen debugging technique at the time of coding. Rather, the ability to debug symbolically is decided after compilation—without either recompilation or changes in the size of the object program. Specifically, the following debugging options and features are provided:

1. Symbolic snapshots of data fields can be called for. These fields are named through symbolic references entered by the programmer during coding. They are printed according to the programmer's own data definitions.

2. The programmer, at any time, can call for a trace of the object program. In such printouts, identifying page and line numbers of branch commands are included, as contained in the original, source coding.

3. Conditional snapshots or tracings may be requested based on comparisons of specific fields, named symbolically.

Any of these options, or combinations of these routines, can be requested to solve the problems which develop during actual program testing. These features are always available, requiring no special coding or advance provision at the time programs are written.

I will not attempt to discuss our developments which go beyond the minimum Century configurations. It is important to note, however, that the Century software design encompasses the area of real time and multi-programming. In fact, many of the design tradeoffs mentioned earlier are highly influenced by requirements in this area.

In conclusion, it is hoped that this article summarizes the philosophical element and implementation approaches whch have gone into the creation of system software for the Century Series. We must stress the ease and simplicity of working with the software philosophy enunciated here is fully expansible to provide for any operating requirements of modern computer applications. User programs written for the smallest Century can run on any other system in the family. The universality and availability of the NCR software package surpasses the goals established at the beginning of the development. Without question, this combination of hardware and software provides a system unequaled in the industry today. ■

new products

Atlantic Software Inc., a firm which specializes in marketing computer software, announces the **Data I program.** Data I is a special-purpose System 360/Spectra 70 utility designed to reduce file maintenance programming costs, and to eliminate one-time programs for data and file manipulation.

Data I is said to update tape or disk files from cards, tapes or disks in any format. It can also generate tape, disk or card files. Output formats are completely variable. Program requires 64K core, will operate under DOS and OS with any combination of disk and tape I/O.

Experience with Data I already totals several man/years, with programming time savings running as high as 95%, according to Atlantic Software.

For more information circle No. 80 on the Reader Service Card

* * *

A proprietary software package, **ALTRAN,** has been recently introduced by Applications Software Inc., San Pedro, California. ALTRAN (Assembly Language **TRAN**slator), translates SDS 900 Series SYMBOL or META-SYMBOL source language programs to SDS Sigma 5/7 META-SYMBOL. ALTRAN runs on either SDS Sigma 5/7 or SDS 9-Series computers. This latter operational characteristic is especially significant, since it enables a user to begin program conversion long before the new machine is installed.

ALTRAN is said to yield 90% translation to SDS Sigma META-SYMBOL. A translated program is more than twice as fast as the old program. This is in contrast to simulated programs, which seriously degrade the performance of the SDS Sigma 5/7 computer, since the simulated programs are much slower than the original programs executed on SDS 900 Series computers. ALTRAN provides a user with many input/output options through the availability of in excess of 16 translation parameters.

For more information, circle No. 79 on the Reader Service Card

* * *

A **program documentation binder** made of Wilson Jones "Super-Flex" has been added to its line of EDP equipment specifically for use by programmers.

The Program Documentation Binder (Stock No. 03–1411) is one-piece construction, suited for systems in which the source documents, parameter cards and the program printout are of such size that they can be housed in a single unit. It is designed for maximum compactness and convenience. The printout side can be loaded from top or bottom of the sheet body. The document side includes a vinyl envelope for the program's control cards and an index with printed insertable EDP titles. Available in 6 colors for coding programs.

For more information, circle No. 78 on the Reader Service Card

* * *

New **electronic thermometer and switchbox** by Atkins Technical Incorporated, monitors up to 20 points, displaying temperatures on 4½″ taut-band meter and providing 0–50 mv signal to optional recorder. The thermistor probe switchbox can be programmed for dwell time on each point of 10 seconds to 5 minutes, and for sweep through 3, 7, 12, or the full 20 probes. The temperature indicator and the switchbox can be packaged together in a single cowling for portability as one device, or separately as shown.

For more information, circle No. 77 on the Reader Service Card

* * *

Potter Instrument Company has introduced a new, low-cost, **single-capstan digital magnetic tape transport system** for use with small and medium scale computers.

