

GENERAL  ELECTRIC

DIAL COMM 8*433 2569 DATE 1970 July 28 MAIL ZONE M2



DEPT. •

ADDRESS •

COPIES •

SUBJECT •

TO: J. B. Stroup
 T. A. Vanderslice

FROM: R. W. Bemer

RECEIVED
JUL 29 1970
Dr. L. A. Vanderslice

Attached for approval is "Origins of Report Generators for Computers". This is to serve as introductory material in the fifth G. P. Lecht book for McGraw Hill. I wrote the same sort of thing for his other books (PL/I, ALGOL, etc.).

R. Bemer
↑
po

*Looks great
enjoyed reading it
TMB
8/7/70*



ORIGINS OF REPORT GENERATORS FOR COMPUTERS

R. W. Bemer, General Electric Company



A majority of today's programmers are unaware of the history of computation and data processing prior to the advent of stored program computers, even though the tools they use sometimes owe their form to that era. Mechanical data processing existed for several decades, using devices such as tabulators, collators, sorters and calculating punches. All of these utilized the punched card as the main data medium. Their operations were determined by control panels, commonly called plugboards. Into the holes of these panels one plugged wires (usually color-coded to varying length for mechanical reasons) which, when the panel was inserted in the machine, provided paths of electrical connectivity that completed the determination of the manner in which the machine would operate.

Some of these holes represented the columns of the punched card, at one or more reading or punching stations. Others represented the column on a printer (or tabulator, when more than just printing was done). These pairs were connected to link a particular card column with a particular printing position. Other holes represented timing impulses, filters to separate the rows of the card, different reading stations, and alternate exits of two-way switches which could be interconnected to represent logical decision conditions such as AND and OR.

Because machine runs were expensive, these control panels were often wired to do very complex processes, or multiple processes, on a single run. Despite the fact that the panels were also expensive, most installations kept a large inventory of permanently-wired panels for the various types of runs. This was

because it was very tedious and mistake-prone to remove the old pattern and rewire for a new run, according to a diagram of interconnections.

One of the several advantages of the stored program computer was that the operational pattern of the machine could be controlled by a program instead of a panel, and this program could be read into the computer on a deck of punched cards to set up the job, instead of inserting a control panel. The deck of cards was much cheaper than a control panel, and it was much easier to copy than a panel, which meant that program interchange could begin between users.

Perhaps the programming was a little trickier, and it certainly required retraining, but we got around that by developing languages that another program could understand and convert into machine programs. First there were assembly languages, then macroinstruction facility, and then symbolic languages for procedure statement. The instruction repertoire of the central processor of a computer is almost totally procedural. Thus the procedural languages gained much usage and development.

For some time the concentration on procedure-oriented languages, which say "do this, then do that", caused only a little attention to be paid to problem-oriented languages, which say "This is what I put in, and this is what I have to get out". Possibly the computer manufacturers and users did not realize that a tabular form is as much a structure for a language as is a set of rules for syntax and semantics.

Filling out a form can be the counterpart of writing narrative sentences in a natural language (such as English), and indeed this is a powerful concept. There are many usages where the problem can be expressed more concisely and simpler in a multidimensional form. The Chinese ideogram is an example. Report generators and decision tables are major examples for computer programming. Not only are they easier to use in stating the original conditions, but much easier to change as well. Programs do not stay static, and this is an important consideration.

I recall visiting General Electric at Hanford in 1957 March to discuss IBM's work on Commercial Translator. They were using extensively a Report Generator they had written for the IBM 702. Because their plans called for reworking this to run on the IBM 709, it seemed reasonable that the work should be presented to the SHARE organization of IBM scientific users. The SHARE Data Processing Committee met in San Diego on 1957 October 2. Harrison Tellier of GE gave an extensive presentation on their Report Generator. Two other systems were also presented, the DuPont APG1 (Autocode Report Generator) for the IBM 705 (as modified and extended from their UNIVAC program), and the Boeing Output Generator for the UNIVAC 1103A. After much haranguing it was agreed not to make a distinction between scientific and business data processing in either input or output editing. Working committees were established separately for the 704 and the 709.

In 1961 May I began a survey for the programming language standardization work which was just then beginning. A questionnaire was distributed widely and internationally, so the following list of additional file- and problem-oriented languages may be taken as fairly thorough coverage in the report generator field:

SURGE	IBM 704	Operational 1959 June. Modeled after the GE Hanford Report Generator. The name stands for Sorter, Updater, Report Generator, Etc. SHARE maintained the program via several interested installations; IBM did not.
9PAC	IBM 709	Operational 1959 October. IBM took over maintenance by 1960 February. As nobody could think of a good name, it stands for 709 <u>PACK</u> age. Also a file maintenance, sort, and report generator.
QD SURGE	IBM 709/90	Operational 1960 November. A modification to adapt SURGE to a different machine, QD standing for Quick and Dirty. A regular 709/90 SURGE came later, in 1962.
PROGENY	UNIVAC SS90	Operational 1961 October, primarily a report generator.
GECOM	GE 225	Operational 1961 December. A 9PAC-like report, file maintenance, and sort generator, attached to a COBOL-like language.
FACT	Honeywell 800	Operational 1961 December. The report, file maintenance and sort generators were included in a COBOL-like language.

Beyond this the list needs no expansion, for from this time the Report Writer specifications of COBOL started to move toward de facto standardization, as given in the COBOL 61 Extended Report. Also excluded are report generators in use at local installations during this early period.

From the earliest days, RPG was a misnomer. As long as there was a file to update or maintain, the record or report might as well be printed at the same time. Since the master file and the change file had to be in the same order, the sort has to be associated. Actually the file is ordered, and this is further evidence of the transitional character of the RPG. On a mechanical card sorter, the machine sorted cards by type, and the way the operator picked

them out of the pockets and adjoined them gave them the ordering. However, the original term has persisted.

Thus the characteristic of report generators is not that we made them important, but that they are naturally important from the percentage of data processing work which they can accomplish, particularly on a small computer which cannot accommodate COBOL. Additionally, the parallelism to tab and punched card usage is strong, which makes a very convenient transition mechanism to stored program computers for such users. The pictorial aspects make comprehension easier, and reduce training problems.

61

FOREWORD

Author of the *History of COBOL*

IBM Corp., Armonk, New York

THE PROGRAMMER'S COBOL: A Complete Reference

by

MARJORIE BERK

Senior Consultant, Advanced Computer Techniques Corporation

With a foreword by

ROBERT BEMER, General Electric Company

Inter-ACT Publications

New York • Washington • Phoenix • Boston • Chicago • Rhode Island • Milan

THE PROGRAMMER'S COBOL

Copyright © 1970 by Inter-ACT. All Rights Reserved. Printed in the United States of America. This book, or parts thereof, may not be reproduced in any form without permission of the publishers.

07-004 895-9

FOREWORD

A VIEW OF THE HISTORY OF COBOL

R.W. Bemer, General Electric Company

COBOL is a programming language known to a large number of those involved in data processing. It is less known to the general public, and chemists becoming involved in data processing for the first time are prone to confuse it with Element #27. It is unique among the major programming languages in that factual histories of its inception and development abound. Standard summaries of the history are carried in all official documents and manuals for specific machine implementations, a policy of the original sponsors.

A very complete history and summary of activities is to be found in the American National Standard X3.23. With this, one can track meetings, participating personnel, and technical motivation. Omitted are elements of personality, background, competition, infighting, and significance to the data processing world. Reading this history, and others, one would conclude that there was never any excitement, strategy, or corporate and individual struggles. Not so. These notes attempt to be more illuminating to those now concerned with the use of this language, which is still living.

The very fact that these standard histories are carried along with COBOL documents is the key to understanding the COBOL effort. COBOL is intended to conserve costs and human resources, but any of the proprietary languages of its class could have done that, and very possibly they would all have grown and matured in the same way. IBM's FORTRAN became an industry standard because it was operational in volume before its competitors, and because IBM placed it in the public domain. In the business data processing world, the race was much closer.

In time sequence of development, the three progenitors of COBOL were: FLOW-MATIC (from UNIVAC), Commercial Translator (from IBM, which ran into legal conflicts with the original name COMTRAN), and AIMACO (from the Air Materiel Command in Dayton).

FLOW-MATIC was an outgrowth of the A-series of algebraic and scientific compilers. The concept of the compiler is largely due to Dr. Grace Murray Hopper, in charge of these projects. The new series started as B-0 (B for business, as opposed to A for algebraic). A predecessor BIOR (Business Input Output Rerun) was developed by a different group, becoming somewhat operational in 1955 April. FLOW-MATIC, as B-0 was

rechristened, became operational in 1956 December, for the UNIVAC I and II. It was not what we would call today a commercial grade software product, and it still underwent continual change and improvement, both language and compiler.

The competitive threat potential of FLOW-MATIC did not go unnoticed at IBM, and some research was started in the Fall of 1956 on alternate solutions for a business language. The original approach tended to high level operators and set notation, such as "MERGE FILEA WITH FILEB ON KEY3" and "UPDATE THIS WITH THAT." I began to worry that this approach might take too long to bring to practicality, and asked Roy Goldfinger to develop a language with the more specific procedural capability of FLOW-MATIC, yet which would retain the set principles. Public notice of the Commercial Translator work was given to the SHARE group in 1957 October, and Roy produced formal specifications in 1958 March.

Reading the standard histories, one gets the impression that the early meetings were spontaneous. Actually, Mary Hawes of Burroughs had buttonholed Dr. Saul Gorn of the University of Pennsylvania at the Western Joint Computer Conference in San Francisco on 1959 March 3-5, asking if he didn't think it was time for a common business language. Saul agreed and later that month held a meeting in his office at the Computer Center (UNIVAC I). One attendee was Howard Bromberg, and it changed his whole life, as we will see later. At a second meeting on April 8, various names were suggested for leadership. Grace Hopper suggested Charles Phillips of the Department of Defense. It seemed most reasonable for DOD to sponsor such an effort, which would take energetic leadership and neutrality, together with the stature (and pocketbook) to command the attention of the manufacturers. On May 21, I sent the President of the Association for Computing Machinery the agenda for the May 28 meeting, probably as a needle for the disdain of business languages in the ACM Programming Languages Committee, although they were within their scope to develop.

The Washington meeting of May 28 noted the separate development of three similar languages, and agreed that the example of ALGOL warranted an effort to develop a common business language. A steering (later executive) committee was formed with Phillips as Chairman, Joe Cunningham of the Air Force as Vice Chairman, Gene Albertson of U.S. Steel, Greg Dillon of DuPont, Mel Grosz of ESSO, plus the chairmen of the three task groups formed. They were: the Short Range, under Joe Wegstein of NBS (one of the founders of ALGOL); the Intermediate Range, under Gene Smith of the Bureau of Ships; and the Long Range, under Bob Curry of Southern Railway. I name these people here because they were all active for a long time in the COBOL world, and it is doubtful if history would have been the same without their efforts. Grace Hopper and I were appointed as technical advisors. My non-technical contribution was the coining of CODASYL, for the Conference on Data Systems Languages, at the meeting following the destructive computer fire in the Pentagon.

Some of the old hands laughed disbelievingly when the Short Range task group was charged to come up with a composite within three months. Still, they worked assiduously, and held twelve meetings between June 23 and the end of August. Membership included representatives of Burroughs, IBM, Minneapolis Honeywell, RCA, Remington Rand UNIVAC, and Sylvania, plus the Air Materiel Command and the Bureau of Ships. The report was presented to the Executive Committee on September 4. Wegstein said charitably that, "it contains rough spots and requires some additions." It was, in fact, a committee hodgepodge, which hardly caused IBM to abandon plans for Commercial Translator. They were enjoined to get it in shape by December 1, and to continue in existence beyond that to monitor implementations on various computers.

Now uncertainty enters. A news item in June had stated that a new corporation was being formed (Computer Sciences Corporation) whose first responsibility would be the construction of a compiler for the Honeywell 800. Joe Wegstein gave details to the Executive Committee on September 4, when he presented his report. There is no question that the Intermediate Range Committee was dismayed by the first results of the Short Range Committee. One of its members was Dr. Richard Clippinger of Honeywell, who was able to furnish a copy of the specifications for the Honeywell Business Compiler Language (later FACT) at their October 8 meeting. FACT was a medley of the three approaches tried for Commercial Translator: (1) modular report, file maintenance and sort generators, (2) high level operators such as "UPDATE," and (3) the English language procedural statements. As a language, it was undoubtedly rich and well-defined for that period; one could well say ahead of its time.

FACT was too attractive to the Intermediate Range Committee in the face of the shortcomings of the first COBOL report; they endorsed FACT to be the basis of the common business language. The word spread with shock waves. IBM and UNIVAC, having consented to work on a composite, could scarcely be happy to scrap the work and accept in toto the language of a competitor. General doubt was expressed about the sanity of the Intermediate Range Committee, which retaliated by convening again just five days later and reaffirming its support of FACT, fifteen in favor, one opposed, and two abstaining.

Unfortunately for FACT, the implementation failed the language. Bob McDowell (first with CSC and later, Honeywell's liaison with CSC) says that this was primarily caused by an attempt to force the compiler into too small a configuration, resulting in missed schedules and customer problems (unlike third generation customers, they were not resigned to such difficulties). He feels that if FACT had been at least partially demonstrable at the time, the battle may have been won. However, in view of the competitive situation, the war might have been lost.

As it turned out, FACT was not operational until a year after the first COBOL processors. Nevertheless, the language was a substantial contributor to COBOL, and this is reflected in the later acknowledgments. In contrast, ALMACO as a language was not, being a derivation of B-0 undertaken as a joint project with UNIVAC; the contribution here was from the personnel of the Air Materiel Command. The Short Range Committee submitted its next report in 1960 January. It was accepted subject to editing for "typographical and other minor errors." The editing committee was chaired by Phillips, together with Wegstein and Betty Holberton. The minor work took from January through April, and the COBOL 60 Report was issued by the Government Printing Office in June.

Despite the inadequacies of the Report, the list of manufacturers announcing or committing COBOL implementations grew. It was simply the thing to do. However, many qualms were felt about a language defined in large part by example rather than by syntax and semantics, particularly at IBM, whose John Backus had presented his metalinguistic notation (BNF) the previous summer. General Electric's Charlie Katz, another old ALGOLer, warned in the public announcement of their language GECOM that while COBOL could be accepted by GECOM, it was not yet to the form where only one interpretation was possible.

The official IBM position on COBOL was a critical element for acceptance in the industry. Commercial Translator had been announced for the 7070, 709/90, and 705 III. Barry Gordon was responsible for the compiler implementations. Roy Goldfinger and I were working both within IBM and within the Short Range Committee to reduce the differences between Commercial Translator and COBOL, allowing the former to have extra features, particularly the computational forms of FORTRAN. As a result, the GUIDE organization was told on 1960 January 27, that IBM would include basic COBOL in Commercial Translator. The February 15 survey by the SHARE organization showed Commercial Translator as IBM's version of COBOL (oddly, IBM claimed only 80% machine-independence, Honeywell declined to quote on FACT because it was for a single machine, but Wegstein claimed 100% for COBOL).

At the February 17 meeting of SHARE (XIV), Al Harmon, Manager of Applied Programming and my boss, said that it "appears that time schedules for achieving a version of COBOL that will be satisfactory for all existing and proposed computers would unduly delay IBM's production of processors for Commercial Translator. We are revising our present Commercial Translator manual to represent our best solution to these problems. Our intentions are to revise the Commercial Translator language to include new developments, both from our own efforts and those of the COBOL committee." My verbiage in the official

IBM position, announced in Datamation magazine, was "the Commercial Translator is being reworked as nearly in the COBOL spirit as possible . . . We wish to ensure that the end result will be a single workable language for data processing." The accent on workable came from the Short Range Committee's reluctance to accept the many demonstrated logical flaws that we found in their specifications. Joe Cunningham reported in 1963 March that when the total cleanup had been made, the syntax was just as definable as ALGOL, but the semantics were prone to ambiguities.

All of this was very noble and elder-statesmanly. The only problem was that there were two versions of Commercial Translator within IBM: the one that Tom Glans, Roy Goldfinger and I specified to merge to COBOL, or perhaps even reconcile to identity; and the other that was written by Barry Gordon, that diverged. Because Gordon was in charge of compiler implementation, our good intentions came to naught. Seven big pages of differences between the two versions were compiled. I recall trying for free form in the statements in order to avoid punched card limitations, which would be quite handy for terminals today.

At the sixth meeting of CODASYL on 1960 April 7, NCR and General Electric announced their intentions to build COBOL compilers. The latter would make it part of GECOM, which also had algebraic statements like Commercial Translator (antedating PL/I), and a tabular structure facility. Because new manufacturers were asking for participation, Dr. Hopper and I were discharged as advisors, to avoid any appearance of partisanship. In May, Jack Jones of the Air Materiel Command announced the start of an 1105 COBOL. By September, there were eleven manufacturers represented on CODASYL and all had indicated that they would supply compilers--(the original six plus Bendix, CDC, GE, NCR and Philco). C.G. Holland-Martin of International Computers and Tabulators had sent a representative to the first CODASYL meeting, and ICL announced a COBOL compiler in October, superseding their own CODEL language.

The internal dissent at IBM kept building, and outside pressures were felt from the user groups. The problem escalated to T.V. Learson, now President of IBM, who solved the problem in the style of one who has access to sufficient spendable money. IBM would do both, and work toward reconciliation. Accordingly, Al Harmon told the 1960 May 17 meeting of GUIDE that IBM would supply COBOL processors for the 705 11 (without IOCS), the 705 111, 7080, 7070, and 709/90, but declined to give delivery dates. He outlined the two-phase solution of Learson, first a modified Commercial Translator and then a conversion to COBOL. This was most unsatisfactory to GUIDE, and they resolved that IBM should stick to its original statement, and that there should be only one compiler per machine, with COBOL an integral part of Commercial Translator.

IBM did not do so, however. Not because it would not, but because it could not. Roy Goldfinger made an extensive comparison in 1960 July between COBOL and the version of Commercial Translator implemented by Gordon, showing that the original objective was missed by a wide margin. To give IBM management their due, they were honestly chagrined to find that good intentions do not guarantee compatibility in programming languages.

Difficulties persisted. IBM finally announced their COBOL production schedules on 1960 October 1, but it was not until 1962 September that the success of COBOL was sufficiently apparent for Bob Ruthrauff to tell SHARE XIX that "we intend to make COBOL our development language and plan no further development of the Commercial Translator language itself." IBM did not want to put Commercial Translator under the 7090 operating system, but said it would negotiate with die-hard users. (Actually, the situation would never have come to this had not the 7090 compiler for Commercial Translator, done by a West Coast group under the direction of Dr. Richard Talmadge, been so good compared to the then-existing COBOL compilers.) In 1963 February, SHARE abandoned all hope for Commercial Translator and prepared to switch existing programs over to COBOL via translation routines and some hand work. Honeywell went through much the same difficulties and extra expense, for FACT was completed and customers were supported, despite their parallel support of COBOL.

The participants in the COBOL effort did not disdain publicity. The New York Times of 1960 August 26 announced RCA's victory in the "Computer Translating Race," although the language was not a full COBOL by any means, nor was the compiler released until December, along with the UNIVAC II compiler. A joint compatibility demonstration was held for the public, and both equipments operated the test programs identically. Nevertheless, Datamation announced that RCA was "making publicity capital out of The Sacred Project" and had "made off with the first publicity marbles." IBM received similar publicity with the COBOL 61 compiler on the 1410.

In general, the compiling techniques of early COBOL processors were primitive, resulting in very low compilation rates. Navy evaluations reported in 1962 May showed five compilers ranging from 3 to 11 statements per minute. Measurements in mid-1964 showed a range of 11 to 1000 statements per minute. Hardware did not get that much faster in two years, so we must conclude that we found out how to build compilers better. At this time, it was first noticed how drastically compiling rate varied with the size of store (memory) available, and how much variance there was in compiling cost, which ranged in this study from \$0.23 to \$18.91 per statement. Presently we are in the range of several thousands of statements per minute.

COBOL publications via the CODASYL route were COBOL 60, COBOL 61, COBOL 61 Extended, and COBOL 65. It was intended originally to update yearly, but this was changed with the advent of formal standardization in early 1963. Gradually, the CODASYL COBOL committee concentrated on development, leaving the intermediate and formal publication to the standards bodies. The European Computer Manufacturers Association had set up Technical Committee 6 to deal with COBOL, in correspondence to American National Standards Committee X3.4.4 (Howard Bromberg made his career as Chairman). ECMA TC6 and X3.4.4 worked jointly to produce the twelve COBOL Information Bulletins published.

Under the Brooks Bill, Public Law 89-306, Federal Information Processing Standards are the three-way responsibility of the NBS, the Bureau of the Budget, and the General Services Administration. The first man in the NBS position was Norman Ream, previously of Lockheed. Short funding at NBS prevented the monitoring and measuring efforts scheduled there originally. These difficulties were surmounted when Ream was appointed Special Assistant to the Secretary of the Navy, for he was imbued with the idea that standards would not be too effective without a means of checking conformity. Many compilers were claimed to be COBOL, but without a truth in packaging law or detection device, the user was sometimes misled. Ream's ingenious solution was to call back to active service Commander Grace Hopper, USNR (Ret) to lead the accelerated effort in COBOL standardization for the Navy. Of course he checked first with the President of UNIVAC. Grace was to return to active duty for six months starting 1 August 1967. Only one thing was wrong with the memo announcing this move: she did not get out in 1968, nor in 1969, nor will she in 1970. This Spring, her orders were reissued to read "indefinite." As a result, the Navy constructed comprehensive COBOL certifiers that are available to anyone.

By being brought under standardization control, COBOL profited by being related to other information processing standards. To give an example, CODASYL had an early report on the 1960 January 13 meeting of the American Standards Association, at which was authorized Committee X3, Computers and Information Processing. Character sets were very prominent in the discussion. FLOW-MATIC had 63 characters available due to the UNIVAC computer, whereas IBM had only 48 due to the punched card. Furthermore, the collating (ordering) sequences were different between the several equipments, and none of the existing sequences survived in the American Standard Code for Information Interchange (ASCII). Yet COBOL had statements in the language which gave different actions depending upon the collating sequence of the hardware. For reasons unknown to me, CODASYL chose to ignore this problem, and it was not resolved until 1969.