The new transport system—Model SC-1030, provides bidirectional tape speeds to 37.5 ips at 800 bpi, NRZI, as well as 1600 bpi phase modulated recording. The unit is fully 7- or 9-channel (IBM 729, 7330 and 360/2401, or ASCII) compatible.

Single-capstan design is used to drive the tape. A precision edge guidance system provides reliable tape control. Tape loading is semi-automatic in less than 15 seconds. In operation, the oxide side of the tape touches no stationary surface except the read/write head and tape cleaner, greatly increasing tape life and data reliability.

Other specifications: Rewind: Less than 4 minutes (2400 foot reel). Start time: 8ms; Start distance: 0.165 ± .025″; Stop time: 7ms; Stop distance: 0.110 ± .020″; Speed tolerance: ±5%; Dynamic skew: 5 usec.

For more information, circle No. 76 on the Reader Service Card

* * *

A new high-speed **information retrieval software system** has been announced by Computer Corporation of America. Called the Model 101, the system is designed for use with IBM System/360 hardware using disk packs as the data storage medium.

The Model 101 has the capability of storing large as well as small files—over 100,000 individual records may be accommodated. Information may be retrieved simultaneously from several independent files through the systems cross-referencing capability.

Data input and output is handled by COBOL-compatible magnetic tape. Alternately, input and output may be handled through punched cards and high speed line printer.

Hardware requirements for operation of the Model 101 information retrieval system include: IBM System/360, Model 30 or larger with 65K bytes of core storage (minimum); 1 to 4 Model 2311 disk drives for data storage; DOS operating system; three magnetic tape drives, line printer, and card reader.

For more information, circle No. 75 on the Reader Service Card

* * *

EPSCO, Incorporated, has introduced the DATRAC IV, **a high-speed analog-to-digital converter plug-in-module** for data processing applications.

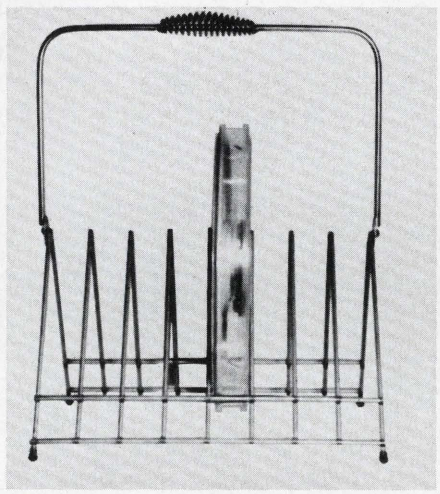
DATRAC IV's conversion speed of one microsecond per bit allows up to 100,000 conversions per second. At top speed, the unit's overall accuracy is .05 per cent (plus-or-minus one-half the least significant bit).

Special features of the DATRAC IV include exclusive use of DTL integrated circuits for the input buffer amplifier, comparator, reference supply and all logic functions. Precision attenuator networks and switches are in discrete form.

Operating characteristics of the DATRAC IV include 10 or 11 binary bit resolution, parallel output data for micro-logic interfacing, negative 2's complement code and a 100,000 cycle word rate (KHz).

For more information, circle No. 74 on the Reader Service Card

* * *



Dawn Products Co. is offering a new product to hold computer reels. With this **Reel Caddy** one can carry from 6 to 12 reels at a time. The Reel Caddies are of heavy gauge wire, nickel plated with rubber feet to protect table or desk top.

For more information, circle No. 73 on the Reader Service Card

* * *

PACFAC—**a disk pack analysis and control program for the IBM/360**—was announced by Software Resources Corporation. PACFAC interrogates the volume table of contents of any IBM disk pack to determine what files are present, their activity status, their organization, size and location as well as the size and location of available free space. It also provides a permanent history of inoperative tracks.

PACFAC permits complete machine analysis of file space in place of laborious manual methods. By providing improved documentation and control of file storage, PACFAC can reduce disk pack inventory costs in many installations.

PACFAC output lists all files alphabetically and lists unused space sequentially by cylinder and track. It can also sequentially list an entire disk pack or data cell, both files and free space.

For more information, circle No. 72 on the Reader Service Card

* * *

The GECENT III Postprocessor System, a **software support program written for General Electric Mark Century** numerical control systems used in conjunction with the APT III computer system, is now available to users of Mark Century NC systems.