A COBOL specification has been adopted by the International Standards Organization, which refers to standards as Recommendations. Through the efforts of an international editing committee, and careful arrangements by the standards bodies, it is the same as the American National Standard. ECMA also contributed heavily with a formal description of the syntax of COBOL, which was invaluable in reducing ambiguities and validating constructs.

Here we are, beginning a new decade with a regularized COBOL world in both French and English. One might think that the future would hold no surprises. Theoretically, COBOL could continue to hold a prominent position in computer usage. There is an organizational structure for development, and several standardization bodies to normalize the changes and additions which are developed. Practically, there are some strong factors opposed to its immortality:

- 1) IBM has introduced a competing language, PL/I, which has attained substantial usage. PL/I also has development and standardizing bodies, at least in the USA and in ECMA. Action to find an international standard will occur at the 1971 meeting of the international committee for standardizing programming languages. It appears that COBOL programs are translatable to PL/I programs with some difficulty, but not too much more than from FORTRAN II to IV. I have no satisfactory theory of IBM's motives for promoting PL/I.
- 2) Although COBOL now has additions for data communication and data manipulation, they are appended to the same old COBOL structure originated for a uniprogramming environment. As I warned in my address to the 10th Anniversary Meeting of CODASYL, data communication and manipulation are but different aspects of general data movement, and should be unified and common to all programming languages that must exist in the same multiprogramming environment. PL/I, COBOL, and FORTRAN, having different development committees, contain data communication facilities that are not common or similar, and yet they must coexist under a single operating system. Now the data base becomes king, and there is really no need for more than one data-procedure language. The user pays heavily for this language plurality.

- 3) The concept of levels may prove vitiating in a communications world. If interchange is stressed too heavily, the tendency will be to write programs in the lowest level of COBOL, in order to assure the widest use on a maximum number of computers. This is akin to Gresham's Law, and low level programs will drive out high level capability. This would be unfortunate, for those lower levels are really unnecessary. The only, and original, reason for their existence is to permit COBOL compilation on small computers, which is something that small computers are particularly unsuited for. Far better to use the full power of a high level of COBOL, hook up by communications to a remote compilation service, and get back an object program, the running of which does not tax machine capacity the way compilation does.
- 4) Finally, COBOL and the other major languages all possess a major defect for public use of data. They carry the data description in the program. For interchange and multiple use of public data, the data must be self-descriptive. Some corresponding change in the structure of these languages is certain.

However, these technical difficulties should not deter the working user from extensive use of the COBOL language. For every successful user there are still several floundering in the morass of emulation because they just could not bring themselves to break away from machine language. The problems of program transferability are great, and the losses from not using transferable languages such as COBOL run well over a billion dollars each year. We find countless examples of programmers who have moved on, leaving programs that are undecipherable for maintenance, let alone for modification. Had they been written in COBOL, they would have had more than zero scrap or salvage value.

In my opinion, emulation is a crutch that never lets you get over the disease. Fight wastage of human resources; COBOL is a great weapon.

R W Bemer

Director of Programming Development, GE, Arizona

INTRODUCTION

Management problems arise with software systems, not for small components and subroutines. There is little difficulty in designing paper systems, but a system that is to be built and used demands extensive management. Thus the possibility of managing successfully a large software project becomes a competing design criterion. The constrictions of communication, control, decision and trade-off increase non-linearly with project size, often becoming of such magnitude that they outweigh technical design choices that are apparently independent.

It is evident that this has not been recognized adequately in the design and construction of many past large software systems. There is no need to name these failures here; many of us would like to be spared the pain of memory. It can't be size alone that causes these administrative difficulties, for there are many examples of successful administration of large projects. Is there something special about software that introduces new management traps? Some of the factors that commonly cause difficulty in software production are:

- 1 The invisibility of software
- 2 Intangibility and poor definition of the task to be done
- 3 The micro time scale of component actions with respect to time scale of human interaction, which masks inefficiencies
- 4 The entwinement of the engineering, manufacturing and distribution functions
- 5 The inability to prove the correctness of software.

However, these factors apply to varying degrees in other fields. To my mind they represent only excuses for the real problem. The real problem is that, as a relatively new profession (?), we are obsessed with reinvention and forget that there is something known

as management science. We fail to go through a simple exercise that should be standard for all development: asking a series of questions, each of which is not answered fully until there is feedback from the next question. The newspaper business has its formula of who, what, when, where and how. Similarly, the software producer should answer the questions shown in Figure 1.

<i>Question</i>	<i>Decision completeness</i>
1 What should be produced?	5-10%
2 Should it be produced?	30%
3 Can it be produced?	70%
4 How should the producer be organized?	80%
5 How should the product be tested?	90%
6 How should the product be introduced?	95%
7 How should the product be improved and serviced?	100%

Figure 1: Questions the software producer should ask.

One may quibble with these percentages, but this table gives the structure of this paper.

WHAT SHOULD BE PRODUCED?

Does it fill a need?

Answer this question carefully. At a 1958 GUIDE meeting it was reported that a user programmer had rewritten an IBM input routine to run 10% faster. Based upon programmer cost, machine time for test, and percentage of usage, it could be calculated that the rewriting would pay off in the year 2040, at which time not many 705s will be around. A trivial case, perhaps, but there is evidence that this happens also on a larger scale. Don't be afraid to discuss DP systems with your management. They have found out how much they cost and will probably listen carefully.

Don't be too ambitious initially. Goals can change as you go along; there is nothing with lower salvage value than a DP system that does a job you do not want and is too difficult to modify.

For what market?

Is it for use within your own company or can some generalization or modification in design enable it to be sold to other companies for the same purpose? Or can the algorithm be compartmentalized from the application so that it may be used for different purposes by you and others?

What advantages?

What are the advantages and disadvantages, such as efficiency and cost effectiveness? Beware the apparently aesthetic choices; don't forget production costs, use costs, life cycle, durability, reliability and maintainability. Make sure that the trade-offs are expressed quantitatively.

What are the characteristics and side effects?

Because software is supposed to educate the computer to do useful work in conjunction with humans, it should fit human capacity and characteristics. Some notable failures have been caused by ignoring this requirement.

The software system almost always overshadows the hardware system and should be treated accordingly. Raw hardware power can be degraded as much as 90% by improper software.

SHOULD IT BE PRODUCED?Will it pay for itself?

Adjust the projected gain for optimism and the costs of conversion, introduction of the new system and disruption of continuing processes.

Will it be useful when introduced?

Large software systems have long production cycles, which are commonly underestimated. David (32, p69) says "...in the past, and probably in the foreseeable future, estimates of the effort (man-years) to complete tasks involving new software concepts are likely to be low by factors of 2.5 to 4." Periodic reviews are useful during the production cycle to see if the original assumptions still hold. An added margin of flexibility at design time pays off in facilitating adaption to the dynamics of change.

Is it timely?

There is danger of missing the right point in technology, of being either too soon or too late. One should ask if it is possible to get along without the new system until a jump to a new technique can be made. This is a question of best strategy; all factors of the business should be considered.

Make or buy?

It may be possible to get it elsewhere, in whole or in part, either cheaper or at the same cost. Figure 2 indicates that there is a vast amount of software available, although much of it is not portable to other equipment or installations.

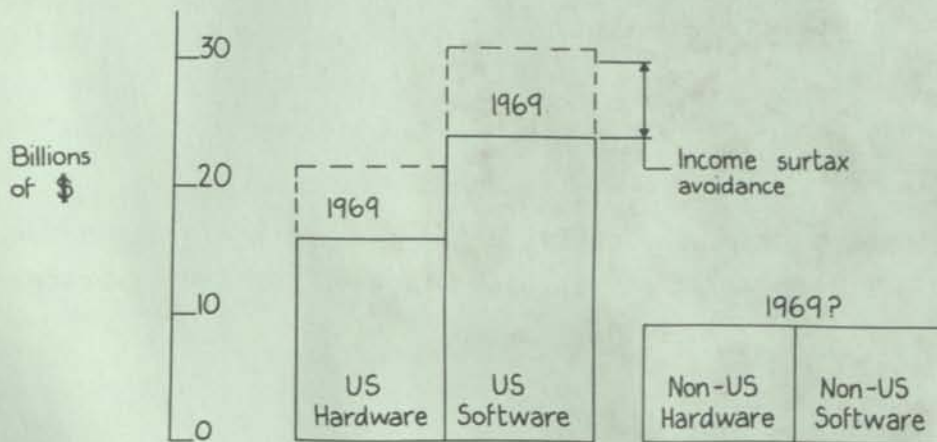


Figure 2: Data processing inventory

Salable software packages are in the ascendancy, although most still have portability difficulties. The question as to whether the software will pay for itself may have a different answer if the software is planned for resale. The best sources for obtaining outside software are user associations, software houses and trade associations.

Remember that usually only one of a kind is necessary. Don't buy, externally or internally, more than is necessary. I know one operating system that has 20 different Get/Put routines in it, written by 20 different programmers.

How should the final decision be made?

I have not seen a quantitative answer to this question. When it was first asked of me, at IBM in 1957, the reply was to get a man with the best batting average in extrapolation and trust proportionately to his judgement. This still seems the best answer to such a complex question.

CAN IT BE PRODUCED?

Is it possible at all?

My most lasting impression of J. Paul Getty came from a Playboy article in which he said that the smart man does not take on the impossible. There are software systems that are neither feasible nor possible to build, given even unlimited resources of programmers and computers. There are 2 extreme methods of production.

- 1 Plan the system all at once, then build it all at once.
- 2 Follow my 5-word motto, "Do something small, useful, now" with, of course, an eye to the changing future.

I don't have much faith in the first method, particularly for data processing, because every big management scheme I have seen has died for 2 reasons:

- 1 The planning was so monolithic and it took so long to do that the proposed system was out of date before it could be implemented.

- 2 Even then, it could not be corrected or modified because the lack of results led management to put the planners out of a job. Obviously, the function that the proposed system was to carry out was being performed somehow during this period.

Assuming all else is OK, one should keep an eye upon the permanence of one's management and its goals. A new boss will often redirect effort and restart nearly from scratch. This is common for elected public officials.

Are the resources adequate?

Here we speak of all resources: money, talent (not manpower), time, technology and direction; they must all be allocatable to the project. Be careful when offered miracles. For years I have carried a little cartoon in my billfold. It shows two programmers looking at a printout, one of whom says, "Hey Joe! It says our jobs are next!" Don't you believe it; we couldn't get rid of those two in any way!

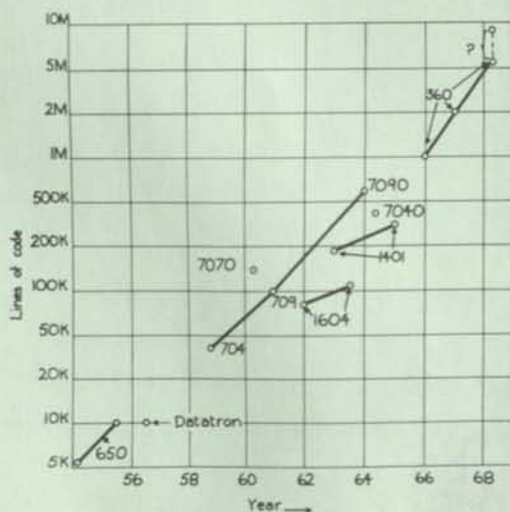


Figure 3: Growth in software requirements

Figure 3 is the McClure Chart (32,p66). It shows how many instructions you will get if you ask a manufacturer to give you his software for a certain system. Note that the vertical scale is logarithmic. This makes me fear that, in a few years, not only will Joe and his buddy still be around, but so will 20 more programmers and there may not be that much suitable programming talent

available, even with a massive educational effort.

Is the size of a software system a worrying factor? It certainly is, because productivity for basic software has not increased with system size. If anything, it has decreased markedly; this is not surprising when one considers the inevitable increase in connectivity. Figure 4 is my compilation of some productivity statistics, in terms of instructions per hour; both scales are logarithmic.

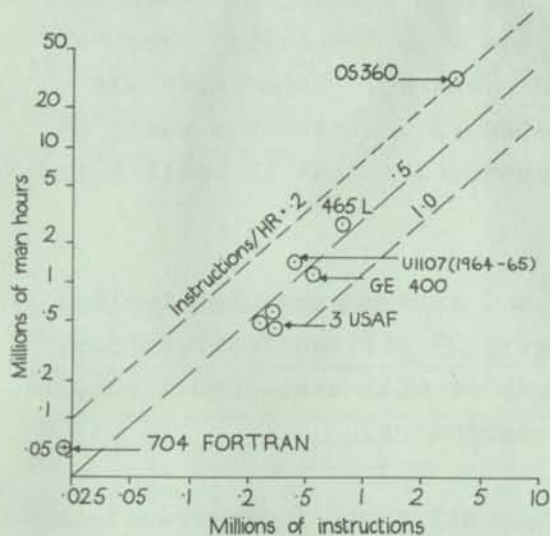


Figure 4: Software productivity

This chart is designed to reflect total budget figures, on the basis of approximately 30% for design and implementation, 20% for test and 50% for management, documentation and support. I have arrived at the OS/360 figure in several consistent ways, which are worth enumerating here:

- 1 Conway, in his paper in *Datamation*, October 1968, postulated an expenditure of \$15 million in 1963, \$45 million in 1964 and \$60 million for the years 1965 through 1968. This is consistent with official IBM figures as reported in *Fortune* magazine for October 1966. This means an expenditure of \$300 million to produce the 5 million instructions the McClure Chart shows for the end of 1968; in other words, \$60 per instruction.
- 2 Original information released on the System/360 software showed an expenditure of \$160 million for about 3 million instructions, as produced by 3000 programmers at peak. This averages \$53 per instruction.

- 3 An IBM spokesman asked me at the 1967 February SHARE meeting, "Would you believe \$53.50?".
- 4 Assume programmers at \$20 000 per year for 2000 hours of work. At this \$10 per hour rate, 0.2 instructions per hour would cost \$50, which is quite consistent.
- 5 3000 programmers at \$20 000 per year yield the \$60 million per year figure that Mr T J Watson gave to the 1966 March meeting of SHARE.

These production figures will seem low to many. One should not forget that they are for very large, mature systems of basic software. The cost of an instruction rises with longevity, because these systems must be maintained and enhanced. Some parts are rewritten several times and the superseded instructions can't be counted anymore, even though their production cost is still a factor.

My nightmares come from imagining a new system scheduled for 1972. If the McClure chart holds true to give 25 million instructions, then the best figures we have say that it will cost 1250 million dollars and be produced by 15 000 programmers.

An obvious objection is to ask whether all those instructions are necessary; isn't it likely that there is some deadwood? According to David (32) this is very possible, as demonstrated with the MULTICS system, as shown in Figure 5.

MULTICS: 1,000,000 reduced to 300,000			
Module	Improvement		
	size	perf	man/months effort
Page fault mechanism	1/26	50/1	3
Interprocess communication	1/20	40/1	2
Segment management	1/10	20/1	.5
Editor	1/16	25/1	.5
Input/output	1/4	8/1	3

Figure 5: Why so many instructions?

One would hardly express it as a law, on the basis of so few samples, but in these cases it appears that, if the program is reduced to $1/N$ its former size, it will run $2N$ times faster. Part of the excess was due to use of a higher level language, of course, but this should not be taken as an argument against higher level languages per se. The sin is in using them in disregard of hardware characteristics. Code expansion is not the only culprit; duplication and unuse constitute an area of very high potential for deletion of excess instructions.

Now, if we can get a defined minimum of useful instructions to produce, let's consider the people who are going to produce them. Figure 6 is a serendipity product of work done by Sackman, Erikson and Grant, which was published in the Communications of the ACM, January, 1968.

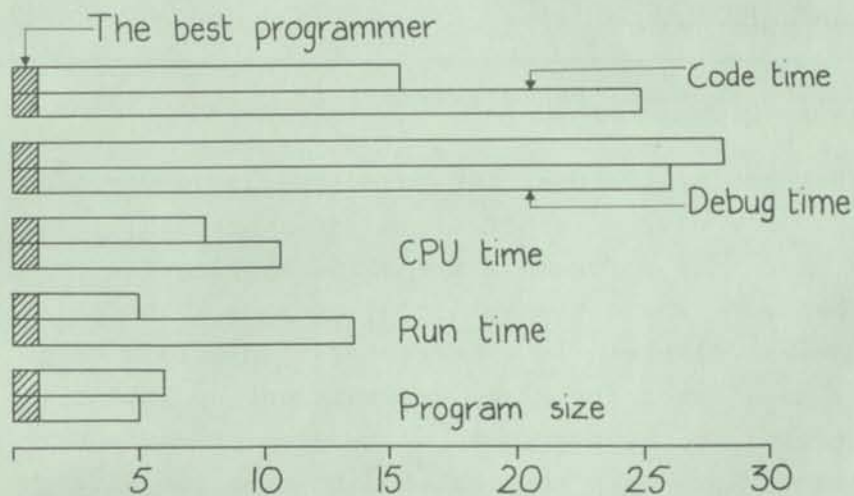


Figure 6: Some variability in programmers

In explanation of the serendipity, these results came from an experiment to measure the effects of on-line versus off-line programming, but differences between individual programmers were so great that they voided any possibility of measurements for the original purpose. Two identical problems were given to a group of 12 programmers with an average experience of 7 years.

I told the authors that I considered the paper in which these data appeared to be the most important work in the computer field in 1968; this opinion is unchanged. Here we see more justification for asking if all those instructions are necessary. Naturally,

not all the worst cases in each category are due to the same bad programmer, but the correlations are somewhat monotonic.

Note that the product of CPU time and program size would be the degradation factor in a multiprogramming system. This speaks strongly for the position that the best programmers should be selected and screened for the production of basic software. So does Figure 7, which deals with not the originating cost, but the cost of usage by the computer world.

- | | |
|--|---|
| <ol style="list-style-type: none"> 1 2 | <p>During early instrumentation, 7% of GE 600 FORTRAN compile time was found to be in 4 instructions, easily reduced to 2 (3.5% saving). Suppose this were true for all software on third generation systems, with an installed value of:
\$14 Billion.</p> <p>If 10% of usage is FORTRAN, and 40% of that is compilation, then 2 redundant instructions waste:
\$14 billion \times 0.1 \times 0.4 \times 0.035 = \$19 600 000 or about
<u>\$10 million per instruction!</u></p> |
|--|---|

Figure 7: Magnification factors: basic software

To me, this is a frightening picture, for these are unrecoverable costs to the user. Don't pass this off as an isolated case, for it is demonstrable that most software systems and application programs are honeycombed with waste elements that surpass this one. Don't think that General Electric is embarrassed to disclose this case; after all, we have taken our waste elements out by intensive application of instrumentation programs! We're even proud, considering that congressmen have been re-elected with ease for saving the public this much money! In fact, there are potential savings of over a thousand million dollars to be realized by demanding instrumentation and measurement of software.

Thus we see one critical input to the pragmatic question as to whether the software can be produced. Many people have given recipes for extrication from this dilemma, all put forward with great fervour and, inversely, little hard justification. My list is as follows.

- 1 High level languages to write in; no one will quote more than a 3:1 advantage and we have seen how that can be abused.
- 2 Good software management; if you can find it, train it, allow time to train it, keep it and keep it programming!
- 3 A software production environment (the factory).

4 Good programmers.

A few heads may nod, thinking I have listed these items in decreasing order of importance. On the contrary, they are in increasing order of importance. The good programmer is the key, just as the top engineer is the key for hardware. The Univac 1004, a very successful piece of equipment, was designed and built by not more than a dozen people in what was called "The Barn", in Rowayton, Connecticut. As an aside, they did not build it to fit the existing market; their product shaped the market!

Are the production methods available?

The generality of a data processing system makes possible the finest production methods, yet these are seldom exploited to advantage. A major drawback of large software systems is that a substantial portion of the production cycle is often wasted by the invisibility of software; when the programmer finally builds something, we find it is not what we wanted. Then we must build something else and the lost time cannot be regained. Large systems are too complicated to depend upon intuitive design; one individual cannot comprehend them totally, nor can a group of people cooperate in their construction and the communication surrounding them without mechanical aids. An ideal plan is to build a model or skeleton and, if it acts as we wish, to replace the simulated units by real units. This must be done carefully, replacing units one at a time, to avoid confusion.

Production identification, change control and labour distribution are important tools. Their very tangibility for estimating provides the capability of recalibrating to better estimates. They also make it more difficult for programmers to lie to themselves and, perhaps, to their management. I also have a personal predilection for standing in a machine room and sensing what is really happening. Then this can be matched against the production control, scheduling and costing. PERT has failed in many large software projects, sometimes because there was not enough time to provide the inputs, sometimes because it only gives the latest time one can do something provided absolutely nothing goes wrong, with all of the other things that people waited until the deadline to do!

Is a good production environment available?

If computers are useful for a general class of problem, then they should be useful for producing their own systems. Some manufacturers are now building a software factory, or an environment residing upon a computer within which all software production takes place. Programmers are directly on-line via terminals and key-punchers are bypassed.

It is true that such systems will be very useful eventually for management control in large projects but, in the beginning, the accent and priority for delivery must be on service to the programmer. The programmer is independent, so we must make the system attractive to him and worth his while to use. One of the most important aspects is increasing the number of accesses to the computer per day. This is as low as 1 or 2 in much of our industry. At this slow pace, programmers tend to lose the thread of their thought and spin wheels. The difference between the good and the bad programmer may well rest upon the need for cohesiveness and pattern. Perhaps it is like half-life decay of radioactive materials. In any event, this is a crucial factor in the success of large projects, yet it has been treated as virtually unimportant!

Greg Williams of GE has proposed a lesson for management on why software is so expensive under limited access conditions. He would like to ask managers to use the BASIC time sharing system to convert clock time to Gregorian, to head an output or for use in an accounting system. A simple, everyday problem, but he hasn't had the nerve to try it yet. His estimate for management is 100 mistakes, finding 2 per day. That is a long clock time!

HOW SHOULD THE PRODUCER BE ORGANIZED?

There is no one answer to this question and there may be more than 100. Software mirrors intellectual processes, which are capable of infinite variation. Some organization is required, for large software projects must be subdivided. It is important to observe Conway's law, that the form and size of the product reflects the form and size of the planning organization. The sub-organizations should be structured to follow the design and architecture, with the program interfaces under the control of higher management.

How large should modules be?

Several authorities fix this figure at from 400 to 1000 instructions per programmer. If this seems low, recall that a gross of 0.2 instructions per hour means about 1 per hour by the programmer actually writing them, so that 1000 instructions represent about a half-year's work. This figure may be raised significantly for smaller projects with fewer interactions, but seems to hold well for systems of more than 250 000 instructions.

Who should be in charge?

Software engineering has much to learn from hospitals, where the doctor does the work, with his decisions generally overriding the administrator's. For large projects, I favour a leader who is a working programmer, not just a supervisor. If possible, it should be at least his third project of that type, although he need not necessarily have been in charge previously. The reason for this is that, the first time, he re-invents, ignoring literature, competition and scrounging; the second time, he is too confident that he can avoid all the mistakes made the first time. The GECOS III operating system for the 600 is a splendid example of such avoidance of the Peter Principle, which takes us to another question.

How should design and implementation be partitioned?

Here we run head-on into the old argument about system analysts versus programmers versus coders. Contrary to intuition, such a division may make sense for small projects, but not for large systems! A much more careful and practical design will originate from the man who knows he will be stuck for a year or more in its production! Additionally, the reasons for the design are so much in his cognizance that he is alert to signals that a design change may be desirable.

How does the new project co-exist with present work?

Present stratification is usually by job title or project assign-

ment. Perhaps we should try to stratify the individual programmer, giving him concurrent responsibilities in several aspects: design, implementation, maintenance of his previous work, at least on call, and, in the case of software houses, assistance in customer sites. Admittedly, certain projects may be too complex to permit distraction, but these are few. A side benefit may be found in closer connection between present and future software, in the area of data and program transferability.

HOW SHOULD THE PRODUCT BE TESTED?

The proper design and use of extensive testing is mandatory. Automobile manufacturers have their test tracks to detect failure and weakness before they make multiple copies to be driven by customers over whom they have no control. In fact, the French call software testing "rodage". This testing becomes even more vital with separate software pricing and also with considerations of public welfare and safety as computers become further integrated into human activity. This means that a significant portion of the total production costs must be allocated to this function.

What should be tested?

The two major categories of quality standards are performance and compliance. Unfortunately the first is only now getting its full share of attention.

What are the testing tools?

A few tools are enumerated here; many more are possible and in use. Performance testing is necessary because any given process can fall into one or more of the following categories.

- 1 Unnecessary
- 2 Done more times than necessary; that is rerun
- 3 Too slow due to hardware
- 4 Too slow due to software
- 5 Too slow due to hardware/software imbalance
- 6 Undesirable, but imposed by conflicting or non-existent standards
- 7 Not helpful because of logic conflicts

8 Satisfactory

Any of these can occur in the following.

- 1 Basic system software, in which case the supplier should fix it.
- 2 Application usage, in which case the supplier should advise and also control the default options for preferred usage when possible.

For performance testing the tools are as follows.

- 1 Standards of comparison. With parameters of hardware performance, such as the Gibson mix, number of object instructions and precision of input and output, certain common functions can be compared against what is considered good quality in the industry. For example, if the sine/cosine evaluation routine runs extraordinarily slowly compared to how comparison standards say it should do, it should be considered for re-writing.
- 2 Periodic instrumentation, either by hardware (zero time) or software (finite time, not supportable continuously). Hardware instrumentation is accomplished normally by tapping in a second computer system or a special hardware device. Software instrumentation can consist of any of the following.
 - (a) interface tracers, for connectivity
 - (b) trapping analysis of module usage, timing, control acquisition and release and so on.
 - (c) hardware-initiated actions for later software analysis.

As an example, Ellison set the rundown timer on the GE 600 extraordinarily low, so that a given process could barely get started before interrupt and relinquishment of control occurred. In this way a normal 24-hour usage took almost 48 hours, while the actual store location of the instruction being executed at interrupt time was recorded. A later count and distribution gave an excellent Monte Carlo simulation of the frequency of usage of the various software modules. This was how the anomaly shown in Figure 7 was detected.
- 3 Continuous instrumentation (low and supportable time allocated to this purpose). This would include the following.
 - (a) gathering statistics during operating system time for later analysis.
 - (b) monitoring resource allocation and usage for real time display to the operator, preferably by CRT.

For compliance testing the tools are as listed below.

- 1 Generalized tests for well known standards, such as the US Navy COBOL Certifier.
- 2 Special tests written, concurrently with the software production cycle, to test conformity to specifications. At Bull General Electric, I had at least 1 programmer in every 10 allocated to such tasks, which is not a surprising ratio, considering that the testing function for large systems can use as much as 20% of the total budget.
- 3 Test cases. Formerly, these have been considered as primarily for application programs but they are also particularly valuable for testing successive system revisions. One accumulates a test file of the malfunctions reported for previous versions of the system, together with a sampling of small applications. The file resulting from processing with the new system is mechanically compared with the previous answer file and deviations displayed for analysis.

Quality in both performance and compliance is checked by field test. One would wish to avoid such a procedure if possible but, most of the time, this is impossible for large systems, time sharing systems being a particularly visible example. It is not known, nor have the computer scientists provided us with any insight, how to simulate and test a large multi-access system by means of another computer program that exhibits the real time properties listed below.

- 1 Any randomly possible selection from the US communication system.
- 2 The US population making other demands upon that system.
- 3 An unpredictable user population, either in loading or arbitrary usage.

My company has found that many strenuous measures must be taken to check out new time sharing systems. We have even switched a large number of internal users from the regular national system to the field test system in order to provide instantaneous overloads, peculiar usage combinations and time-of-day variations.

When should it be tested?

Quality control is continuous testing, during production, by the

producer. Quality assurance is discrete testing, after production, by an agency acting on the behalf of the user.

Both of these functions must be recycled for major revisions. A reasonable criterion for a large operating system is that it shall perform continuously for at least 2 weeks without a malfunction affecting the user. It will usually be a minimum of 3 months before such a status is achieved. Needham of Cambridge University says, "There are very few bugs in our operating system that weren't put there in the last 2 weeks".

What are the authorities?

This is a management decision but it may be delegated by management to the quality assurance group. Seldom, or never, should the release responsibility reside with the producing group. Where public safety and welfare depend upon the software, perhaps a quality assurance group veto should be protected from a management override.

When is the system correct?

For large systems, it has been recognized that the answer is, "probably never". One should reject the interpretation of 100% "mathematical" or "logical" correctness for software engineering purposes, for reasons of statistical frequency of exercise and because of the program interaction with the data.

- 1 A 99.9% correct program is no better than a 99.8% program, if the data is only 80% correct.
- 2 If the hardware has a logic flaw, but has a superimposed FORTRAN processor that never exercises that feature or causes it to be exercised, then the combined system may be said to be correct (apart from other possible flaws).

In short, correctness to the software engineer means that a system should do the "proper" thing rather than perform exactly the actions that were specified with such imprecise knowledge.

I prefer the following interpretations of correctness.

- 1 Design correctness
 - (a) efficient utilization of production resources

- (b) efficient utilization of system resources during running
 - (c) maintainable and reliable
 - (d) constructible
 - (e) flexible, for purposes of change and the addition of functions.
- 2 Implementation correctness
- (a) matches the specifications
 - (b) solves the problem envisaged
 - (c) free from malfunction
 - (d) free from hang-up or locking.

For those who may feel dissatisfied with this thesis, I quote Schorr of IBM (7). "Apollo 11 software acceptance testing took about 2 months; it was at least 30 days before anything would even start to run in real time. Bugs were taken out of the software up until the day before launch."

Thus we see that system planning for incorrectness is far more effective than excessive emphasis upon an absolute correctness that cannot be achieved in finite periods of time.

HOW SHOULD THE PRODUCT BE INTRODUCED?

What are the general requirements?

The introduction of a software product is dependent upon the constitution of the product, which may include the following.

- 1 The working software or the wherewithal to generate the working software.
- 2 Operating instructions and rules.
- 3 Technical documentation on data forms, source, actions, flow-charts and all the other elements normally associated with program transferability.
- 4 System support, if the product is from an external supplier.

What are the requirements for a new, rather than a replacement, product?

In this case the main problem is the effect it may have upon data

and data file structures used by other programs. It may be completely independent of the action of other programs and yet have strong interactions with common data.

It is desirable to have sample runs supplied for duplication in the production environment.

If it replaces a previous product?

- 1 If it is an update, performing basically the same functions, the main requirement is for a period of parallel running with the old program, comparing production answers.
- 2 If it is a new product, performing similar but not identical functions, there must be an overlap period to allow the old product to be phased out. Whenever possible, the interface to humans should be consistent with the former interface, as in operating modes, messages and so on.

If data conversion is required?

Several types of conversion may be required, such as those listed below.

- 1 Graphic set content, encoding and character size
- 2 Precision and range of numerals
- 3 Data formats
- 4 File content (added, changed or deleted)
- 5 File structure
- 6 Media labelling
- 7 Physical media formats

Once-and-for-all conversion is the exception and it may be advisable to have it done on a service basis, particularly by an outside supplier. More commonly, the new and old products must co-exist until the new one is proven sufficiently. In this case, it may be useful to have separate files for both the old and the new product, with a bi-directional conversion program to verify identity between the 2 versions at each stage.

HOW SHOULD THE PRODUCT BE IMPROVED AND SERVICED?

What types of improvements are possible?

- 1 Progress towards data and program transferability may require any of the following.
 - (a) changes to comply to standards: programming languages, character sets, data structures, media labelling and so on.
 - (b) making the data files self-descriptive and self-identifying; for instance, copying the data division of a COBOL program on to the data medium.
 - (c) keeping programs in source form, without patches. If patches must be employed, recover source form periodically and promptly.
 - (d) making the program self-documenting.
 - (e) improvement in ease of usage, attention to human factors.
- 2 Additional capabilities, as in the list below, may be introduced.
 - (a) new functions or features not previously available.
 - (b) functions or features of existing programs that may be taken over, obviating the need for those programs.
 - (c) more choices of algorithms, for better efficiency in alternative situations. This is a particular requirement for basic software, which is optimized for a hypothetical, often non-existent, user. The user should be provided with instrumentation software and generators for specialization.
- 3 Better performance may be obtained by the following means:
 - (a) instrumentation
 - (b) design analysis
 - (c) restructuring data files,
- 4 More reliability may be obtained by means of the following:
 - (a) elimination of hang-up conditions
 - (b) confidence and range testing, checking for reasonableness.

Who should service the product?

- 1 Trainees? This is usually thought to be a good method of indoctrination and of giving trainees experience. However, there are severe drawbacks. The trainee can pick up bad habits, will get bored and discouraged easily, takes excessive time to correct malfunctions and may disrupt other parts of the program

- in the process of making a specific symptom disappear.
- 2 Experienced support personnel? It is rare to find programmers who will be happy on a steady diet of correcting other peoples' mistakes. Forcing them to continue in this function for long periods of time leads to job dissatisfaction and resignations.
 - 3 The originator? Why not? He should know it best and he doesn't necessarily spend all of his time in support, if other attractive duties are made available to him. If he considers it a trap, let him know that nothing but excellent and self-explaining documentation will release him; he himself can construct the key for release. If the product is substantial and used on a customer site, as a major basic system for a computer line would be, it could be desirable to put originators at these sites to keep up-to-date on field experience. They can still participate in new software production via remote terminals.

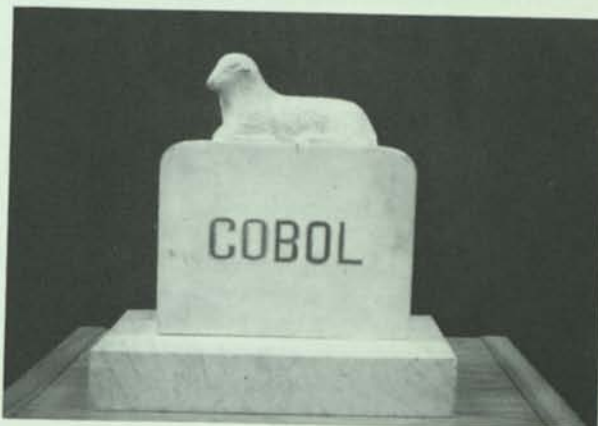
CONCLUSION

Software engineering is in a crisis of identity and maturation and this has and will lead to the promotion of various panaceas, justified by saying that nothing similar has existed before. These panaceas should not be believed. We need to use our present tools, under good management practices, more than we need new and spectacular developments, many of which do not pay off. Art must be reduced to engineering and software made visible to management in order to avoid the present high spoilage and non-transferability rates. The most profit lies in tooling for production, building new systems via old systems that are stable and mature, instrumenting for effectiveness and standardizing to make user-developed software re-usable and to reduce needless variety.

A personal recollection of the development of the COBOL programming language, complementing the formal histories. With this background, some possibilities and recommendations for the future of COBOL are given.

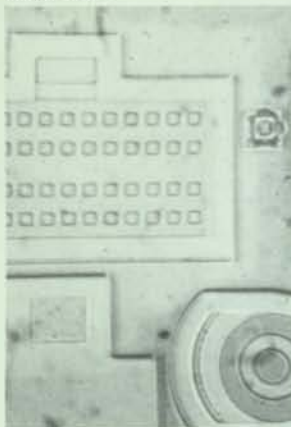
a view of The History of **COBOL**

R. W. Bemer

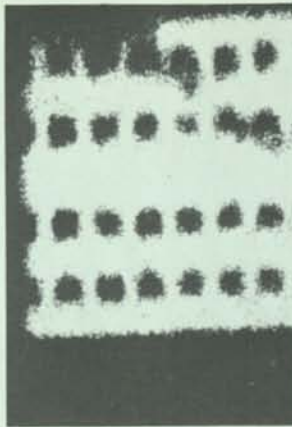


At a time when the future of COBOL was less certain, Charlie Phillips received a package from Howard Bromberg, express collect. This is the tombstone that he paid \$15 for.

Visual Optics



28_{Si}^+



1_{H}^+



23_{Na}^+



100 μ m

FIGURE 13 Ion Probe Mass Analyzer Scanning Images
— Silicon Semiconductor (150X)

ACKNOWLEDGEMENTS

The author is indebted to Chris Mitsios, who performed much of the intricate sectioning and light-optical microscopy described in this paper. He acknowledges the help of Tom Koch with the scanning electron microscopy, and the stimulating discussions with many members of the Phoenix Computer Operations. Glenn Brockman, Manager, Process Development Laboratory, reviewed the manuscript and encouraged the work.

The infrared thermogram of Figure 6 was furnished through the courtesy of Barnes Engineering Company, Stamford, CT.

The EBIC photomicrographs were furnished through the courtesy of the Jet Propulsion Laboratory, California Institute of Technology.

REFERENCES

1. F. K. Möllring, *Microscopy From the Very Beginning*, Carl Zeiss, Oberkochen, WEST GERMANY.
2. G. Nomarski and Mme. A. R. Weill, "Application of Double Polarized Beam Interferometry to Metallography", (in Fr.) *Rev. Met.* **52**, 121 (1955).
3. S. Tolansky, *Surface Microtopography*, Wiley, New York (1960).
4. D. McLachlan, Jr., "Extreme Focal Depth in Microscopy", *Appl. Opt.* **3**, No. 9, 1009-1013 (1964).

5. C. W. Oatley, W. C. Nixon and R. F. Pease, "Scanning Electron Microscopy", *Advan. Electron. and Electron Phys.* **21**, 181-247 (1965).
6. J. C. Russ, "Light Element Analysis Using the Semiconductor X-ray Energy Spectrometer with Electron Excitation", *Energy Dispersion X-ray Analysis*, ASTM Publ. 485, 217-231 (1971 May).
7. W. Kayser, "Magnetization Creep in Magnetic Films", *IEEE Trans. Magn.* **3**, No. 2, 141-157 (1967).
8. F. Lee, Private Communication (1971 Jul).
9. P. B. Mee, "SEM Examination of Thick Film Multi-layer Substrates for Beam Lead Attachment", Internal Rep. (1971 Mar).
10. K. C. Evans and J. Devaney, Private Communication (1971 May).
11. A. N. Broers, "Some Experimental and Estimated Characteristics of the Lanthanum Hexaboride Rod Cathode Electrode Gun", *J. Sci. Instrum. (J. Phys. E.) Series 2*, Vol. 2, 273-276 (1969).
12. A. V. Crewe, J. Wall and L. M. Welter, "A High Resolution Scanning Transmission Electron Microscope", *J. Appl. Phys.* **39**, No. 13, 5861-5868 (1968).
13. N. C. MacDonald, "Auger Electron Spectroscopy for Scanning Electron Microscopy", *Proc. 4th Annu. SEM Symposium*, Chicago, 89-96 (1971).
14. G. R. Booker, "Applications of the SEM Channeling Pattern Technique to Materials Problems", *Proc. 4th Annu. SEM Symposium*, Chicago, 467-472 (1971).
15. G. W. Stroke, M. Halioua, A. J. Saffir and D. J. Evans, "High Resolution Enhancement in Scanning Electron Microscopy by A Posteriori Holographic Image Processing", *Proc. 4th Annu. SEM Symposium*, Chicago, 57-64 (1971).
16. M. Hatzakis and A. N. Broers, "High Resolution Micromachining Techniques Using Electron Beams", Notes from 3rd Annu. Conf. Solid State Devices, Exeter Univ., ENGLAND (1969 Sep).
17. L. Curran, "Electron Beams Shine On IC Layouts", *Electronics* **44**, No. 13, 83-84 (1971).
18. T. H. P. Chang, "Device Fabrication Using a Scanning Electron Beam System", *Proc. 4th Annu. SEM Symposium*, Chicago, 417-424 (1971).
19. F. E. Holmstrom and J. M. Engel, "New Magnetic Identification of Magnetic Flaws in Oxide Coatings", *J. Appl. Phys.* **42**, No. 1, 1354-1356 (1971).
20. R. F. M. Thornley and J. D. Hutchinson, "Magnetic Field Measurements in the Scanning Electron Microscope", *IEEE Trans. Magn.* **5**, No. 3, 271-275 (1969).
21. J. R. Banbury and W. C. Nixon, "A High Contrast Directional Detector for the Scanning Electron Microscope", *J. Sci. Instrum. (J. Phys. E.) Series 2*, Vol. 2, 1055-1059 (1969).
22. J. F. Norton and W. C. Hughes, "Integrated Circuit Diagnostics Using Electron Beams", *GE Rep. 68-C-170* (1968 May).
23. H. Liebl, "Ion Probe Mass Analyzer", *J. Appl. Phys.* **38**, No. 13, 5277-5283 (1967).

FROM THE EDITOR

This article first appeared in M. Berk's "The Programmer's COBOL", Inter-ACT/McGraw-Hill. It is reprinted with permission of the publishers for the large number of people who use the COBOL language without any real idea of how it came to be. A historical framework, if not too dull, can add some interest to the daily work.

The content has been verified by many of the active participants. The article attempts to give full credit to the various organizations that participated. Although the viewpoint (not bias, hopefully) is necessarily that of an IBM employee at the time, the author is now with Honeywell Information Systems, and can take much pride in their part of the development of COBOL.

It isn't all history, however. There are some predictions for the future. Some of these will come to pass without user pressures; some will not. Perhaps readers will see here some viewpoints and standards actions that they should be supporting for their own best interests, and for the best interests of the computer-using world.

COBOL is a programming language known to a large number of people who work in data processing. It is less known to the general public, and chemists becoming involved in data processing for the first time are prone to confuse it with Element No. 27. It is unique among the major programming languages in that factual histories of its inception and development abound. Standard summaries of the history are carried in all official documents and in the manuals for specific machine implementations, a policy of the original sponsors.

A very complete history and summary of activities is to be found in the publication "American National Standard X3.23, COBOL". With this, one can track meetings, participating personnel, and technical motivation. Omitted are elements of personality, background, competition, infighting, and significance to the data processing world. Reading this history and others, one might conclude that there was never any excitement, strategy, or corporate and individual struggles. Not so.

The very fact that these standard histories are carried in official COBOL documents is a key to understanding the COBOL effort. COBOL is intended to conserve costs and human resources, but any of the proprietary languages of its class could have done that, and very possibly they would all have grown and matured in the same way. IBM's FORTRAN became an industry standard because it was operational in volume before its competitors, and because IBM placed it in the public domain. In the business data processing world, the race was much closer.

ORIGINS

In time sequence of development, the three progenitors of COBOL were: FLOW-MATIC (from UNIVAC), Commercial Translator (from IBM, which ran into legal conflicts with the original name COMTRAN), and AIMACO (from the Air Materiel Command in Dayton).

FLOW-MATIC was an outgrowth of the A-series of algebraic and scientific compilers. The concept of the compiler is largely due to Dr. Grace Murray Hopper, who was in charge of these projects. The new series started as B-0 (B for business, as opposed to A for algebraic). A predecessor, BIOR (Business Input Output Rerun), was developed by a different group, becoming somewhat operational in 1955 April. FLOW-MATIC, as B-0 was renamed, became operational for the UNIVAC I and II in 1956 December. It was not what we would call today a commercial-grade software product, and both language and compiler were still undergoing continual change and improvement.

The competitive threat potential of FLOW-MATIC did not go unnoticed at IBM, and some research was started in the Fall of 1956 on alternate solutions for a business language. The original approach tended to high-level operations and set notation, such as "MERGE FILEA WITH FILEB ON KEY 3" and "UPDATE THIS WITH THAT". I began to worry that this approach might take too long to bring to practicality, and asked Roy Goldfinger to develop a language with the more specific procedural capability of FLOW-MATIC, yet which would retain the set principles. Public notice of the Commercial Translator work was given to the SHARE group in 1957 October, and Roy produced formal specifications in 1958 March.