Taking advantage of the flexible overlay capabilities in third generation computers, and by varying the number of overlays, the new GECENT III Postprocessor can be made to fit large and small computers, multiple programming computers, or eventually a time-sharing computer. Initial tests of the new program indicate computer processing times of GECENT III to be from 1/6th to 1/12th as long as the GECENT II program which it replaces.

The structure of GECENT III is based upon the ability to sectionalize those subroutines which apply to a particular type of machine tool (i.e. positioning, lathe, mill, etc.).

For more information, circle No. 71 on the Reader Service Card

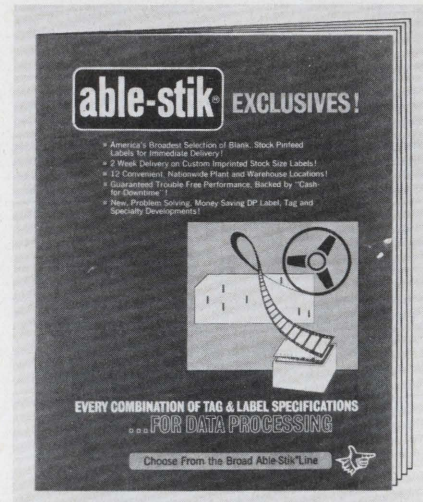
* * *

The Scanbe Model 600 **vertical equipment drawer** has been designed to provide modular storage for any desired number of card file units.

The Model 600 provides complete access to circuit cards, connectors and wiring during assembly and later servicing; allows extra space between front and rear panels and the large card storage section; features a built-in flexibility of the card files, card-mounting sets and vertical drawer system that allows portions of the drawer to be reserved for power supplies or auxiliary units; and includes frame dimensions which can be modified to fit any desired depth of file.

For more information, circle No. 70 on the Reader Service Card

* * *



Data processing managers can gain time-saving ideas from the new 16-page **data processing labels, tags and specialties catalog** just issued by the Allen Hollander Co., Inc.

The line, consisting of many stock and custom imprinted variations, offers a wide range of sizes, shapes, substances and adhesives that make it possible to retrieve information from data processing media, print it out at high speed and then use it for such functions as addressing, identifying, numbering, recording, posting, pricing, grading, coding, inventory, mailing, reporting and accounting.

For more information, circle No. 69 on the Reader Service Card

* * *

Now available from Keltec Industries, a Division of Aiken Industries, Inc., is the Model CP-2811 **Data Conversion System,** a compact unit (5¼ x 19 x 21½ inches)

SUBSCRIBER SERVICE

Mail to: Software Age, 1020 Church St., Evanston, Ill. 60201

Please fill out the blanks below whether you are changing your address or requesting a new subscription.

☐ Please change my address ☐ Please enter my subscription

ATTACH LABEL HERE

FOR CHANGE OF ADDRESS

If you're moving please let us know 5 weeks before changing your address. If you do not have your label print your **old** address below.

Old home address ______________________

City, State ______________________ (Zip)

Name ______________________

New home address ______________________

City, State ______________________ (Zip code)

My Specialty ______________________

prime experience in what industry? ______________________

☐ Technical Degree

☐ Non-Technical Degree

☐ No Degree

year born ______

☐ I have analog/hybrid exp.

which accepts analog information from up to 20 different sources and generates digital data in IBM format at minimum cost and with maximum simplicity. Design features include a 20-channel solid-state scanner (multiplexer), a 12-bit analog-to-digital converter, and a scan rate of 2000 channels per second. Outputs and controls are compatible with Digi-Data Incremental Tape Recorders and most other magnetic and punched-paper-tape recorders.

For more information, circle No. 68 on the Reader Service Card

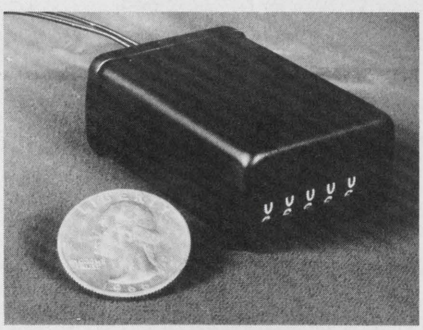
* * *

A proprietary **software package** introduced by California Computer Products, Inc. eliminates the need for special software in the conversion of CRT images to plotted hard copies.