Reading the standard COBOL histories, one could get an impression that the early meetings were spontaneous. Actually, Mary Hawes of Burroughs had buttonholed Dr. Saul Gorn of the University of Pennsylvania at the Western Joint Computer Conference in San Francisco on 1959 March 3-5, asking if he didn't think it was time for a common business language. Saul agreed and later that month held a meeting in his office at the (UNIVAC I) Computer Center. At a second meeting on April 8, various names were suggested for leadership. Grace Hopper proposed Charles Phillips of the Department of Defense. It seemed most reasonable for DOD to sponsor such an effort, which would need energetic leadership and neutrality, together with the stature (and pocketbook) to command the attention of the manufacturers. I sent the agenda for the May 28 meeting to



Some members of the CODASYL Executive Committee witnessing the compatibility demonstration on 1960 December 6. From left to right — Smith, Dillon, Albertson, (Danny Goldstein), Phillips, Grosz, (Jack Jones), Curry.

the President of the Association for Computing Machinery (probably as a needle for the disdain of business languages in the ACM Programming Languages Committee, although they were within their scope to develop).

THE PEOPLE

The Washington meeting of May 28 noted the separate development of three similar languages, and agreed that the example of ALGOL warranted an effort to develop a common business language. A steering (later executive) committee was formed with Phillips as Chairman, Joe Cunningham of the Air Force as Vice Chairman, Gene Albertson of U.S. Steel, Greg Dillon of DuPont, Mel Grosz of ESSO, plus the chairmen of the three task groups formed. They were: the Short Range, under Joe Wegstein (one of the founders of ALGOL) of the National Bureau of Standards; the Intermediate Range, under Gene Smith of the Bureau of Ships; and the Long Range, under Bob Curry of Southern Railway. I name these people here because they were all active for a long time in the COBOL world, and it is doubtful if history would have been the same without their efforts. Grace Hopper and I were appointed as technical advisors. My nontechnical contribution was the coining of CODASYL, for the Conference on Data Systems Languages. We can't find a single individual who admits coining the acronym "COBOL".

EARLY RESULTS

Some of the old hands laughed disbelievingly when the Short Range task group was charged to come up with a composite within three months. Still, they worked assiduously, and held twelve meetings between June 23 and the end of August. Membership included representatives of Burroughs, IBM, Minneapolis Honeywell, RCA, Remington Rand UNIVAC, and Sylvania, plus the Air Materiel Command and the Bureau of Ships. The report was presented to the Executive Committee on September 4. Wegstein said charitably that "It contains rough spots and requires some additions". It was, in fact, a committee hodgepodge, which hardly caused IBM to abandon plans for Commercial Translator. The task group was enjoined to get it in shape by December 1, and to continue in existence beyond that to monitor implementations on various computers.

COMPLICATIONS

Now uncertainty enters. A news item in June had stated that a new company was being formed (Computer Sciences Corporation) whose first responsibility would be the construction of a compiler for the Honeywell 800. Joe Wegstein gave details to the Executive Committee on September 4, when he presented his report. Unquestionably the Intermediate Range Committee (the task groups had by now been upgraded) was dismayed by the first results of the Short Range Committee. One of its members was Dr. Richard Clippinger of Honeywell, who was able to furnish a copy of the specifications for the Honeywell Business Compiler Language (later FACT) at their October 8 meeting. FACT was a medley of the three approaches tried for Commercial Translator: (1) modular report, file maintenance and sort generators, (2) high-level operators such as "UPDATE", and (3) the English language procedural statements. As a language, it was undoubtedly rich and well-defined for that period; one could well say that it was ahead of its time.

FACT was too attractive to the Intermediate Range Committee in the face of the shortcomings of the first COBOL report; they endorsed FACT to be the basis of the common business language. The word spread with shock waves. IBM and UNIVAC, having consented to work on a composite, could scarcely be happy to scrap the work and accept *in toto* the language of a competitor. General doubt was expressed about the sanity of the Intermediate Range Committee, which retaliated by convening again just five days later and reaffirming its support of FACT: 15 in favor, 1 opposed, and 2 abstaining.

Unfortunately for FACT, the implementation failed the language. Bob McDowell (first with Computer Sciences Corporation, and later Honeywell's liaison with them) says that this was caused primarily by an attempt to force the compiler into too small a configuration, resulting in missed schedules and customer problems (unlike 3rd-generation customers, they were not resigned to such difficulties). He feels that if FACT had been at least partially demonstrable at the time, the battle may have been won. However, in view of the competitive situation, the war might have been lost.

As it turned out, FACT was not operational until a year after the first COBOL processors. Nevertheless, the language was a substantial contributor to COBOL, and

this is reflected in the later acknowledgments. In contrast, AIMACO as a language was not; it was a derivation of B-0 undertaken as a joint project with UNIVAC, and the contribution here was from the personnel of the Air Materiel Command. The Short Range Committee submitted its next report in 1960 January. It was accepted, subject to editing for "typographical and other minor errors". The editing committee was chaired by Phillips, together with Wegstein and Betty Holberton. The minor work took from January through April, and the COBOL 60 Report was issued by the Government Printing Office in June.

COBOL 60

Despite the inadequacies of the Report, the list of manufacturers announcing or committing to COBOL implementations grew. It was simply the thing to do. However, many qualms were felt about a language defined in large part by example rather than by syntax and semantics, particularly at IBM, whose John Backus had presented his metalinguistic notation (BNF) the previous summer. General Electric's Charlie Katz, another old ALGOLer, warned (in the public announcement of their language GECOM) that while COBOL could be accepted by GECOM, it was not yet developed to the point where unambiguous interpretation was possible.

The official IBM position on COBOL was a critical element for acceptance in the industry. Commercial Translator had been announced for the 7070, 709/90, and 705 III. Barry Gordon was responsible for the compiler implementations. Roy Goldfinger and I were working both within IBM and within the Short Range Committee to reduce the differences between Commercial Translator and COBOL, allowing the former to have extra features, particularly the computational forms of FORTRAN. As a result, the GUIDE organization was told, on 1960 January 27, that IBM would include basic COBOL in Commercial Translator. The February 15 survey by the SHARE organization showed Commercial Translator as IBM's version of COBOL (oddly, IBM claimed only 80% machine independence, Honeywell declined to quote on FACT because it was for a single machine, but Wegstein claimed 100% for COBOL).

At the February 17 meeting of SHARE (XIV), Al Harmon, Manager of Applied Programming (and my superior), said: "It appears that time schedules for achieving a version of COBOL that will be satisfactory for all existing and proposed computers would unduly delay IBM's production of processors for Commercial Translator. We are revising our present Commercial Translator manual to represent our best solution to these problems. Our intentions are to revise the Commercial Translator language to include new developments, both from our own efforts and those of the COBOL committee". My verbiage for the official IBM position, announced in *Datamation* magazine, was "The Commercial Translator is being reworked as nearly in the COBOL spirit as possible . . . to ensure that the end result will be a single workable language for data processing". The accent on *workable* came from the Short Range Committee's reluctance to admit the many demonstrated

logical flaws that we found in their specifications. Joe Cunningham reported, in 1963 March, that when the total cleanup had been made, the syntax was just as definable as ALGOL, but the semantics were prone to ambiguities.

All of this was very noble and elder-statesmanly. The only problem was that there were two versions of Commercial Translator within IBM: the one that Tom Glans, Roy Goldfinger and I specified to merge into COBOL, or perhaps even reconcile to identity; and the other that was written by Barry Gordon, that diverged. Because Gordon was in charge of compiler implementation, our good intention came to naught. Seven pages of differences between the two versions were compiled. I recall trying for free form in the statements in order to avoid punch card limitations, which would be quite handy for terminals today.

At the 6th meeting of CODASYL on 1960 April 7, NCR and General Electric announced their intentions to build COBOL compilers. The latter would make it part of GECOM, which also had algebraic statements like Commercial Translator (antedating PL/I), and a tabular structure facility. Because new manufacturers were asking for participation, Dr. Hopper and I were discharged as advisors, to avoid any appearance of partisanship. In May, Jack Jones of the Air Materiel Command announced the start of an UNIVAC 1105 COBOL. By September, there were 11 manufacturers represented on CODASYL and all had indicated that they would supply compilers (the original six plus Bendix, CDC, GE, NCR and Philco). C.G. Holland-Martin of International Computers and Tabulators had sent a representative to the first CODASYL meeting, and ICT (now ICL) announced a COBOL compiler in October, superseding their own CODEL language.

The internal dissent at IBM kept building, and outside pressures were felt from the user groups. The problem escalated to T.V. Learson, now IBM's Chairman of the Board, who solved it in the style of one who has access to sufficient spendable money. IBM would do *both*, and work toward reconciliation. Accordingly, Al Harmon told the 1960 May 17 meeting of GUIDE that IBM would supply COBOL compilers for the 705 II (without IOCS), the 705 III, 7080, 7070, and 709/90, but declined to give delivery dates. He outlined the two-phase solution of Learson: first a modified Commercial Translator and then a conversion to COBOL. This was most unsatisfactory to GUIDE, which resolved that IBM should stick to its original statement, and that there should be only one compiler per machine, with COBOL an integral part of Commercial Translator.

IBM did not do so, however. Not because it *would* not, but because it *could* not. Roy Goldfinger made an extensive comparison in 1960 July between COBOL and the version of Commercial Translator implemented by Barry Gordon, showing that the original objective was missed by a wide margin. To give IBM management their due, they were honestly chagrined to find that good intentions do not guarantee compatibility in programming languages!



Danny Goldstein of UNIVAC, one of the six members of the Short Range Task Group, prepares to run at the 1960 December 6 compatibility demonstration.

Difficulties persisted. IBM finally announced their COBOL production schedules on 1960 October 1, but it was not until 1962 September that the success of COBOL was sufficiently apparent for Bob Ruthrauff to tell SHARE XIX that "We intend to make COBOL our development language and plan no further development of the Commercial Translator language itself". IBM did not want to put Commercial Translator under the 7090 operating system, but said it would negotiate with diehard users. (Actually, the situation would never have come to this had not the 7090 compiler for Commercial Translator, done by a West Coast group under the direction of Dr. Richard Talmadge, been so good compared to the COBOL compilers existing then.) In 1963 February, SHARE abandoned all hope for Commercial Translator and prepared to switch existing programs over to COBOL via translation routines and some hand work. Honeywell went through much the same difficulties and extra expense; FACT was completed and customers were supported, despite their parallel support of COBOL.

VICTORY?

The participants in the COBOL effort did not disdain publicity. The *New York Times* of 1960 August 26 announced RCA's victory in the "Computer Translating Race", although the language was not full COBOL by any means, nor was the compiler released until December, when RCA and UNIVAC held a joint compatibility demonstration for the public. Both computers operated the test programs identically, UNIVAC II on December 6 and RCA 501 on December 7. Nevertheless, *Datamation* announced that RCA was "making publicity capital out of The Sacred Project" and had "made off with the first publicity marbles". IBM received similar publicity with the COBOL 61 compiler on the 1410.

QUALITY PROBLEMS

In general, the compiling techniques of early COBOL processors were primitive, resulting in very low compilation rates. Navy evaluations reported in 1962 May

showed five compilers ranging from 3 to 11 statements per minute. Measurements in mid-1964 showed a range of 11 to 1000 statements per minute. Hardware did not get that much faster in two years, so we must conclude that we found out how to build compilers better. At this time, it was first noticed how drastically compiling rate varied with the size of store (memory) available, and how much variance there was in compiling cost, which ranged in this study from \$0.23 to \$18.91 per statement. Presently we are in the range of several thousands of statements per minute, and the costs are very much lower.

STANDARDS

Publications via the CODASYL route were COBOL 60, COBOL 61, COBOL 61 Extended, COBOL 65, CODASYL COBOL Journal of Development-69, and CODASYL COBOL Journal of Development-70. The original intent was to update yearly, but this was changed with the advent of formal standardization, in early 1963. The CODASYL COBOL committee concentrated on development, leaving the intermediate and formal standardization to the standards bodies. The European Computer Manufacturers Association (ECMA) set up Technical Committee 6 to deal with COBOL, corresponding to American National Standards Committee X3.4.4, which was responsible for developing a COBOL standard (Howard Bromberg, Chairman). These two committees worked jointly to produce twelve COBOL Information Bulletins.

Under the Brooks Bill, US Public Law 89-306, Federal Information Processing Standards are the three-way responsibility of the National Bureau of Standards (NBS), the Bureau of the Budget (now Office of Management and Budget), and the General Services Administration. The first man in the NBS position was Norman Ream, previously of Lockheed. Short funding at NBS prevented the monitoring and measuring efforts scheduled there originally. These difficulties were surmounted when Ream was appointed Special Assistant to the Secretary of the Navy, for he was imbued with the idea that standards would not be too effective without a means of checking conformity. Many compilers were claimed to be COBOL, but, without a Truth In Packaging law or detection device, the user was sometimes misled. Ream's ingenious solution was to call back to active service Commander Grace Hopper, USNR (Ret) to lead the accelerated effort in COBOL standardization for the Navy. Grace was to return to active duty for six months starting 1967 August 1. Only one thing was wrong with the memo announcing this move: she did not get out in 1968, nor in 1969, nor in 1970, when her orders were reissued to read "indefinite". As a result, the Navy has constructed comprehensive COBOL certifiers that are available to anyone.

By being brought under standardization control, COBOL could profit by being related to other information processing standards. To give an example, CODASYL had an early report on the 1960 January 13 meeting of the American Standards Association, at which Committee X3, on Computers and Information



Chairman Phillips and the compiler builders: Howard Bromberg of RCA on the left, and Dr. Grace Hopper of UNIVAC on the right (at the compatibility demonstration).

Processing, was authorized. Character sets were very prominent in the discussion. FLOW-MATIC had 63 characters available, due to the UNIVAC computer, whereas IBM was limited to 48, due to the punch card set at that time. Furthermore, the collating (ordering) sequences were different between the several equipments, and none of the existing sequences survived in the American Standard Code for Information Interchange (ASCII). Yet COBOL has statements in the language which give different actions depending on the collating sequence of the hardware! For reasons unknown to me, CODASYL chose to ignore this problem, and it is still not resolved.

A COBOL specification has been adopted by the International Standards Organization, which refers to standards as Recommendations. Through the efforts of an international editing committee, and careful arrangements by the standards bodies, it is the same as the American National Standard. ECMA also contributed heavily with a formal description of the syntax of COBOL, which was invaluable in reducing ambiguities and validating constructs.

WHAT NOW?

Here we are, beginning a new decade with a regularized COBOL world in both French and English. One might think that the future would hold no surprises. Theoretically, COBOL could continue to maintain a prominent position in computer usage. There is an organizational structure for development, and several standardization bodies to normalize the changes and additions as they are developed. Practically, however, there are some strong factors opposed to its immortality:

- PL/I, a competing language, has been introduced and has attained substantial usage. PL/I also has development and standardizing bodies, at least in the US and in ECMA. It appears that COBOL programs are translatable to PL/I programs with some difficulty, but not too much more than from FORTRAN II to IV programs.

- Although COBOL now has additions for data communication and data manipulation, they are appended to the same old COBOL structure originated for a uniprogramming environment. As I warned in an address to the 10th Anniversary Meeting of CODASYL, data communication and manipulation are but different aspects of general data movement, and should be unified and common to *all* programming languages that must coexist in the same multiprogramming environment. PL/I, COBOL, and FORTRAN, having different development committees, contain data communication facilities that are not common or similar, and yet they must coexist under a single operating system. As the data base becomes king, there is really no need for more than one data procedure language. The user pays heavily for this language plurality.
- The concept of levels may prove vitiating in a communications world. If interchange is stressed too heavily, the tendency will be to write programs in the lowest level of COBOL, in order to assure the widest usability on a maximum number of computers. This is akin to Gresham's Law, and low-level programs will drive out high-level capability. This would be unfortunate, for those lower levels are really unnecessary. The only (and original) reason for their existence is to permit COBOL compilation on small computers, which is something that small computers are particularly unsuited for. It is far better to use the full power of a high level of COBOL, hook up by communications to a remote compilation service, and get back an object program, the running of which does not tax machine capacity the way compilation does.
- Finally, COBOL and the other languages all possess a serious defect for public usage of data: they carry the data description in the program. For interchange and multiple use of public data, the data must be self-descriptive, and the description must be carried along with the data on the storage medium (i.e., disk or tape). Some corresponding change in the structure of these languages is certain.

However, these technical difficulties should not deter the working user from extensive use of the COBOL language. For every successful user there are still several floundering in the morass of emulation because they just could not bring themselves to break away from machine language. The problems of program transferability are great, and the losses from not using transferable languages such as COBOL run well over \$1 billion each year. We find countless examples of programmers who have moved on, leaving programs that are undecipherable for maintenance or modification. Had they been written in COBOL, they would have had more than zero salvage value.

In my opinion, emulation is a crutch that never lets you get over the handicap. Fight wastage of human resources; COBOL is a great weapon.

Data structuring, program modularity, machine and device independence, code efficiency, and compatibility with popular languages are all vital to major software engineering problems, particularly to those involving computer graphics. The AED (Automated Engineering Design) System, developed over the past ten years by the Air Force-sponsored M.I.T. Computer-Aided Design Project, and now distributed and maintained as Public AED by SofTech, possesses all of these attributes. Because of its versatility, AED has been used in a wide range of applications. It can be compiled to be compatible with FORTRAN, PL/I, or COBOL programs, and is more efficient than any of them for system programming.

The development of user-oriented interactive systems, particularly those using graphic terminals, provides an excellent test of the utility of AED and the over-all "AED Approach" to system building. The AED software component GRAPHYSYS, which provides all the commonly required graphic input/output functions, demonstrates that an efficient device and machine independent interface can be provided between a graphics terminal and an application program.

AUTOMATED ENGINEERING DESIGN (AED) used for graphics

*D. T. Ross and J. W. Brackett
SofTech, Inc.*

INTRODUCTION

A software technology based on software engineering discipline and supported by system building tools and reusable software components is needed for wide industry use. Essential to such a technology is a complete and consistent system programming language applicable to many types of software application systems and a variety of computer hardware systems. Engineering discipline allows software design and implementation tasks to be broken into well-defined and reusable units, adaptable for supporting all applications. These software components must also be portable, so that they can operate on a variety of hardware configurations.

SOFTWARE COMPONENTS

AED supports a software engineering discipline. The AED software components provide a high-level approach to the use of devices, storage, and CPU resources via a two-part interface between the application system and the hardware. The "upper part" of the interface provides a stable support level for the applications, while the "lower part" is adaptable to suit dif-

WHAT'S THE DATE ?

By now our readers will have noticed that this Journal always presents the calendar date in the year-month-day order. This is to conform to:

- ISO Recommendation 2014 - Writing of Calendar Dates in All-Numeric Form
- American National Standard X3.30-1971 (Representation for Calendar Date and Ordinal Date for Information Interchange)
- FIPS PUB 4, Calendar Date (US Government)

Although the American Standard derives from Committee X3, on Computers and Information Processing, the need for a standard method of writing the date is universal. ISO had a special committee, DATCO, for this purpose. Here is how ISO put it in the announcement:

"How to avoid confusion when writing the date

The worldwide membership of ISO (the International Organization for Standardization) has agreed to standardize the manner of writing dates. Thus, the most junior typist has the opportunity of making a contribution to the removal of one unnecessary obstacle to international communication.

How?

By using the universally-agreed system of descending order when writing the date on any letter or document. This rule applies only when an all-numeric form is used, which might lead to confusion.

Provided that the month is clearly spelled out (even in an abbreviated form), one can write the date however one wishes - 1 April 1971, April 1 1971, or 1971 April 1. There is no ambiguity, so no standard is required.

However, when an American writes the date 4-1-1971 to indicate April the first, an Englishman or a Norwegian, for example, would read the date as the fourth of January.

After studying all the implications of the question, an ISO committee of experts recommended the *descending* order, and ISO Recommendation 2014 has now

WK	1975							1976							1977							WK		
	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S			
1								29	30	31	01	02	03	04	03	04	05	06	07	08	09	JAN	1	
2								05	06	07	08	09	10	11	10	11	12	13	14	15	16		2	
3	JAN	30	31	01	02	03	04	05	12	13	14	15	16	17	17	18	19	20	21	22	23		3	
4									19	20	21	22	23	24	24	25	26	27	28	29	30		4	
5									26	27	28	29	30	31	01	02	03	04	05	06		FEB	5	
6									02	03	04	05	06	07	07	08	09	10	11	12	13		6	
7	FEB	03	04	05	06	07	08	09	09	10	11	12	13	14	15	16	17	18	19	20		7		
8									16	17	18	19	20	21	21	22	23	24	25	26	27		8	
9									23	24	25	26	27	28	28	29	30	01	02	03	04		9	
10									01	02	03	04	05	06	07	08	09	10	11	12	13		10	
11									08	09	10	11	12	13	14	15	16	17	18	19	20		11	
12	MAR	10	11	12	13	14	15	16	15	16	17	18	19	20	21	22	23	24	25	26	27		12	
13									22	23	24	25	26	27	28	29	30	31	01	02	03		13	
14									29	30	31	01	02	03	04	05	06	07	08	09	10		14	
15									05	06	07	08	09	10	11	12	13	14	15	16	17		15	
16	APR	07	08	09	10	11	12	13	12	13	14	15	16	17	18	19	20	21	22	23	24		16	
17									19	20	21	22	23	24	25	26	27	28	29	30	01		17	
18									26	27	28	29	30	01	02	03	04	05	06	07	08		18	
19	MAY	28	29	30	01	02	03	04	03	04	05	06	07	08	09	10	11	12	13	14	15		19	
20									10	11	12	13	14	15	16	17	18	19	20	21	22		20	
21									17	18	19	20	21	22	23	24	25	26	27	28	29		21	
22									24	25	26	27	28	29	30	31	01	02	03	04	05		22	
23									31	01	02	03	04	05	06	07	08	09	10	11	12		23	
24	JUN	02	03	04	05	06	07	08	07	08	09	10	11	12	13	14	15	16	17	18	19		24	
25									14	15	16	17	18	19	20	21	22	23	24	25	26		25	
26									21	22	23	24	25	26	27	28	29	30	01	02	03		26	
27									28	29	30	01	02	03	04	05	06	07	08	09	10		27	
28									05	06	07	08	09	10	11	12	13	14	15	16	17		28	
29	JUL	07	08	09	10	11	12	13	12	13	14	15	16	17	18	19	20	21	22	23	24		29	
30									19	20	21	22	23	24	25	26	27	28	29	30	31		30	
31									26	27	28	29	30	31	01	02	03	04	05	06	07		31	
32									02	03	04	05	06	07	08	09	10	11	12	13	14		32	
33	AUG	04	05	06	07	08	09	10	09	10	11	12	13	14	15	16	17	18	19	20	21		33	
34									16	17	18	19	20	21	22	23	24	25	26	27	28		34	
35									23	24	25	26	27	28	29	30	31	01	02	03	04		35	
36									30	31	01	02	03	04	05	06	07	08	09	10	11		36	
37									06	07	08	09	10	11	12	13	14	15	16	17	18		37	
38	SEP	08	09	10	11	12	13	14	13	14	15	16	17	18	19	20	21	22	23	24	25		38	
39									20	21	22	23	24	25	26	27	28	29	30	01	02		39	
40									27	28	29	30	01	02	03	04	05	06	07	08	09		40	
41									04	05	06	07	08	09	10	11	12	13	14	15	16		41	
42	OCT	06	07	08	09	10	11	12	11	12	13	14	15	16	17	18	19	20	21	22	23		42	
43									18	19	20	21	22	23	24	25	26	27	28	29	30		43	
44									25	26	27	28	29	30	31	01	02	03	04	05	06		44	
45									01	02	03	04	05	06	07	08	09	10	11	12	13		45	
46	NOV	03	04	05	06	07	08	09	08	09	10	11	12	13	14	15	16	17	18	19	20		46	
47									15	16	17	18	19	20	21	22	23	24	25	26	27		47	
48									22	23	24	25	26	27	28	29	30	01	02	03	04		48	
49									29	30	01	02	03	04	05	06	07	08	09	10	11		49	
50	DEC	01	02	03	04	05	06	07	06	07	08	09	10	11	12	13	14	15	16	17	18		50	
51									13	14	15	16	17	18	19	20	21	22	23	24	25		51	
52									20	21	22	23	24	25	26	27	28	29	30	31	01		52	
53									27	28	29	30	31	01	02									53

been approved by 25 countries (Austria, Belgium, Canada, Ceylon, France, Germany, Greece, Hungary, India, Italy, Japan, North Korea, South Korea, Netherlands, Poland, Portugal, South Africa, Spain, Sweden, Switzerland, Thailand, U.A.R., United Kingdom, U.S.A., and Yugoslavia). Four countries only (Czechoslovakia, Ireland, Norway, and Iraq) disapproved the proposal - they preferred the ascending order. Four more countries (Australia, Denmark, New Zealand, and Turkey) abstained.

The ISO committee concluded that the advantages of the *descending* order clearly outweighed any disadvantages. In particular, are cited:

- the ease with which the whole date may be treated as a single numeral for the purpose of filing and classification (e.g., for insurance or social security systems);
- arithmetical calculation, particularly in some computer applications;
- the possibility of continuing the order by adding digits for hour-minute-second (see Note 3).

To be precise, ISO recommends that if numbers only are used, the first day of April 1971 should be written: 1971-04-01.

For technical reasons, the hyphen (or the space) is recommended as a separator rather than the point or the stroke.

Of course, a standard is of little use unless it is widely known and understood. That is why ISO now seeks the cooperation of men and women all over the world who prefer order to chaos - and who would not wish to miss an important date because of a misunderstanding.

....and when numbering the weeks

A second document, ISO/R 2015 - Numbering of Weeks, lays down the agreed standard for the numbering of weeks. The calendar week is an important unit for planning and accounting purposes. Delivery dates in purchasing contracts and similar documents are frequently designated by referring to a certain week number. Since methods of numbering the weeks of the year vary from country to country, a uniform system for the numbering of weeks has an increasing importance for international trade and industrial planning.

Briefly, ISO recommends that Monday (rather than Sunday) be regarded as the first day of the week for business and commercial purposes. ISO/R 2015 recommends, too, that the week should always be of seven days and that a week divided by the turn of the year should be attached to the year containing the higher number of days of that week. Thus, for practical purposes, the first week of the year could begin, at one extreme, on December 29, and at the other extreme, on January 4."



For consistency reasons, this Journal has adopted the year-month-day ordering even when the month is spelled out or abbreviated, as well as the 24-hour clock for diurnal time.

Note 1:

X3.30, being primarily computer oriented, has these additional options and constraints:

- The 4 digits of "year" may be reduced to 2 digits for "year of the century", or to 1 digit for "year of the decade". It should be obvious, however, that extreme caution should be exercised in using these options for mechanical processing.
- The 4 digits of "day-month" may be replaced by 3 digits (from 001 through 365 or 366) for "day of the year", thus giving the ordinal date rather than calendar date. Again, caution, although this option is easier for mechanical differencing of two dates.
- No separators are to be used in internal data representation.

Note 2:

FIPS PUB 4, Calendar Date, is available (price - 20 cents US) from either:

- NBS Clearinghouse for Scientific and Technical Information, US Department of Commerce, Springfield, VA 22151, or
- Supt. of Documents
US Government Printing Office
Washington, DC 20402
(refer to SD Catalog No. C 13.52:4)

Note 3:

See Document X3L8/177, 1971 Dec 01, Draft American National Standard for Representations of Local Time of the Day for Information Interchange. It provides standard formats for both the 12-hour and 24-hour timekeeping systems, but only for civil clock time at point of origin. Therefore proper precautions should be taken for interchange, depending upon where the time was recorded, and if it must relate to any other times recorded in different world time zones.

PUBLISHING A JOURNAL WITH THE SI SYSTEM AND MAKING IT WORK

Robert W. Bemer

Honeywell Information Systems

Honeywell is a multinational company, with only 58% of its employees in the US, the nonmetric country. Furthermore, due to the structure including Honeywell BULL (based in France), HIS Italia, and HIS Ltd. (UK), equipment is fabricated in several countries and marketed in many more. Wherever interchangeability for manufacture, compatibility, and maintenance is required, the SI system is used.

As the premier publication of Honeywell Information Systems, the Honeywell Computer Journal is in a unique position. It must reflect the fact that, by necessity, almost all design, manufacture, and documentation is based upon international standards. To be most comprehensible to employees, customers, and others, usage of SI units is mandatory. The Journal itself is in ISO A4 size, a considerable surprise to the US contingent, but very palatable once they learn the reasons. We feel that this standard for paper sizes illustrates splendidly the opportunities for consistency provided by the SI system.

We have chosen to conform rigorously to ISO R1000, wherein the SI system is defined. Some examples:

- Prefixes are limited to those representing 10 to powers that are multiples of 3. Centi- and deci- are forbidden. Perhaps this is easier for computer people, who are accustomed to such scaling in their everyday work (nanosecond circuits have been with us for some time, and we are now beginning to hear about picosecond switching times). This means that we use "nanometre" rather than "angstrom".
- We differ even from the SI components of the *Abbreviations for Terms used in Electronics* of the Society for Technical Communication. We use "metre", not "meter", and this is very acceptable in the US. "Micron" is not allowed as an alternative to "micrometre".
- Only rarely will the old English equivalents be given in addition to SI units, and then in following parentheses. The only justification for this is original design to nominal English units. (We had the odd experience of converting the dimensions of the Intel 1101 circuit chip to SI, only to find that Intel had done the original design to nominal metric dimensions, which our authors had laboriously converted to inches!) We make an extra effort to be readable and attractive, so the reader will not mind doing his share of the work to think SI.

Fortunately, it is easy for us to control SI usage as a by-product of the computerized text processing and setting system (as you see in this copy). Because the Honeywell Computer Journal is set by photocomposition, there is no need for ISO/TC97 N 442, *Representation for SI Units in Systems with Limited Character Sets*. We can set in Japanese Kata Kana if need be.

At present the author's text is entered into computer storage by our own staff, but we expect to get HIS employees to enter their work directly rather than going through the typewriting process. Following this, there is a conversion from the internal form to the form required by the photocomposition system. One product of this conversion is a KWOC (Key Word Out of Context) concordance. This concordance shows every word, numeral, and symbol of the original text in collated order (vertically aligned), together with the text line in which it appeared, and the number of the line for reference.

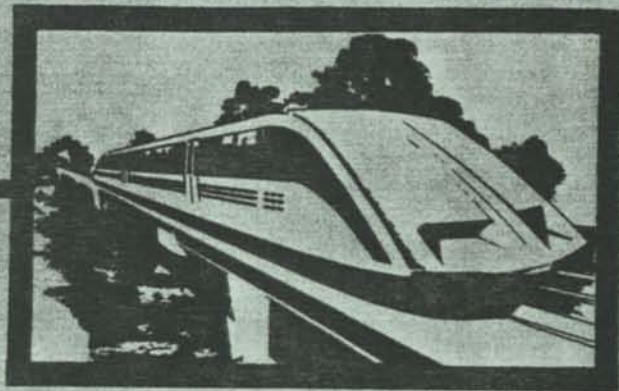
This concordance is a powerful tool for controlling typographic mistakes, style, and conventions (including SI). It even gives a count of words by length, as an aid in alerting to possible difficulties in readability.

Now that we have experience in manual (and visual) use of the concordance, we know how and what to check automatically by computer program. This is now in process of construction. In the manual form, one checks all occurrences of units and abbreviations to see if they are acceptable and, if so, in the correct form (i.e., mandatory space between scalar and abbreviation - 5 nm, not 5nm; correct scalar usage - 0.6 kV, not .6 kV). Looking under the "f's" for "feet", they had better be something to go in shoes, and "inch" should not even be used in "inching along", for such colloquialisms are often mystifying.

Granting that the proper and intelligent conversion and usage of units must still be checked by the editor, these mechanical aids free him to read with understanding, and not just check for typos. Thus a single person can easily control a large amount of usage, and in conformity to the principles of the SI.

For a final illustration, the microfiche copy of the issue, which comes in a back cover pocket of the issue itself, is 105 x 149 mm!

The 19th International Technical Communications Conference



proceedings

May 10-13, 1972
Boston, Mass.



a view of the history of the ISO character code

FROM THE EDITOR

This is another documented personal recollection of the development of basic data processing tools. The character code is perhaps the most fundamental of all such tools. We can program without COBOL, but not without coded representations of our most common symbols for calculation and language. Even the Japanese have found it necessary to represent their most elementary alphabet, Kata Kana, in code suitable for computer processing.

ORIGINS

Encoding of character sets has taken many forms -- semaphore, Morse code for telegraphs, and mechanical linkages for typesetting machines. It did not become permanent and interchangeable until the advent of perforated tape and punch cards, representing the continuous and discrete media. Except for looms, punch cards were used first for counting and statistical applications, requiring at first only a 12-character set -- 10 digits and 2 signs. Perforated tape was used first for message communication, and the 5-track tape offered 60 characters -- 30 encodings had double meaning following Figure Shift and Letter Shift, the other two combinations. This code was due to Baudot, and the assignment of codes to characters was made (for alphabet only) on the basis of letter usage frequency, to conserve electrical energy.

Both perforated tape and cards, being available, replaced manual switches as input devices for computers (cards predominating in the US, perforated tape in Europe and elsewhere). Most early computing work was computational, and the punch card set did not receive much impetus for enlargement until application to business problems.

In the late 1950s it became apparent that code expansion had grown uncontrolled. There were more than 20 variations of the punch card code [129,131], and more than 60 different internal representations in computers [12].

The Baudot Code was not without its variations, although the CCITT (Consultative Committee International for Telegraph and Telephone) Working Alphabet No. 2 was used for interchange throughout much of the world. Even so, the binary positional notation was perturbed by different assignments of bits to the tracks.

*R. W. Bemer
Honeywell Information Systems
Phoenix, AZ, US*

PRESSURES AND MOTIVATIONS

Some early pressures for code expansion came from programming languages. Univac I, the first computer with alphabetic input, had a 51-character set on its printer. The IBM set had mechanical keypunch limitations to 48 characters, mostly business symbols. Starting in 1954, the need for FORTRAN (mathematical and arithmetic) symbols to take the same encodings as these business symbols led to much confusion in installations that did both types of work. Pressures arose for unique encodings [1,3,4,126]. In 1956 September it was decided that the IBM Stretch computer would have a 64-bit word, and that the characters would be represented by 8 bits [19,70]. This decision had much influence on subsequent computer architecture. The Los Alamos group devised its own input character set, and IBM built a special entry typewriter for it [10].

The advent of ALGOL in 1958, with (in theory) 116 single symbols, was the impetus for many proposals. H. S. Bright proposed one such set to the IBM SHARE organization on 58 September 11, as a "point of departure for hardware implementation of the IAL" [4]. Often one to be ahead of his time, his tableau showed the European "one" and the letter "oh" with a curlicue. Yet he objected to SHARE asking IBM for more than 128 characters presentable in media and equipment unless someone was able to prove the need. Others foresaw the need very well, and there was much work going on within IBM at the time, following on the Stretch decision. The 1403 chain printer was being developed, the 240 positions around the chain permitting 2 sets of 120 or 4 sets of 60, in addition to the planned 5 sets of 48 for the traditional IBM set.

EARLY STANDARDIZATION EFFORTS

The real future for computers became clearer in 1958 and 1959. Data processing centers were outgrowing the period when information processing was primarily a local affair, and some connection to communications networks had been made. First to emerge was data interchange, as for Social Security Administration and Internal Revenue Service records. Then came program interchange, more difficult because both symbols and encodings varied.

Several independent standardization projects began at about this time:

- The British Standards Institution (BSI), starting with paper tape and punched cards, gradually moving into the general problem.
- SHARE, seeking to coordinate its IBM equipment.
- The Electronic Industries Association, which also started first from the paper tape aspect, partly due to impetus from Numerical Control.
- The US Department of Defense, for whom the Signal Corps developed the Fielddata Code (which, despite drawbacks, was a great improvement on existing codes, and many of its features are seen in the ISO code). The Fielddata Code still exists as the internal code in the Univac 1107, 1108, and 1110.

IBM was also doing much work in codes, for in 1959 there were 9 different internal codes existing in IBM equipment, and interchange on media was a problem. Development of the 8000 series (precursor of the 360) was under way, and my group (in IBM) was planning the coordination between codes and media representation, following on the Stretch work. A 1959 November memo to V.P. John McPherson outlined a plan for a "single character set to be used by all future computer systems ... the code to be the external storage code as well as internal machine code" (i.e., direct representation in punch cards, as well). It was an 8-bit set.

John Gosden saw this work in the Fall of 1959 and arranged for me to be invited to make a presentation to the BSI codes committee, which occurred in 1960 February. This established a very important contact point with Hugh McGregor Ross (for a while the code was known colloquially in ECMA (European Computer Manufacturers Association) circles as the Bemer-Ross code).

EXPANDING THE JURISDICTION

It was natural that IBM, with its rapid multinational growth, would take a strong interest in the standardization process. Jim Birkenstock, V.P. of Commercial Development, drew up plans to revitalize the Office Equipment Manufacturers Institute and convert it to the Business Equipment Manufacturers Association. A vital consideration in this move was the preparation for BEMA to act as sponsor for the US standards effort in data processing for ASA (American Standards Association, now ANSI). To that end, scopes and programs of work for the various activities were drafted at IBM and presented at the 1960 January 13 meeting convened by ASA.

in the ISO (International Standardization Organization) area, it was Sweden (probably Olle Sturen) that recommended, in late 1959, activities in data processing standards. This led to a Round Table Conference in Geneva on 1961 May 16, which in turn led to the formation of ISO/TC97, Computers and Information Processing, for which the US was assigned the secretariat. Several Working Groups were established, in particular WG B, on Coded Character Sets. ECMA had just been formed in 1961 May, and had liaison representation at the Geneva meeting. Its corresponding body for codes was Technical Committee 1.

US CONTRIBUTIONS

The 1960 January meeting of ASA had led to the formation of Sectional Committee X3, with the same name as TC97. X3 held its first meeting on 1960 August 4, and X3.2 (Codes and Input-Output) its first on October 4, chaired by I.C. Liggett of IBM, who was directing IBM efforts (other work continued in the US until X3.2 jurisdiction was established authoritatively). Membership was from the Federal Government, data processing, and communications interests.

The latter were represented primarily by the Bell System, which was to convert its TWX service from manual to dial operation in 1962. Along with this was a new transmission arrangement that enabled 100 word-per-minute service in addition to the existing 60 wpm service. This meant that the majority of teletypewriters (Model 15, limited to 75 wpm) would have to be replaced. To Bell, and to John Auwaerter of Teletype, this seemed an excellent opportunity to escape from the limitations of 5-level Baudot Code and the 3-row keyboard to a 4-row keyboard more similar to conventional typewriters. The Bell System was not only an active participant, but actually hastened development by adopting conclusions as fast as they were reached and incorporating them in their new designs. It even chose the perforated tape track assignment in advance of that standard, and thus made it rather easy to reach agreement because of the volume of Model 33 and 35 equipment in actual operation.

X3.2 did extensive work on development of criteria for a code. It considered aspects such as collating sequence and natural orderings of subsets, keyboard design considerations, and programming effects such as manipulation of graphics by classes, simpler language scans, and more graphics for clarity of printed output. This was all based upon the assumption of a dense 6-bit code, as used by almost all computers of the time (Stretch was the exception, at 8 bits).

No existing code was found to be satisfactory. The Fielddata Code, as Military Standard 188 [72], was in final stages of adoption in 1959 December, and was offered for consideration. However, it was primarily for communications, although computer equipment was used in the process. For data processing, it had the defect that control codes and graphics were intermingled in 3 of the 4 columns. The alphabet was pushed to the bottom of its columns, leaving no room for additional characters of other alphabets, such as the Scandinavian. The IBM codes had a peculiar and overriding problem - none of them had the same sequence as the standard IBM collating sequence.

THE 7-BIT ERA BEGINS

After trying many months without success to select 64 characters that would satisfy the requirements of both data processing and communications, X3.2 finally settled upon a 7-bit set at its 61 June 7-9 meeting. The alphabet was pushed to the bottom of its second column, as inherited from Fieldata. Nevertheless, the US Department of Defense was beginning to get concerned about conflict with Fieldata. Secretary Macon, in his summary report of the first 9 meetings, suggested that the code be called Fieldata No. 2, to give that credit.

IBM continued to be concerned about aspects of international compatibility. Two sets had been presented at the June 9 meeting with the Department of Defense - one with the alphabet at the bottom, and one with the alphabet starting in the second position, as now (column 1 of Table 1). This second condition was furnished by Ross, for BSI.

At its next meeting X3.2 had to decide between these two proposals. Study of Scandinavian phone books showed the need to have at least 3 extra positions following the 26-letter alphabet. I took this position strongly in the ensuing argument (which, although vociferous, is not indicated in the minutes). Placement of the special symbols was another matter; the traditional IBM collating sequence was specials (S), alphabet (A), digits (D). I had a ready solution - the major special symbols (particularly "minus") were to be taken out of the digit column and placed in the previous column. For ordering purposes, the digit column could then be made to collate higher than the alphabet (for IBM's needs) by passive logic of bit inversion. The specials following the digits had not appeared previously in key fields used by IBM customers, and it did not matter if they were high to the alphabet. The 4-bit subset was thus constructible with a 4-position jog, or displacement. This was also important to IBM, for the packed decimal notation in the new line. On this basis, John Auwaerter and I prepared a presentation that convinced X3.2 to settle the code in its present structure; the basic 7-bit standard was agreed - the alphabet starting in the second position! The draft then stood as in column 2 of Table 1.

I had called a joint meeting of IBM, SHARE, and GUIDE, to regularize the IBM 6-bit set to become the standard BCD Interchange Code [76]. Frequency studies of symbol occurrence had been prepared, particularly from ALGOL programs. The meeting of July 6 produced general agreement on a basic 60-64-character set, which included the two square brackets and the reverse slant, which was chosen to be used in conjunction with "/" to yield 2-character representations for the AND and OR of early ALGOL. This is reflected in the set I proposed to X3.2 on September 18 (column 3). The lower case alphabet was also shown, but for some time this was resisted, lest the communications people need more than the two columns then allotted for control functions.

At the 61 November 8-10 meeting, X3.2 constructed the first formal proposal, X3.2/1 [73] (column 4), with the explanations for its structure and content, including the compatibility with Europe, where the UK standard was at the printers. The arrows were moved up so as to be expendable when replaced by 10 and 11 for pence of the English monetary system of that time.

GOING TRULY INTERNATIONAL

The Americans, less accustomed than now to international standardization work, had their eyes opened at the first ISO/TC97 meeting. At the 1961 Fall Joint Computer Conference I broached the possibility of direct discussions at working level to Philippe Dreyfus of ECMA. This was discussed at the concurrent meeting of X3.2 (December 11-13). Dr. J. Barker of BEMA, as chairman pro tem of X3, was irate with such a lack of national loyalty. Nevertheless, (now) X3.2 Chairman Macon announced that arrangements had been made for Bloom (NCR) and Auwaerter (Teletype) to go to Europe for the period of 62 January 2-22. The resulting frank discussions with BSI and ECMA had very good effect. On January 29, X3.2 received from Hugh McGregor Ross a proposed a compromise that could meet the requirements of ASA, ECMA, BSI and the EEA (Electrical Engineering Association of the UK). Thus the groundwork was laid for the first meeting of TC97 Working Group B on 62 May 4.