Designed for use with the IBM 2250 display unit and any CalComp plotter, the new program automatically converts a 2250 CRT image, or selected portion, into the correct format for plotter reproduction, either on-line or off-line.

For more information, circle No. 67 on the Reader Service Card

* * *



A new, **sub-miniature digital readout elapsed time indicator** has been developed by the Industrial Timing Division of Sessions, Forestville, Connecticut.

The compact unit, which reads out in tenths to 10,000, is available immediately in a 120-volt, 60-cycle package with a variety of mountings available to meet customer specifications. The device may be equipped at an optional cost with a device that will enable the meter to be reset. The maintenance timer incorporates ⅛" high characters in easy-to-read boldface figures. Standard leads are stripped and induction soldered; length and color per customer specifications.

The digital ETI supplies exact computations.

For more information, circle No. 66 on the Reader Service Card

* * *

Computer Lab workbook presents a wide array of **logic gates,** gate combinations, flip-flaps, etc. for student use in learning how a computer works—from a single gate operation, to wiring and controlling the operation of subsystems. Computer Lab is manufactured by Digital Equipment Corporation, and uses integrated circuits similar to those found in the company's latest computers.

For more information, circle No. 65 on the Reader Service Card

* * *

Featuring a double safe construction with an outer safe and an inner repository, the new **data-bank fire vault** by Wright Line, a division of Barry Wright Corporation, is designed to provide maximum protection for vital EDP records stored on either tape or disk packs.

The new vault carries the Underwriters Laboratories, 150° two hour label for storage of EDP media. It will hold interior temperature and humidity well below the 150° F. 85% R.H. limit at which read out errors and losses can occur even when the vault is exposed to a raging fire of up to two hours duration.

For more information, circle No. 64 on the Reader Service Card

* * *

Digitronics Corporation has extended its line of paper tape equipment to cover low-speed, asynchronous reading and punching applications. The company's new **low-speed reader,** Model 2060, reads any EIA standard 11/16 (5 channel) to 1-inch (8 channel) punched tape and operates at speeds to 60 characters per second. A simple drive, utilizing solenoids which do not require lubrication, is used to move tape. For high reliability, bifurcated contacts are employed to perform the read function.

The new Model 1560 **Paper Tape Punch** operates asynchronously at rates to 60 characters per second and records data in five through eight-level code. Featuring in-line loading of tape, solenoids which require no lubrication and optional tape supply reel and take-up winder, the Model 1560 is available in four packages styles to meet most packaging requirements.

For more information, circle No. 62 on the Reader Service Card

* * *

A new full color brochure describing Rol-a-chart **visual control boards** features a full-scale sample chart section and a guide to visual control, according to the Wm. A. Steward Company.

To assist the planner, the sample chart section is large enough to actually layout a scheduling system. Typical examples are given showing a visual control system charted with marking pencils and the same visual control charted with magnets.

For more information, circle No. 61 on the Reader Service Card

* * *

Mathematical Engineering Associates, Inc. of Dallas has released **MANAGE,** a program which enables a computer installation to maintain its source programs on tapes or disks.

MANAGE is written in COBOL, and can be used on any tape computer which has COBOL capability. MANAGE maintains programs written in FORTRAN, COBOL, AUTOCODER, ALC and other machine languages.

Using MANAGE, a computer installation enters and carries its source language programs on a program master tape or disk. The installation's programmers then use their computer to add to, delete from, and otherwise maintain or resequence their programs. The master tape or disk in the MANAGE system is also used for computer input as well as to list any program or its modifications.

A MANAGE produced program listing is a program log that becomes part of program documentation. This MANAGE produced log shows exact modifications in each update run. The MANAGE log enables a programmer or manager to know exactly what statements have been altered in, added to, or deleted from a program, and the date and sequence of each such action.

For more information, circle No. 60 on the Reader Service Card

* * *

software age

THIS INQUIRY IS IN DIRECT RESPONSE TO YOUR ADVERTISEMENT IN SOFTWARE AGE MAGAZINE

CONFIDENTIAL INQUIRY

Your original copy of this form will be retained at the offices of SOFTWARE AGE and will be used for no other purpose than to notify the specific firms which you have checked (on the reverse side) of your interest.