H. Feissel of BULL was elected chairman. Present were delegations from France, Germany, Italy, the UK, and the US - plus ECMA and UIC (Union Internationale des Chemins de Fer). Despite the official UK position for its own standard, Ross put forth his compatible proposal. Although Germany wanted the alphabet moved to the first position, there was considerable agreement, and a 6- and 7-bit code were resolved. The latter was in two versions, A and B (columns 5 and 6 of Table 1), differing only in the two control columns. Germany preferred A, all others preferred B.

The great significance of moving the work to the ISO level is evidenced in these proposals, noting how many choices remain fixed after this time. Furthermore, the controls were regularized and grouped to 7 transmission controls, 6 format effectors, and 5 device controls; the improvement from the haphazardness of the previous proposals is quite apparent.

It was noted subsequently that the positions of ":" and ";" were unsatisfactory for pence replacement in sterling. Cooperation was so well established that interchange with "" and "+" was effected by a telephone call to ECMA in Europe, who confirmed its agreement to WG B on June 27.

X3.2, whose written vote on X3.2/1 was all affirmative except IBM and an abstention from ITT, proposed to X3 that the new document, X3.2/4 (column 7), be supported, as these were minor changes. International consistency with Proposal B was stressed, and X3.2 recommended extending the X3 ballot period to September 15, as WG B needed official opinions for the October plenary meeting of TC97.

There followed a difficult period in the US, eventually leading to the first American Standard Code, X3.4-1963.

GETTING AN AMERICAN STANDARD CODE

The original negative vote of IBM on X3.2/1 (at the X3.2 level) was cast by Robert Blue, who had replaced H. J. Smith, Jr., as the IBM representative. Smith was asked, however, to remain as a consultant to X3.2 in recognition of his many fine contributions. This was permitted by IBM, with the explanation that "We have tried to participate ... according to ASA rules, and have allowed our participants to act freely on a

purely technical and professional basis. Since the actions of our participants have been construed as company actions and endorsements, we shall, in the future, insure that these are one and the same .." (Bartelt to Macon on March 2). Accordingly, Blue's negative stated that "IBM studies have shown that the proposed 7-bit code with its contiguous alphabet would be expensive to implement in the present domestic data processing environment oriented to the use of 6-bit processors and punched cards". IBM work on 8-bit codes or the new product line was not mentioned.

William E. Andrus, Jr., IBM Group Director of Standards, explained IBM's negative vote in X3: "After extensive study of the entire problem of code standardization, this investigation, together with the experience gained during our many years in the data processing field, indicates that the benefits derived from the adoption of this standard would not justify the economic burdens involved with its implementation. We submit that the BCD Interchange Code [76] is, and must be formally recognized as, a standard because of its wide use both in punch cards and magnetic tape ... IBM supports the development of standard codes (sic plural) for information interchange, but will not support any proposal for a single standard code. X3.2 should consider the possible interaction of future 8-bit codes on present standardization activities".

ITT Communications Systems also took a negative position. A 62 June 18 paper by L. A. DeRosa said that "The proposed code standard ASCII is considered unsuitable as a standard" (it wasn't Fieldata).

As decisions moved from the technical realm to the administrative levels of X3 and the BEMA Data Processing Group (DPG), support lessened. Friden said it must vote negative because terminal costs would be increased 30% to support ASCII (but presented no data). Many minority position papers were written, and X3.2 made extensive and detailed replies to the (some less than rational) objections. The paper of Blue (IBM) and Ungar (ITT), for example, observed that "positive numbers would collate lower than negative numbers". X3.2 took 2 pages to prove that, no matter what positions were assigned to + and -, it is still impossible to order numerals with a binary sort!

Another objection was that it would cost \$124 million and take 2 years to change just the 465L computers from Fieldata, a position not supported by the owner of those computers - the US Department of Defense. Thomas D. Morris, US Asst. Secretary of Defense (Installations and Logistics) wrote to X3 on 62 August 6, saying that "This office is wholeheartedly in support of the ASA Sectional Committee X3 standards development effort both domestically and internationally". He recognized the differences with, but similarity to, Fieldata in organization and structure.

On 62 July 29, Lin Griffin, now chairing X3.2, informed the Chairman of X3, C. Phillips, that X3.2 had approved (by majority vote) a motion that none of the minority positions nor the negative and abstaining votes were deemed necessary or sufficient reasons to warrant revision of the proposed ASCII [75]. "The subcommittee reaffirms its belief that the proposed standard code is the best code it can develop for information interchange purposes, and has the best chance of acceptance and implementation on a national and international basis".

On August 17 IBM abandoned protection of the 6-bit code and exposed a paper by C. Mackenzie, "The construction of 8-bit coded character sets", to the BEMA DPG, and to X3.2 on August 22. Here it was admitted that although the IBM Standard BCD Interchange Code did *not* have binary collatability embodied, this proposal did. In short, IBM furnished the prototype of the EBCDIC (Extended BCD Interchange Code). X3.2 rejected it by a 7-1-1 (yes-no-abstain) vote. "Since it is based upon a structure which was rejected earlier by X3.2, the subcommittee does not recommend revision or withdrawal of X3.2/4".

IBM then asked for an evaluation of economic impact, which was rejected as having no precedent in the records of ASA. At the September 10 meeting of X3, postponed by Phillips from August 21 because of what he felt was a current lack of consensus, X3 voted to send Document X3.2/13 (containing editorial revisions to meet requirements of X3.3 for data communication, thus changing the AIEE vote to affirmative) to ASA for approval. The vote was 20-4-5, with the incognito vote of the BEMA members allocated 6-2.

At this point one might expect that the standard would go to ASA for processing and publication, but there were several more hurdles. First was the BEMA DPG, sponsor of the X3 work. Andrus of IBM wrote this group on October 11 to say "We find it necessary to continue to abstain from voting on the X3.2 code until our request of August 17" (for the IBM code to be named a de facto standard) "is acted upon ... be assured that if BEMA will not act upon our requests, and if the X3.2 Code is issued as an American standard by ASA, IBM will cooperate, making the code available where the economic disadvantages to IBM customers are not a factor".

On October 31 the BEMA Standards Review Board voted 11-3-3 for forwarding. Nays were from Friden, Addressograph-Multigraph, and Monroe, all of whom were represented on both X3 and X4 (X4, Office Machines, chaired by MacFarlane of IBM, had objected to the code as it might apply to office equipment). Abstentions were from IBM, Moore Business Forms, and Smith-Corona-Marchant. The RCA position was unknown, but computer manufacturers voting yes included Honeywell, Burroughs, NCR, and Univac.

On 62 November 5, Frank White of the Air Transport Association wrote to ask: "Rumor has it that BEMA has turned down the recommended action of X3 relative to the standard code for data processing. Do you plan to provide ... details ... so that we may take under advisement what may be done to achieve effective implementation of the policy we recommended?" Phillips replied that "The BEMA Standards Review Board decided to defer the transmittal of the proposed Standard Code for Information Interchange until something further was done toward achieving a consensus. The Board also recommended that X3.2 take appropriate action to submit de facto standard codes concurrently with the pASCII (proposed ASCII). It is believed that such actions would satisfy the objections raised by the X4 Sectional Committee". He further notified the Joint Steering Committee of X3, X4, and X6 that the "Sponsor requests that X3 and X4 propose a de facto standard or standards for consideration concurrently with proposals for pASCII expressed in media".

col.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
document	X3.2 to DoD	X3.2	RWB	X3.2/1	TC97 WG B A	TC97 WG B B	X3.2/4	ISO #1	STD X3.4 -1963	ISO #2	ISO #3	X3.2.4 /64	CCITT	X3.2 /172	X3.2 /206 #4	ISO DR 1052	X3.2 /384 /411	X3.2	X3.2
date	1961 06 29	1961 09 14	1961 09 18	1961 11 10	1962 05 04	1962 05 04	1962 05 25	1963 01 01	1963 06 17	1963 12 12	1964 05 05	1964 10 20	1964 10 09	1964 12 03	1965 01 21	1966 03 04	1966 05 20	1966 07 15	1966
0/0	NULL	RU	VT	SOM	EOT	SOM	FE ₀				SOH								
0/1			rcv 1 on	m off	EOM	EOA	HT				STX								
0/2			rcv 1 off	RU	EOA	EOM	FF				ETX								
0/3					SOM	EOT													
0/4		WRU			WRU	EOT					ENQ								
0/5		motor 1 stop			RU						ACK								
0/6		feed out			BELL														
0/7		HT																	
0/8		feed out																	
0/9		tmt start			FF														
0/10		BELL			VT														
0/11		rec 1 on			LF														
0/12		rec 1 off			HT														
0/13		rec 2 off			CR														
0/14		rec 2 on			SO														
0/15		tmt stop			SI														
1/0																			
1/1					DC ₄ stop	DC ₀													
1/2					DC ₃	DC ₁													
1/3					DC ₂	DC ₂													
1/4					DC ₁	DC ₃ stop													
1/5					ERROR	ERROR													
1/6					SYNC	DC ₄ stop													
1/7					EOB	ERROR													
1/8					S ₅ LEM	EOB													
1/9					S ₄	S ₀													
1/10					S ₃	S ₇													
1/11					S ₂														
1/12					S ₁														
1/13					S ₀														
1/14																			
1/15																			

Assignments Permanent

Steps in the Evolution of the ISO Code

col.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
docu- ment	X3.2 to DoD	X3.2	RWB	X3.2/1	TC97 WG B	TC97 WG B	X3.2/4	ISO #1	STD X3.4 -1963	ISO #2	ISO #3	X3.2.4 /64	CCITT	X3.2 /172	X3.2 /206 #4	ISO DR /384	X3.2 /411			
date	1961 06 29	1961 09 14	1961 09 18	1961 11 10	1962 05 04	1962 05 04	1962 05 25	1963 01 17	1963 06 17	1963 12 12	1964 05 05	1964 10 20	1964 10 09	1964 12 03	1965 01 21	1965 03 03	1966 04 04	1966 05 20	1966 07 15	
2/0	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
2/1	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!
2/2	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"
2/3	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
2/4	\$	<	>	↑	#	\$	↑	CS	\$	CS	CS ₁	\$	⌘	\$	\$	CS ₁	\$	CS ₁	\$	\$
2/5	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!	!
2/6	&	%	%	↑	%	%	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2/7	&	&	&	↑	&	&	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2/8	(((↑	((↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2/9)))	↑))	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2/10	:	:	:	↑	:	:	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2/11	;	;	;	↑	;	;	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2/12	*	*	*	↑	*	*	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2/13	/	/	/	↑	/	/	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2/14	.	.	.	↑	.	.	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2/15	,	,	,	↑	,	,	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
3/10	+	+	+	↑	+	+	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
3/11	}	}	}	↑	}	}	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
3/12	<	<	<	↑	<	<	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
3/13	>	>	>	↑	>	>	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
3/14	X	X	X	↑	X	X	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
3/15				↑			↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
4/0	=	=	=	↑	=	=	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
5/11	#	#	#	↑	#	#	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
5/12	#	#	#	↑	#	#	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
5/13	%	%	%	↑	%	%	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
5/14	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
5/15	↓	↓	↓	↑	↓	↓	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
6/0																				
7/11	ansr back	ansr back	ansr back	ACK																
7/12																				
7/13																				
7/14	ESC																			
7/15	DEL																			

Assignments
Permanent

Phillips then wrote to Roger Gay of the ASA on December 5, saying "The problem here is one of attempting to introduce a new and nonexistent code convention as the first and only standard in the character code field of information processing systems. Such an action would render widespread de facto code practices "nonstandard" and obsolete, with potential economic and technological hardships to at least the suppliers of equipment affected by such standardization". X3 was informed of the sponsor's actions as of this date.

X3.2 was as distressed as X3. At its 62 December 5-7 meeting, a motion passed 5-2-3 that "X3.2 is of the opinion that if any existing coded character set is to be considered a de facto standard, many must be considered, and that, if any one is found allowable jointly with the pASCII, many will be found allowable ... Analysis of documents [12,129, 131] supports this position". Griffin then prepared a summary memo for the Joint Steering Committee, dated December 18.

Gay of ASA replied to Phillips on 62 December 21: "It is my understanding that ... each of the existing codes had serious shortcomings which precluded its being recommended ... the Constitution of the ASA specifically prohibits the promulgation of conflicting standards ... In X3, however, there was never a formal recognition to any of the 'de facto' codes, and the very insertion of such codes in the American Standard may promote their continued use rather than urge their discontinuance. In other words, you would be maintaining the status quo which has been greatly criticized in the past". He noted that James Birkenstock, V.P. of IBM, had said that "IBM believed industry as a whole stood ready to cooperate in coming up with a single language", and William Suchors, V.P. of Remington Rand "too felt that all manufacturers could harmonize their thinking in the new project". He then reminded BEMA that "At all levels, the views of the sponsor are taken into account. However, the sponsor has no veto power". He advised calling a meeting, and that "if the decision of X3 is that de facto codes should *not* be added, BEMA should transmit the pASCII without delay to ASA".

The Underwood Corporation accented the anomaly of the situation by writing to ASA to say that they were against inclusion of de facto standards but, if there were to be any, here was theirs! Others took frontal action. Paul Riley, Deputy Asst. Secretary of Defense, in his 63 January 18 letter to Phillips, urged approval of ASCII, particularly to put the US in a better position at ISO. G. D. Osborn, of the Executive Office of the President, Budget, wrote X3 members saying that it was important that the proposed code be forwarded to ASA and adopted now.

At the 63 January 24 meeting of X3 it was voted, by a 2 to 1 majority, to transmit the pASCII directly to ASA without recommendation of the sponsor, the first (and perhaps only) time such an action had occurred in the history of the American Standards Association!

In a final attempt, Phillips wrote to Gay on February 8, saying "It appears that a part of the action ... of October 31 was in conflict with this interpretation of ASA regulations ... the sponsor's new recommendation on the pASCII was developed as ... The sponsor accepts the principle of a single

code for information interchange. This code will be universally useful only when it can be adapted to the common methods of machine-to-machine communication, i.e., paper tape, punched cards, and magnetic tape. The sponsor, therefore, believes that submission of the pASCII to ASA is premature". This position, adopted by a 12-3 vote of the BEMA Standards Review Board, was ineffective, however. X3.4-1963 [79](column 9) was approved by ASA on June 17, after a hearing by its Miscellaneous Standards Board.

The action was newsworthy. Ted Merrill of Business Week Magazine made a significant point [21] when he said that "ironing the kinks out of pASCII took about \$3 million worth of man-hours". He noted that AT&T was one of the earliest to gamble, with its new Teletype line [29], and that "some equipment manufacturers, notably IBM, agree the basic code is a good one, but would like to wait and see how it will be standardized for punched cards before they give their stamp of total approval". He suggested that "many computers will undoubtedly use it as their own internal language to avoid the translation problem altogether". Adoption of the ASCII was the occasion for a number of other articles publicizing it [24,25,26].

THE ERA OF COOPERATION AND GOODWILL

A special issue of IBM's Data Processor [22] was devoted to describing the adoption of ASCII and its import. "To support ASCII ... IBM plans to provide whatever means are practical to meet customer needs for using the standard code as soon as possible, after media standardization is approved by the American Standards Association. Because standardization is so important, we have already begun to implement ASCII ... We as a corporation are determined to move ahead with ASCII at the most rapid pace possible so that our customers can gain the benefits of standardization across the industry".

Harry Cantrell, the General Electric representative on X3, reported that "IBM's activity in this standards work, its position papers, and its votes on various questions all point to a clear IBM intention to announce ASA-standard code equipment at an early date ... we can expect IBM's forthcoming 8000 line to be ASA compatible, featuring 8-bit bytes internally".

This heartening face of cooperation was welcomed by everyone, for ASCII was consonant in every aspect with the First Draft ISO Recommendation of 63 January [77](column 8). The ISO Draft did, however, maintain the 6-bit set with equal status, which was reflected in the adoption of ECMA-1, a 6-bit set [78] in March.

TC97/SC2 held its next meeting on 63 October, at which time it was decided to add the lower case alphabet. The Second Draft Recommendation [80] (column 10) gave more consideration to requirements for international alphabets; thus the single quote of the US gave way to accent acute, and the arrows fell to accent grave and circumflex, while underline was added. Format Effector 0 was definitely made Backspace, completing the necessary preparations for diacritical marks. One of the resolutions at this meeting contained the unanimous recommendation that this code be adapted for Numerical Control usage.

In preparation for the 3rd meeting of SC2, X3.2 was advised that it was wished to have the "at" symbol as an additional national use position, which implied a trade with underline. This was tentatively agreed. Also proposed was the replacement of RU by ACK, and ERR with NAK.

SC2 meeting No. 3 was held in New York City on 1964 May 12-15. It was relatively uneventful, except for the presence of a Japanese delegation of five, headed by Professor H. Wada. All of the proposed changes were approved, and ESC was moved in position. The revised proposal was to be circulated as the 3rd Draft [81](column 11) and sent to CCITT. Ad hoc groups were formed for implementation in media, but SC2 would not accept the bit-to-track assignment proposed by the US for magnetic tape, because they had not done any work on it (an 8-bit world was just beginning to seem real, with the announcement of the IBM 360).

The liaison with CCITT was significant. Its Study Group Special A had been considering a new alphabet since its 1958 meeting in Warsaw. At the 63 May meeting in Geneva, CCITT endorsed the principle of the 7-bit code for any new telegraph alphabet, and expressed general but preliminary agreement with the ISO work. It further requested the placement of the lower case alphabet in the unassigned area.

To match the 3rd Draft Proposal, X3.2 adopted Document X3.2.4/64 [83](column 12) on October 20, not knowing that CCITT had met in Moscow on 64 October 6-9 to study the ISO proposal as a working alphabet. CCITT made certain other changes for natural languages, replacing reverse slash by tilde and adding a universal currency symbol (column 13). At the request of the USSR (which did not participate in TC97/SC2), accent grave was interchanged with underline to accommodate the 31-character Cyrillic alphabet [96]. When advised of this, X3.2 accommodated with Document X3.2/172 [84] (column 14). In these last two revisions, however, added symbols for programming languages may be seen. In revising ASCII to remove all controls from column 7, Task Group X3.2.4 added (for the first time) the two braces, a vertical line, and a Logical Not symbol. The latter two were intended to accommodate the Logical Or and Not functions of PL/I. During the discussion, John Windhorst of CDC urged strongly that these two symbols be put somewhere in columns 2 through 5, in order to keep the PL/I 60-character subset within the dense graphic subset of 64 characters. His political wisdom did not prevail, and this led later to the unfortunate usage of exclamation point and circumflex for these functions, thus complicating an otherwise clean mapping to EBCDIC.

X3.2 being ever accommodating, X3.2/172 was superseded by Document X3.2/206 [86] (column 15) to match the 3-character symbol limitation of the new military standard [85]. This document carried an excellent summary of the various changes, with their justifications. However, it also carried a note that caused Univac to threaten a negative vote - "The resultant structure of specials (S), digits (D), and alphabetic (A) does not conform to the most prevalent collating convention (S-A-D) because of other more demanding code requirements". This despite my solution for passive logic in 1962, and despite the fact that it is impossible to provide satisfactory orderings of upper and lower case characters by simple binary comparison [47].

THE DARK AGES AGAIN

The position of IBM was a most important factor for progress of a standard code, and the System 360 was crucial to IBM's position. It was designed to handle both the Extended BCD Code [105] (for upward compatibility of much former equipment) and the eventual ASCII. However, the resistance in X3 and in ECMA to an 8-bit code, together with the fact that the ASCII printer and card reader were not ready when 360 announcement time neared, led to the decision to make EBCDIC the primary code. It was reasoned that ASCII could wait until the matter was settled, at which time the software would be modified slightly, the P-bit switched to ASCII internal mode, and everything would be fine.

Unfortunately, the software for the 360 was constructed by thousands of programmers, with great and unexpected difficulties, and with considerable lack of controls. As a result, the nearly \$300 million worth of software (at first delivery) was filled with coding that depended upon the EBCDIC representation to work, and would not work with any other! Dr. Frederick Brooks, one of the chief designers of the IBM 360, informed me that IBM indeed made an estimate of how much it would cost to provide a reworked set of software to run under ASCII. The figure was \$5 million, actually negligible compared to the base cost. However, IBM made the decision not to take that action, and from this time the worldwide position of IBM hardened to "any code as long as it is ours".

On 64 October 16, C.E.Mackenzie explained to GUIDE the plans of IBM for implementing ASCII in System 360. He stressed the profound difference between supporting the development of a standard and supporting the standard itself, such as actually implementing it in hardware and software. "IBM feels that its primary function is to supply equipment to meet the needs of the marketplace ... IBM will always make a businesslike decision ... the questionnaire distributed to all members of GUIDE at this meeting (and previously to SHARE) is one of the tools used to determine customer needs in the area of information interchange". He noted that the changes and differences between BCD and EBCDIC "should not provide any serious problems for IBM customers", and that "IBM has been unable to determine any appreciable customer needs for ASCII on magnetic tape, or on punched cards, or on perforated tape as input/output for a computer". He related the low need for data communication with ASCII.

In its 360 ESSG Information Letter No. 17 (64 October 14), IBM said that the choice between the two codes "is determined by a mode bit ... a sharp difference between the two codes is the collating sequence. The EBCDIC sequence is consistent with that of previous systems and is therefore largely (sic) compatible with that of our customers' files. The natural ASCII sequence, on the other hand, would place the numbers (sic) before the letters (not yet definitive)".

On this basis it was perhaps strange that it was J. Haddad of IBM who moved, at the 65 February 4 meeting, that X3 support the revised ASCII [86] for concurrent letter ballot and publication. Meanwhile, the Fourth ISO Draft Proposal [87](column 16), as influenced by CCITT, was being circulated as of March.

IBM gave X3 the results of its user surveys on April 22. Nevertheless the June 10 vote for prASCII (proposed revised) stood at 31-0-0. IBM's affirmative ballot on the ISO proposal suggested the vertical line as logical OR, and the tilde as logical NOT. Auwaerter said that this had not been discussed in X3.2, as the comments arrived after all other work was done, and commitments made to the other countries. The US vote on the ISO Draft Proposal was affirmative, unanimously. At the 65 September 16 meeting, X3 voted 24-0-2 to forward the prASCII to ASA, encouraging the US Dept. of Commerce and the General Services Administration to solicit views on making ASCII a Federal standard under the provisions of the Brooks Bill, Public Law 89-306.

The voices of programmers grew louder, inversely with their understanding. Pressures for the two odd characters of the PL/I programming language continued, worldwide, and other pressures existed to put the "at" symbol back in the 64-character subset. Despite the express intent of many years, and rigorous interpretation of the meaning of "interchange code", one B. R. Faden proposed to SHARE, the IBM user group, that it should take a position with ASA for a study of an internal binary code. He said that "It would be a national calamity, truly national and truly calamitous, if ASCII is established as a standard for this purpose". This did not, however, prevent minicomputer designers from using ASCII as an internal code, and it was also chosen for the NCR Century series, the CDC 1800, and the Bull-GE 140.

A DRAFT RECOMMENDATION

TC97/SC2 held its 3rd meeting in Paris on 66 April 25-29, a momentous one. Japan presented Document 163 on extension to non-Latin alphabets, both 7-bit and an 8-bit superset [see 103]; USSR usage was reported as SO to Cyrillic, and SI to Latin [96]. To ensure full agreement, a joint meeting with CCITT, Working Group - Alphabet, was held concurrently. Commercial "at" and accent grave were once more interchanged, and the currency sign problem was alleviated, although not solved permanently, by assigning the pound sterling symbol to 2/3, with the number sign permissible for those countries that did not need it. SS (special sequence) was replaced by SUB (substitute character), and the overline (which could serve as a tilde) was put in 7/14. This produced ISO Draft Recommendation 1052 [90](column 17), for voting under accelerated procedures.

The US position was not stabilized by these actions. Knowing in advance of the proposal to interchange "at" and accent grave, the Information processing Systems Standards Board of ASA ordered suspension of printing of X3.2/206 as Standard X3.4-1965. The doors were in fact reopened, and at the next X3 meeting SHARE presented its views that the rASCII did not meet the pressing needs of the programming community, which was agreed by X3.4 also. However, X3.2 reported that it had listened to these arguments and still approved X3.2/384 (column 18), reflecting the changes of the Paris meeting. Following some editorial changes to allow programming people to stylize certain graphics for the logical operations, this document was sent for 30-day letter ballot.

Argument persisted, and at its 66 December 14 meeting X3.2 approved, in desperation, a modification to stylize the vertical bar by splitting it in two, making X3.2/411 (column 19). One of its members, Doug Kerr of Bell Telephone Labs, discovered in his research that, according to the encyclopaedia, the ASCII "are those inhabitants of the globe, which at certain times of the year, have no shadow ... such are the inhabitants of the torrid zone; by reason the sun is sometimes vertical to them". He observed that inhabitants of the parallel 41° 48' N (running through such cities as ... Poughkeepsie, NY) are never ASCII. Yet when balloting closed, both IBM and API (American Petroleum Institute), carrying the campaign for SHARE, had voted affirmative. At the next X3 meeting it was forwarded to ASA by 33-0-2.

At TC97/SC2's 4th meeting, again in Paris, the vote on DR 1052 stood at 10-3 (2 not received) for the P (Principal) members of TC97. Spain voted negative because the tilde was in a national usage position, without full rank! The vote of the full ISO membership was 13-3-6 (32 not received). So it was sent to the ISO Council, with only Japan voting negative because of the dual assignment of pound sterling and number sign (the currency symbol remains a thorn to this day). The equivalent wording about stylizing for programmers was included. Thus it was a quiet meeting. For 15 resolutions times 10 countries there were 147 ayes, 1 nay, and 2 abstentions.

AN INTERNATIONAL STANDARD

On 67 December 22 the ISO accepted DR 1052 by a 12-1 vote (Ireland opposing), and it has been published as ISO Recommendation 646 [95]. It is entirely consistent with CCITT Working Alphabet No. 5. For the US it matches X3.4-1967 and -1968 [97] (put in force as a Federal Government standard by President Johnson). For Japan it is consistent with JISCII [103]. Many other countries have their national standards derived from it.

Primary emphasis today is not on changing the code. Work is in progress to propose *part* of an additional 128 positions to bring it to 256. This work originates primarily from the fact that the computer world is now predominantly using 8-bit characters on 9-track magnetic tape and other media. This work has been designated "expansion".

Conversely, even more effort has gone to "extension", which is the regularizing of reaching alternate 7-bit and even 8-bit sets via ESCape sequences and their registration. This is to make a harmonized structure wherein a multitude of important non-Latin symbol sets can be represented - Kata Kana, Kanji, Cyrillic, Arabic, meteorological symbols, bibliographic symbols, typesetting symbols and fonts - to mention only a few. It is this work that will demonstrate the vast significance of the establishment of an international standard code.

Note: In the References that follow, the boldface CR followed by a number indicates that the number identifies the review of the referenced article in Computing Reviews, a publication of the Association for Computing Machinery.

REFERENCES

Code (Articles and Papers)

1. A.Vandenburg, "The Lincoln keyboard - a typewriter keyboard designed for computer input flexibility", *Commun. ACM* **1**, No. 7, 4 (1958 Jul).
2. R.F.Johnston, "Character representation and storage systems", *Proc. Canadian Conf. for Computing and Data Processing*, 120-135 (1958).
3. M.D.Grems, "A symbol coder for automatic documenting", *Comput. News*, No. 147, 9-18 (1959 Apr 15) and No. 148, 15-19 (1959 May 01).
4. H.S.Bright, "A proposed 64-character alphabet for hardware implementation of the IAL reference language", *Commun. ACM* **2**, No. 5, 6-9 (1959 May).
5. US Army Signal Corps, "Fielddata equipment intercommunication characteristics", 1959 Aug (revision of 1959 April).
6. J.Green, et al, "Remarks on ALGOL and symbol manipulation", *Commun. ACM* **2**, No. 9, 25-27 (1959 Sep).
CR 00028
7. S.M.Newman, R.W.Swanson, K.C.Knowlton, "A notation system for transliterating technical and scientific texts for use in data processing systems", US Pat. Off. R&D Report No. 15, 1959.
8. R.W.Bemer, "A proposal for character code compatibility", *Commun. ACM* **3**, No. 2, 71-72 (1960 Feb).
CR 00320
[The ESCape character]
9. W.F.Luebbert, "Information handling and processing in large communication systems", Tech. Rep. 099-1, Stanford Electronic Labs, Stanford U. (1960 Jul 11).
10. E.A.Voorhees, "Some thoughts on reconciling various character set proposals", *Commun. ACM* **3**, No. 7, 408-409 (1960 Jul).
[the 133-character set of the Los Alamos STRETCH computer]
11. H.J.Smith, Jr., "A short study of notation efficiency", *Commun. ACM* **3**, No. 8, 468-473 (1960 Aug).
[presages the 8-bit byte of the IBM 360]
12. R.W.Bemer, F.A.Williams, "A survey of coded character representation", *Commun. ACM* **3**, No. 12, 639-641 (1960 Dec).
CR 00639
13. H.McG.Ross, "Considerations in choosing a character code for computers and punched tapes", *The Computer J.* **3**, No. 4, 202-210 (1961 Jan).
14. R.W.Bemer, H.J.Smith, Jr., F.A.Williams, "Design of an improved transmission/data processing code", *Commun. ACM* **4**, No. 5, 212-217, 225 (1961 May).
15. A.A.Markov, "On alphabet coding", *Doklady Akad. Nauk* **139**, No. 3, 560-561 (1961 Jul).
CR 03662 (null)
16. R.R.Shaw, "Parameters for machine handling of alphabetic information", *Amer. Soc.* **13**, No. 3, 267-269 (1962 Jul).
CR 03925
[claims 6-bit codes are insufficient, need 11-bit. No exposure to extension procedures]
17. R.F.Clippinger, "Data processing standards", Computer Applications Symposium, Armour Res. Found., 1962, 176-181.
18. C.A.Phillips, R.E.Utman, "Standardization in computers and information processing", *Proc. Fall Joint Comput. Conf.*, 1962, 177-183
19. R.W.Bemer, W.W.Buchholz, "Character set", Chapter 6, *Planning a Computer System*, McGraw-Hill, 1962, 60-74.
20. H.McG.Ross, "Current positions on standards work relating to computers", *The Comput. Bull.* **6**, No. 4, 133-135 (1963 Mar).
CR 04658
[says First ISO draft primarily a synthesis of BSI and ASA proposals]
21. T.Merrill, "Versatile passkey to communication", *Business Week Magazine*, 1963 Jun 15, 128-131.
22. -, "New ASA standard offers coding uniformity for data interchange", Special Issue, *IBM Data Processor* **7**, No. 7, 12 pp. (1963 Aug 07).
23. "Code standardization", Report for Aerospacecom, Deputy for Communication Systems, by ITT Commun. and Systems, 1963 Aug 15.
24. R.W.Bemer, "The American standard code for information interchange", Parts 1 and 2, *Datamation* **9**, No. 8, 32-36 (1963 Aug) and **9**, No. 9, 39-44 (1963 Sep).
CR 04910
25. L.L.Griffin, et al, "Uniform machine language to speed communications", *The Magazine of Standards (ASA)* **34**, No. 8, 235-239 (1963 Aug).
26. E.H.Clamons, "At last - a standard code for information systems", *Control Engineering* **10**, No. 10, 91-94 (1963 Oct).
27. -, "American standard code for information interchange", *Commun. ACM* **6**, No. 8, 422-426 (1963 Aug).
CR 04943
["no collating sequence is specified"]
28. -, "ALCOR group representation of ALGOL symbols", *Commun. ACM* **6**, No. 10, 597-599 (1963 Oct).
CR 05697
29. J.F.Auwaerter, "A new standard code for teletypewriters", *Bell Laboratories Record*, 1963 Nov, 395-400.
30. -, "Preliminary standards: data processing, representation of ALGOL symbols", *Math. Tech. Wirts.* **3**, No. 10, 117-119 (1963).
CR 05362
[German DIN standard for cards and 5-track perforated tape]
31. -, "Report on CCITT data communications study group meeting", *Commun. ACM* **7**, No. 3, 152 (1964 Mar).
32. C.J.Shaw, "On declaring arbitrarily coded alphabets", *Commun. ACM* **7**, No. 5, 288-290 (1964 May).
CR 06672
[notes the inability of FORTRAN and ALGOL to handle alphabetic data, while COBOL is limited to a single machine-dependent alphabet. The advent of ASCII should give impetus to ability to declare a specific alphabet and/or collating sequence]
33. E.Bukstein, "Binary computer codes and ASCII", *Electronics World*, 1964 Jul, 28-29.
34. V.E.Henriques, "Current position and future of EDP standards", *Proc. ACM 19th Natl. Conf.*, 2 pp, 1964 Aug.
CR 07095
35. V.E.Henriques, "The urgent quest for standards", *Business Autom.* **11**, No. 9, 22-27 (1964 Sep).
CR 08438
36. -, "The BCS standards advisory committee", *The Comput. Bull.* **8**, No. 2, 61-63 (1964 Sep).
37. E.Morenoff, J.B.McLean, L.Odell, "IPC, a coded character set for information processing", Rome Air Dev. Center, Griffiss AFB, AD-608916, 1964 Oct 18, 18 pp.
[8-bit set, subsettable to 7, 6, 5, 4].
38. H.McG.Ross, "The I.S.O. character code", *The Comput. J.* **7**, No. 3, 197-202 (1964 Oct).
CR 07488
39. D.W.Barron, "The ISO character code", *The Comput. J.* **7**, No. 4, 281 (1965 Jan).
CR 11206
[A programmer comments that "standardization of character codes is not as important as sometimes made out. Computers can translate, can't they?]
40. -, "ACM opposes acceptance of EIA standard RS-244 as an American standard on a par with ASCII", *Commun. ACM* **8**, No. 2, 102 (1965 Feb).
CR 08543
41. -, "Proposed revised American standard code for information interchange", *Commun. ACM* **8**, No. 4, 207-214 (1965 Apr).
CR 07784
[X3.2/206, 1965 Jan 21 - see ref. 86].
42. H.Zemanek, "Alphabete und Codes", *Elektronische Rechenanlagen* **7**, No. 5, 239-258 (1965 Oct)
43. W.S.Crosby, "Note on an ASCII-octal code table", *Commun. ACM* **8**, No. 10, 616 (1965 Oct).
CR 08765 (null)
44. L.Davidson, "A pushbutton telephone for alphameric input", *Datamation* **12**, No. 4, 27-30 (1966 Apr).
CR 12922

45. R.W.Bemer, "Information processing - a bit expensive?", Forum Editorial, *Datamation* 12, No. 6, 140 (1966 Jun).
46. -, "Code extension in ASCII (an ASA tutorial)", *Commun. ACM* 9, No. 10, 758-762 (1966 Oct).
47. R.W.Bemer, "ESC facility in ASCII", letter, *Commun. ACM* 10, No. 4, 282 (1967 Apr).
48. M.E.Stevens, J.L.Little, "Automatic typographic-quality typesetting techniques: a state-of-the-art review", *NBS Monograph* 99, 98 pp, 1967 Apr.
CR 13088
["discusses problems of character sets that are adequate, computer programming, keyboard design"]
49. H.Zemanek, "Gedanken zu einem 8-bit-code", *Elektronische Rechenanlagen* 9, 1967 Apr, 65-67.
50. H.Berndt, G.Haberzettl, "Der internationale 7-bit-code", *Elektronische Rechenanlagen* 9, 1967 Apr, 68-73.
51. M.H.Lewin, "Portable electronic keyboard for computer input by telephone", *IEEE Trans. ECU3-16*, No. 3, 332-334 (1967 Jun).
CR 12924
["generates all 128 codes"]
52. D.L.A.Barber, D.O.Claydon, "A keyboard layout for use with the ISO seven bit code", *Comput. Bull.* 11, No. 1, 66-67 (1967 Jun).
53. P.B.Goodstat, "USASCII, What's it all about", *Data Processing Magazine*, 1967 Jun, 20-24.
54. A.M.Bondarev, "Input and output devices for electronic computers", AD-662762, 67 Jul 28, 8 pp.
[translation of *Standartizatsiya* 3, 1964; references the USSR standard GOST-10525-63].
55. R.W.Bemer, "Towards standards for handwritten zero and oh", *Commun. ACM* 10, No. 8, 513-518 (1967 Aug).
CR 14629
56. -, "Proposed USA standard - magnetic tape labels for information interchange", *Commun. ACM* 10, No. 11, 737-743 (1967 Nov).
CR 13712
57. -, "Proposed USA standard - general purpose alphanumeric keyboard arrangement for information interchange", *Commun. ACM* 11, No. 2, 126-129 (1968 Feb).
58. D.V.Black, "Creation of computer input in an expanded character set", *J. Library Autom.* 1, No. 2, 110-120 (1968 Jun).
CR 15755
["101 graphics - less detailed and complex than the recently established MARC II standard"]
59. J.N.Bairstow, "USASCII: Federal Government action on standards challenges computer manufacturers", *Control Engineering*, 1968 Aug, 84-87.
60. P.Hirsch, "Release due for Federal directive on ASCII", *News Scene, Datamation* 14, No. 10, 69 (1968 Oct).
61. -, "Correspondences of 8-bit and Hollerith codes for computer environments - a USASI tutorial", *Commun. ACM* 11, No. 11, 783-789 (1968 Nov).
CR 16119
[reconciles 8-bit ASCII with 8-bit EBCDIC by mapping each to the other via Hollerith assignments]
62. -, "Proposed USA standard - code extension procedures for information interchange", *Commun. ACM* 11, No. 12, 849-852 (1968 Dec).
63. -, "Standards in information processing", *Elektronische Datenverarbeitung* 10, No. 9, 458-459 (1968).
CR 17173
64. G.E.Jones, "The impact of standards", *Computers and Automation* 18, No. 5, 38-39 (1969 May).
CR 17385
["development expense is substantial ... standardization should be preceded by clear statement of need, and de facto practices should be incorporated wherever possible"]
65. J.L.Little, "Some evolving conventions and standards for character information coded in six, seven, and eight bits", *Natl. Bur. Stds.*, NBS-TN-478, 1969 May, 33 pp.
66. C.E.Cohn, "Reading ASCII tape on a BCD computer", *Software Age*, 1969 Jul.
67. R.W.Bemer, "Escape to reality", Forum Editorial, *Datamation* 15, No. 8, 239-240 (1969 Aug).
68. J.P.Ancona, S.M.Garland, J.J.Tropsa, "At last: a standard for keyboards", *Datamation* 17, No. 5, 32-36 (1971 Mar 01).

Code (Drafts and Standards)

69. -, "Fielddata equipment intercommunication characteristics", Memorandum for Director, Data Proc. Facil. Div., Commun. Dept., USASRD, 1959 Apr (rev. 1959 Aug 01).
70. R.W.Bemer, W.W.Buchholz, "An extended character set standard", IBM Tech. Publ. TR 00.721, 1960 Jun (revision of TR 00.18000.705, 1960 Jan).
CR 00813
[as in Stretch].
71. Electronic Industries Assoc., "Basic character set code, tentative standards proposal", TR24.4 Doc. 7233 (1960 May).
[language and format for punched and magnetic tapes, incorporating ESC].
72. US Dept. of Defense, "Fielddata code", in MIL-STD-188A, 1960 Jun 21.
73. X3.2/1, "Proposed American standard code for information interchange", 1961 Nov 10.
74. ISO/TC97/WG B(Sec.-5)16, 1962 Jun.
[proposals A and B].
75. X3.2/4, Proposed American standard code for information interchange", 1962 May 25.
[to meet AIEE requirements].
76. IBM Document J22-6690, "Standard BCD interchange code", 1962. ["defines for the IBM Corporation a standard 64-character set for the IBM 1401, 1410, 7040, and 7044 Data Processing Systems". Not for the 709, 705-7080 series, or any other equipment].
77. ISO/TC97/SC2(Sec.-12)30, "ISO draft proposal, 6 and 7 bit character codes for information processing interchange", 1963 Jan.
78. ECMA-1, "ECMA standard for a 6 bit input/output character code", 1963 Mar 29.
79. ANS X3.4-1963, "American standard code for information interchange", 1963 Jun 17.
[was X3.2/13, 1962 Sep 10].
80. ISO/TC97/SC2(Sec.-22)75, "Second ISO draft proposal, 6 and 7 bit character codes for information processing interchange", 1963 Dec.
81. ISO/TC97/SC2(Sec.-32)125, "Third ISO draft proposal, 6 and 7 bit character codes for information processing interchange", 1964 May.
82. 7-bit code, CCITT version, 1964 Oct 09.
83. X3.2.4/64, "Draft revised ASCII", 1964 Oct 20.
84. X3.2/172, "Draft proposed revised ASCII", 1964 Dec 03.
85. M-64-900A, "Mil Std Code", 1964 Dec 01.
86. X3.2/206, "Proposed revised ASCII", 1965 Jan 21.
87. ISO/TC97/SC2(Sec.-37)130, "Fourth ISO draft proposal, 6 and 7 bit character codes for information processing interchange", 1965 Mar.
88. ECMA-6, "ECMA standard for a 7 bit input/output character code", 1965 Apr.
[1st edition, superseding drafts TC1/67/5 and TC1/66/69]
89. X3.2/384, "Draft proposed revised ASCII", 1966 May 20.
90. ISO Draft Recommendation 1052, "6 and 7 bit character codes for information processing interchange", 1966 Jun.
91. DIN 66 003, "Informationsverarbeitung, 7-bit-code", 1966 May.
92. X3.2/411, "Draft proposed revised ASCII", 1966 Jul 15.
93. ECMA-6, "ECMA standard for a 7 bit input/output character code", 1967 Jun (2nd edition).
94. ECMA-14, "ECMA standard for rules for the definition of 4 bit sets derived from the ECMA 7 bit coded character set for information interchange", 1967 Nov.
95. ISO Recommendation 646, "6 and 7 bit coded character sets for information processing interchange", 1967 Dec (IDR 1052).
96. USSR State Standard, GOST 13052-67, "Computers and data transmission equipment, alpha-numeric codes", approved 1967 Jul 10. [mandatory, by law].
97. X3.4-1968, "USA standard code for information interchange". [supersedes X3.4-1967 (equivalent to FIPS 1, signed by President Lyndon Johnson on 1968 May 11) by adding NL. Approved 1968 Oct 10].
98. ECMA-17, "ECMA standard for graphic representation of control characters of the ECMA 7 bit coded character set for information interchange", 1968 Nov.
99. ISO Recommendation 963, "Guide for the definition of 4-bit character sets derived from ISO 7-bit coded character set for information processing", 1969 Feb (IDR 1322).

100. ECMA-19, "ECMA standard for the coding of character sets for MICR and OCR", 1969 Jun.
101. "Proposed revision of ECMA-6", (3rd edition, 1969, TC1/69/13).
102. ECMA-23, "ECMA standard for keyboards generating the code combinations of the characters of the ECMA 7 bit coded character set", 1969 Jun.
103. Japanese Industrial Standard Code for Information Interchange, "JIS-CII", C6220-1969.
104. ECMA-6, "ECMA standard for a 7 bit input/output character code", 1970 Jul (3rd edition).
105. IBM Corporate Systems Standard, "Extended BCD interchange code, Latin alphabets", CSS 3-3220-002, 1970 Nov, 40 pp.
106. ECMA-35, "ECMA standard for extension of the 7-bit coded character set", 1971 Dec.
107. ISO Draft Recommendation 1672, "Hardware representation of ALGOL basic symbols in the ISO 6 and 7 bit coded character sets".
108. ISO Draft Recommendation 2022, "Code extension procedures for ISO 7 bit code".
109. ISO Draft Recommendation 2033, "Coding of character sets for MICR and OCR".
110. ISO Draft Recommendation 2047, "Graphical representation for the control characters of the ISO 7-bit character set".
111. ISO Draft International Standard, DIS 2375, "Data processing - Procedures for the registration of escape sequences". (1970 Sep).