TYPE OR PRINT CLEARLY FOR PHOTO REPRODUCTION

JOB DESIRED: ______

List computer hardware knowledge (names of systems, tape, disk, terminals, etc.): ______

Programming specialties and years of experience (commercial, scientific, theoretical, experimental, analog, etc.): ______

Systems programming on which you have had development experience (compilers, assemblers, executives, monitors, O.S., etc. Indicate for what computer): ______

Programming languages used and extent of experience (COBOL, FORTRAN, etc.): ______

Applications programmed (aerospace, banking, insurance, math subroutines, compilers, etc.): ______

Systems analysis experience (card design, flow charting, operation analysis, etc.): ______

EDP management experience (include years and number of people reporting to you): ______

SALARY: ______ (current) ______ (desired) **DATE OF AVAILABILITY:** ______

EDUCATION: Indicate major as well as degree unless self-explanatory.

Degrees			
Years			
Schools			

EMPLOYMENT: Indicate present employment and previous jobs below.

Employer			
City			
Years	to	to	to
Title or Function			

Name ______ Age ______

Home Address ______ Home Phone ______

______ (city) (state) (ZIP code) U.S. Citizen? ______

Security Clearance ______ Location Preference ______

Marital Status ______

BE SURE YOU HAVE CHECKED ON REVERSE SIDE THE COMPANIES YOU WANT TO SEE THIS INQUIRY. PUT FORM IN STAMPED ENVELOPE AND MAIL TO:

software age
MAGAZINE
1020 CHURCH ST., EVANSTON, ILL. 60201

AFTER JULY 15, THIS FORM WILL NOT BE PROCESSED
WRITE ADVERTISERS DIRECTLY OR WRITE US FOR CURRENT S/A ISSUE

check your interests here

Fill in the confidential inquiry form on the other side of this sheet. *(Be sure to use this form. We cannot process multi-page resumes; and this form provides all the information advertisers require to screen applicants. If further information is desired, you will hear from the advertiser direct.)* Then, circle below the numbers of those companies to which you want copies of your form sent. Mail to SOFTWARE AGE, 1020 Church Street, Evanston, Illinois 60201.

CIRCLE BELOW:	Page
1. AC Electronics Div. of GMC	2nd Cover
2. Allis–Chalmers	20
3. The Boeing Co., Commercial Div.	5
4. California Blue Shield	4th Cover
5. EMR, Computer Div.	42
6. General Electric Co., Information Service Dept.	33
7. Hamilton Standard Div. of UAC	23
8. Honeywell, Inc., Electronic Data Processing	3
9. IBM Corp.	7
10. ITT Federal Electric Corp.	19
11. Kimberly–Clark Corp.	9
12. LTV Aerospace Corp.	4
13. MIT Instrumentation Laboratory	11
14. Montgomery Ward Data Center	44
15. National Cash Register Co.	3rd Cover
16. North American Rockwell, Rocketdyne Div.	14
17. Northeastern Illinois State College	30
18. Parke, Davis & Co.	36
19. Philco–Ford Corp.	43
20. RCA Information Systems Div.	13
21. Sikorsky Aircraft	17
22. Stearns–Roger Corp.	41
23. Univac Data Processing Div.	22

CIRCLE BELOW:	Page
24. Univac Federal Systems Div.	18
25. Xerox Corp.	15

EMPLOYMENT AND SEARCH AGENCIES

26. Callahan Center for Computer Personnel	43
27. Computer Personnel Agency, Inc.	41
28. Electronic Systems Personnel Agency	26
29. Fox–Morris Associates	30
30. Harper Associates	30
31. Input, Inc.	42
32. Knorr Associates, Inc.	43
33. Lawrence Personnel	40
34. Management Scientists, Inc.	29
35. Nation-Wide Employment Agencies	28
36. Professional Career Centers	28
37. RSVP Services	40
38. Sheridan Associates	30
39. Source EDP	25

PRODUCTS AND SERVICES

Cambridge Communications Corp.	41
Data-Link Corp.	6
McGraw–Hill Book Co.	25
Science/Systems	10, 20, 27 & 37

☐ **I do not now receive S/A. Please enter my FREE subscription.**

....
My Specialty

....
Prime Experience in What Industry

☐ Technical Degree ☐ Non-Technical Degree ☐ No Degree

.... ☐ I Have Analog/Hybrid Experience
Year Born

software age
1020 Church Street
Evanston, Ill., 60201