Magnetic Tape (Articles and Papers)

112. R.R.Bender, D.J.Galage, "Packing mode control", IBM Tech. Disclosure Bull. 4, 1961 Aug, 61-63.
[packed numerics of an 8-bit code on magnetic tape].
113. -, "Proposed American standard - recorded magnetic tape for information interchange", *Commun. ACM* 7, No. 5, 284-286 (1964 May).
[X3.2/73, 64 Jan 31]
CR 06471
114. R.W.Bemer, "Information processing - a bit expensive?", Forum Editorial, *Datamation* 12, No. 6, 140 (1966 Jun).
115. -, "Proposed USA standard - recorded magnetic tape for information interchange (200 cpi, nrzi)", *Commun. ACM* 10, No. 11, 730-737 (1967 Nov).
CR 13711
116. -, "Proposed USA standard - magnetic tape labels for information interchange", *Commun. ACM* 10, No. 11, 737-743 (1967 Nov).
CR 13712

Magnetic Tape (Drafts and Standards)

117. ANS X3.22-1967, "Recorded magnetic tape for information interchange (800 cpi, NRZI)", (equivalent to FIPS 3).
118. ECMA-12, "ECMA standard for data interchange on 9 track magnetic tape at 31.5 bit per mm (800 bpi)", 1967 Nov (2nd edition 1970 Jun).
119. ISO Recommendation 961, "Implementation of the 6 and 7 bit coded character sets on 7 track 12.7 mm (1/2 in.) magnetic tape", 1969 Feb (DR 1320).
120. ISO Recommendation 962, "Implementation of the 7 bit coded character set on 9 track 12.7 mm (1/2 in.) magnetic tape", 1969 Feb (DR 1321).
121. ECMA-5, "Data interchange on 7-track magnetic tape", 1970 Jun (3rd edition).
122. ISO Recommendation 1861, "7-track 3 rpm (200 rpi) magnetic tape for information interchange", 1971 May (DR 1861).
123. ISO Recommendation 1862, "9-track 8 rpm (200 rpi) magnetic tape for information interchange", 1971 May (DR 1872).
124. ECMA-34, "ECMA standard for data interchange on 3,81 mm magnetic tape cassette (32 b/mm, phase encoded)", 1971 Sep.
125. ECMA-36, "ECMA standard for data interchange on 9-track phase encoded magnetic tape at 63 b/mm (1600 bpi)", 1971 Dec.

Punch Cards (Articles and Papers)

126. R.W.Bemer, "A proposal for a generalized card code for 256 characters", *Commun. ACM* 2, No. 9, 19-23 (1959 Sep).
CR 00025
127. British Standards Institution, "Alphanumeric punching codes for data processing cards", Standard BS3174:1959.
128. R.W.Bemer, W.W.Buchholz, "An extended character set standard", IBM Tech. Publ. TR 00.721, 1960 Jun (revision of TR 00.18000.705, 1960 Jan).
129. H.J.Smith, Jr., F.A.Williams, "Survey of punched card codes", *Commun. ACM* 3, 638, 642 (1960 Dec).
130. D.M.Grant, "A new trinary code compared with existing codes for punched cards", *J. Chem. Doc.* 1, No. 1, 68-69 (1961 Jan).
CR 02713 (null)
131. H.McG.Ross, "Further survey of punched card codes", *Commun. ACM* 4, No. 4, 182-183 (1961 Apr).
132. -, "ECMA standard - representation of the standard ECMA 7-bit code in punched cards", *Commun. ACM* 8, No. 7, 441-442 (1965 Jul).
[so-called Decimal ASCII]
CR 09937
133. E.H.Clamons, "The proposed new punched card code standard", *Control Engg.* 14, No. 2 (1967 Feb).
134. -, "Proposed USA standard - Hollerith punched card code", *Commun. ACM* 11, No. 4, 275-281 (1968 Apr).
135. -, "Correspondences of 8-bit and Hollerith codes for computer environments - a USASI tutorial", *Commun. ACM* 11, No. 11, 783-789 (1968 Nov).
CR 16119

Punch Cards (Draft and Standards)

136. British Standards Institution, "Draft British standard for codes for punched cards", CZ(MEE)6770, 1959 May.
137. EIA RS-292, "Punched Card Standard".
138. ANS X3.26-1969, "Hollerith punched card code" (same as FIPS 14).
[128 characters only]
139. ECMA-20, "ECMA standard for the implementation of the ECMA 7 bit coded character set on punched cards", 1969 Jun.
140. ECMA-25, "ECMA standard for representation of 8 bit combinations in 12 row punched cards", 1970 Aug.
[superseded draft TC1/69/78, 1969 Dec]
141. ANS X3.26-1970, "Hollerith punched card code".
[256 characters - revision of X3.26-1969]
142. ISO Recommendation 1679, "Representation of ISO 7 bit coded character set on 12 row punched cards", 1970 Jul (DR 1679).
143. ISO Recommendation 2021, "Representation of ISO 8 bit coded character set on 12 row punched cards", 1971 May (DR 2021).

Perforated Tape (Articles and Papers)

144. -, "Proposed American standard - perforated tape code for information interchange", *Commun. ACM* 7, No. 6, 336-338 (1964 Jun).
[X3.2/16, 63 Dec 19]
CR 06517

Perforated Tape (Drafts and Standards)

145. Electronic Industries Assoc., "Basic character set code, tentative standards proposal", TR24.4 Doc. 7233 (1960 May).
146. British Standards Institution, AA(DPE)3543, "Draft British standard for punched tape coding, part 1, 7 track code", (1960 Sep), (eventually BS3480:1962).
147. ANS X3.6-1965, "Perforated tape code for information interchange", (approved 1965 Jul 09, equivalent to FIPS 2).
148. ISO Recommendation 1113, "Representation of 6 and 7 bit coded character sets in punched tape", (1969 Nov), (DR 1418).
149. ECMA-10, "ECMA standard for data interchange on punched tape", 1970 Jul (2nd edition).

Communications (Articles and Papers)

150. -, "Proposed American standard - bit sequencing of the American standard code for information interchange (ASCII) in serial-by-bit data transmission", *Commun. ACM* 7, No. 6, 333-336 (1964 Jun). [X3.3/16]
CR 06516
[expository remarks say that conclusion for HOF is based upon economical design of equipment; contains complete summary of input documents].
151. F.C.White, "Bit sequence stirs data/comm feud", *Air Transport World*, 1964 Jul, 26-28.
152. R.W.Bemer, J.Booth, "Comments on bit-sequencing of the ASCII in serial-by-bit data transmission", *Commun. ACM* 7, No. 8, 483-485 (1964 Aug).
CR 06519
["offers powerful arguments in favor of 'low-order' bit sequencing, which is contrary to the proposed American standard"].
153. -, "Transparent-mode control procedures for data communication, using the American standard code for information interchange - a tutorial", *Commun. ACM* 8, No. 4, 203-206 (1965 Apr).
CR 07783
154. E.H.Clamons, "Low-order or high-order, ?tsrif tib hcihw", *Control Engineering*, 1965 Aug, 102-103
155. -, "Proposed American standard, character structure and character parity sense for serial-by-bit data communication in the ASCII", *Commun. ACM* 8, No. 9, 553-556 (1965 Sep). [X3.3/23].
156. -, "Proposed American standard, character structure and character parity sense for parallel-by-bit data communication in the ASCII", *Commun. ACM* 9, No. 9, 695-697 (1966 Sep). [X3.3/35].
157. -, "Data communication control procedures for the USA standard code for information interchange", *Commun. ACM* 12, No. 3, 166-178 (1969 Mar).
[same as ISO/TC97/SC6/278].
CR 16858
158. W.L.Harper, "The remote world of digital switching", *Datamation* 17, No. 6, 22-31 (1971 Mar 15).
159. S.R.Rosenblum, "Progress in control procedure standardization", *Proc. 2nd Symp. on Problems in the Organization of Data Communication Systems*, 153-159 (1971).
CR 22425

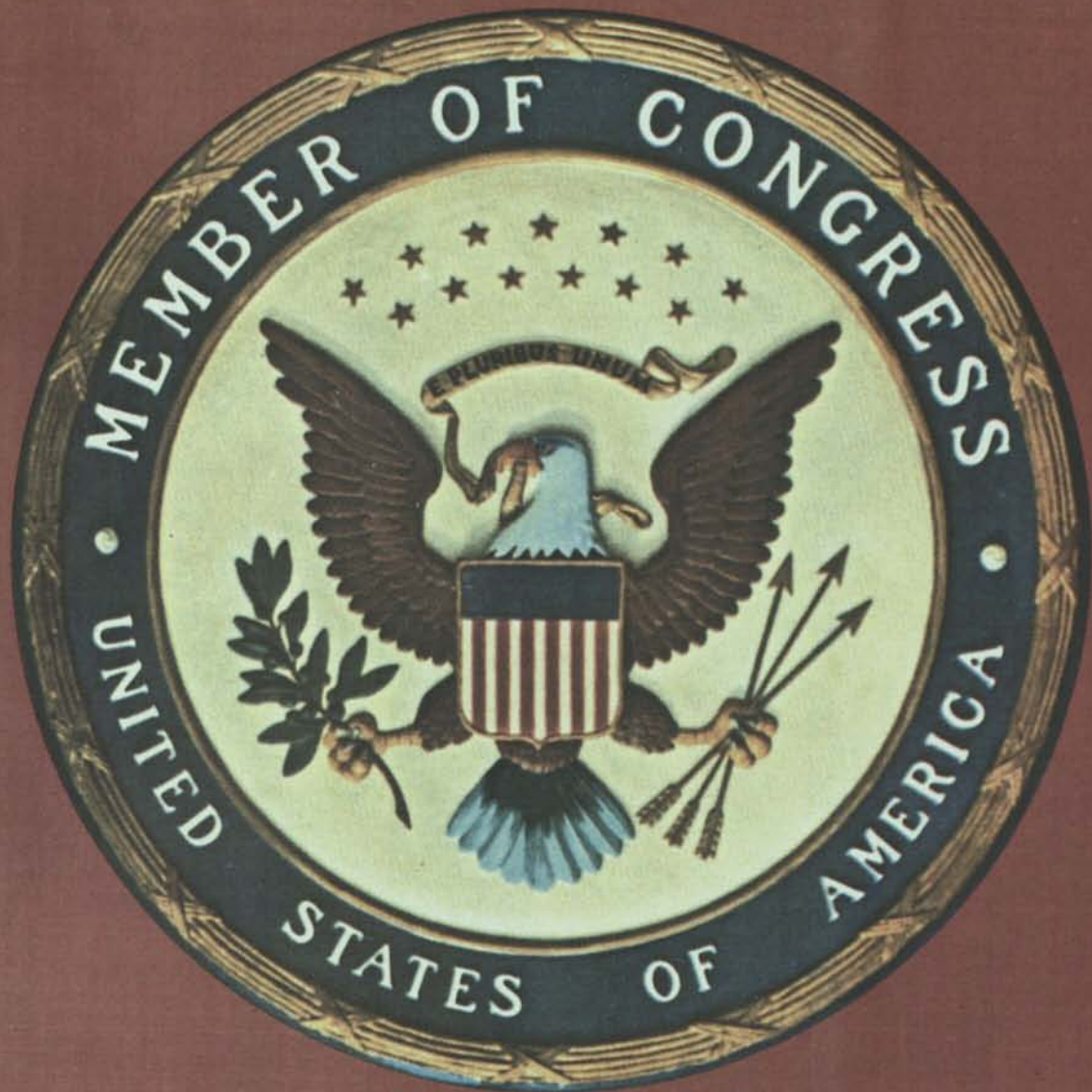
Communications (Drafts and Standards)

160. X3.3/7, "Working paper, character structure for serial-by-bit transmission in the ASCII", 1964 Jul 07.
[first separation of parity sense from bit sequence].
161. ANS X3.15-1966, "Bit sequencing of the ASCII in serial-by-bit data transmission" (equivalent to FIPS 16).
[X3.3/23, 65 Jun 09, approved 66 Aug 19].
162. ANS X3.16-1966, "Character structure and character parity sense for serial-by-bit data communication in the ASCII" (equivalent to FIPS 17).
[X3.3/..., approved 66 Aug 19. ODD for synchronous, EVEN for asynchronous].
163. ECMA-16, "ECMA standard for basic mode control procedures for data communications systems using the ECMA 7 bit code", 1968 May.
164. ANS X3.25-1968, "Character structure and character parity sense for parallel-by-bit data communication in the ASCII" (equivalent to FIPS 18).
[X3.3/35, 66 Feb 09, approved 68 Oct 24].
[equivalent to ISO/TC97/SC6(USA-15)124].
165. ECMA-24, "ECMA standard for code independent information transfers", 1969 Dec.
166. ECMA-26, "ECMA standard for recovery procedures", 1971 Apr.
167. ECMA-27, "ECMA standard for abort and interrupt procedures", 1971 Apr.
168. ECMA-28, "ECMA standard for multiple station selection procedures", 1971 Apr.
169. ECMA-29, "ECMA standard for conversational information transfer", 1971 Sep.
170. ANS X3.28-1971, "Procedures for the use of the communication control characters of the ASCII in specified data communication links".
171. ISO Draft International Standard, DIS 2111, "Code independent transmission procedures".

the **honeywell** computer journal

Vol. 6 | No. 4

1972



STANDARD JOURNAL ISSUE CODE: HNCJA3 6(4) 259-306 (1972)

COMPUTERS IN THE CONGRESS

A VIEW OF THE HISTORY OF THE ISO CHARACTER CODE

EMULATION: A SURVEY

3-char control (7-bit code)
 alternate 2-char mnemonic
 ANSI mnemonic (8-bit code)
 ECMA mnemonic (if different)

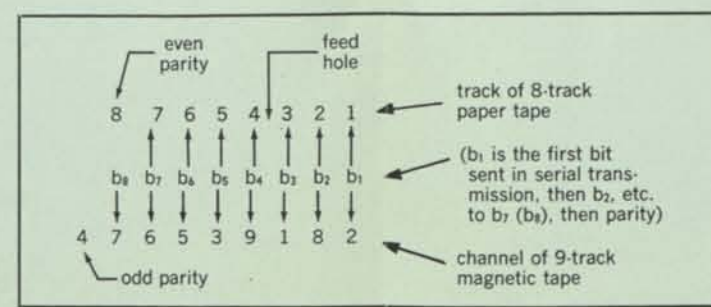
ACK (AK)	Acknowledge
BEL (BL)	Bell
BS	Backspace
CAN (CN)	Cancel
CD	Character Delete
CI	Character Insert
CIF	Character Insert Off
CIN	Character Insert On
CLC	Clear Line from Cursor
CR	Carriage Return
CSC	Clear Screen from Cursor
DC1 (D1)	Device Control 1
DC2 (D2)	Device Control 2
DC3 (D3)	Device Control 3
DC4 (D4)	Device Control 4
DEL (DT)	Delete
DLE (DL)	Data Link Escape
EM	End of Medium
ENQ (EQ)	Enquiry
EO	Eight Ones
EOT (ET)	End of Transmission
ESC (EC)	Escape
ESI	Extended Shift In
ESO	Extended Shift Out
ETB (EB)	End of Transmission Block
ETX (EX)	End of Text
FF	Form Feed
FS	File Separator
GS	Group Separator
HF	Highlight Off
HLF	Half Line Feed
HLR	Half Line Reverse Feed
HN	Highlight On
HT	Horizontal Tab
HTC	Horizontal Tab Clear
HTS	Horizontal Tab Set
LCF	Local Copy Off (full duplex)
LCN	Local Copy On (half duplex)
LD	Line Delete
LI	Line Insert
IL	Line Insert
NAK (NK)	Negative Acknowledge
NP	Next Page
NUL (NU)	Null
PD	Cursor (Pointer) Down
PFF	Protect Format Off
PFN	Protect Format On
PH	Cursor (Pointer) Home
PL	Cursor (Pointer) Left
PM	Cursor (Pointer) Return
PP	Previous Page
PR	Cursor (Pointer) Right
PT	Cursor (Pointer) Tab
PU	Cursor (Pointer) Up
RLF	Reverse Line Feed
RS	Record Separator
SD	Scroll Down
SI	Shift In
SO	Shift Out
SOH (SH)	Start of Heading
SP	Space (a blank)
STX (SX)	Start of Text
SU	Scroll Up
SUB (SB)	Substitute
SYN (SY)	Synchronous Idle
US	Unit Separator
VT	Vertical Tab
VTC	Vertical Tab Clear
VTS	Vertical Tab Set

Note 1
 These 12 positions are variable — 2 for currency, 7 primary national usage, and 3 secondary usage which are diacritical marks used for alphabetical extension when preceded by BS. Positions 2/7 and 2/12 are invariant but also serve as diacritical marks. The presently-known assignments are given in the table below.

JISCI (Japanese Industrial Standard Code for Information Interchange) is an 8-bit code consisting of the ISO characters plus the Kata Kana characters shown in the upper row positions of columns 10-13 (columns 8 and 9 are reserved for additional controls, 14 and 15 for additional graphics).
 GOST 13052-67 defines the USSR set, shown in the lower row entry positions of columns 12-15. Actually, the standard defines these characters for columns 4-7 of a 7-bit set (SO=Russian register, SI=Latin register). Columns 8-11 are identical to 0-3.

COL	b ₈ b ₇ b ₆ b ₅		b ₄ b ₃ b ₂ b ₁		(ESC) (CHAR.)															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
0000	0	NUL	DLE	SP	0	NOTE 1	P	NOTE 1	p	CI	CD									
0001	1	SOH	DC1	!	1	A	Q	a	q	PU	CIN	o	р	ф	я	А	Я			
0010	2	STX	DC2	"	2	B	R	b	r	PD	CIF	г	й	ц	ч	Б	Р			
0011	3	ETX	DC3	#	3	C	S	c	s	PR	SU	л	у	ш	щ	С	Ш			
0100	4	EOT	DC4	\$	4	D	T	d	t	PL	SD	,	и	к	э	Д	Т			
0101	5	ENQ	NAK	%	5	E	U	e	u		NP	.	о	е	у	Е	У			
0110	6	ACK	SYN	&	6	F	V	f	v		PP	э	ка	и	э	Ф	Ж			
0111	7	BEL	ETB	'	7	G	W	g	w	PM	PFN	р	к	х	у	Г	В			
1000	8	BS	CAN	(8	H	X	h	x	PH	PF	й	к	х	ъ	Х	Ъ			
1001	9	HT	EM)	9	I	Y	i	y	PT		у	к	и	ы	И	Ы			
1010	10	LF	SUB	*	:	J	Z	j	z	CSC		и	к	л	л					
1011	11	VT	ESC	+	;	K	NOTE 1	k	NOTE 1	CLC		*	к	т	о					
1100	12	FF	FS	,	<	L	NOTE 1	l	NOTE 1	LI		р	э	л	э					
1101	13	CR	GS	-	=	M	NOTE 1	m	NOTE 1	LD		у	л	м	щ					
1110	14	SO	RS	.	>	N	NOTE 1	n	NOTE 1			э	т	н	ч					
1111	15	SI	US	/	?	O		o		DEL		щ	у	р	о		EO			

	currency		1st 7 national				dia	dia	1st 7 national				dia
	2/3	2/4	4/0	5/11	5/12	5/13	5/14	6/0	7/11	7/12	7/13	7/14	
Netherlands—A													
Australia	#												
Belgium—A													
W. Germany—A													
US													
Japan													
UK													
Italy—A													
Switzerland—A													
France—A													
USSR													
Netherlands—B													
Belgium—B													
France—B													
Switzerland—B													
Italy—B													
Switzerland—C													
Hungary													
W. Germany—B													
Switzerland—D													
Sweden													
Finland													
Denmark													
Norway													
Spain													



Alternate controls in these 5 columns are achieved by preceding the regular character with an ESCape.

The Hollerith card code for 256 characters is constructed from 1 or 2 or 3... or 7 or blank (no punch) and any combination of 12, 11, 0, 8, and 9 (from none to all) $\times 32 (2^5) = 256$

For historical reasons, the assignments present little in the way of a regular pattern, but they are the key to translate to and from IBM EBCDIC.

REFERENCE CHART ISO CODE AND ASSOCIATED RELATIONSHIPS
 Note — this is not a standard in itself. Refer to the appropriate documents (see reverse side). Screened characters in columns 3, 4, 5, 8 and 9 are under consideration.

Reprints of this chart are available from the Honeywell Computer Journal (P.O. Box 6000, Phoenix, AZ 85005) at \$1 each postpaid.

	ISO	ECMA	ANSI	US FIPS PUB	JIS	USSR	UK
Binary code for characters (plus control meanings)	R646 (67 Dec)	ECMA-6 3rd Ed. (70 Jul)	X3.4-1968	1	C6220-1969	GOST 13052-67	
Hollerith card code	R2021 (71 May)	ECMA-20 (69 Jun)	X3.26-1969	14			BS4636/3 1971
Graphics for the controls	DR2047	ECMA-17 (68 Nov)	X3L2/987 (70 Jul 29)				
National usage	97/2/525		X3L2/1238A				
Additional controls			X3L2/1198				
Serial transmission	R1177 (70 Jan)		X3.15-1966				
Track assignment on 25.40 mm (1") punched tape	R1113 (69 Sep)	ECMA-10 2nd Ed. (70 Jul)	X3.6-1965	2	C6221-1969		BS3880/3 1971
Track assignment on 12.70 mm (0.5") magnetic tape	R962 (69 Feb) R1863 (71 May)	ECMA-12 2nd Ed. (70 Jun)	X3.22-1967	3	C6222-1969		

SOURCE DOCUMENTS - STANDARDS AND DRAFT STANDARDS

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	
0	0	0	0	0	0	0	NUL TC _(OLE)
0	0	0	0	1	0	1	TC _(SOH)
0	0	0	1	0	1	0	TC _(STX)
0	1	2	3	4	5	6	TC _(ETX)
0	1	2	3	4	5	6	TC _(EOT)
0	1	2	3	4	5	6	TC _(ENQ)
0	1	2	3	4	5	6	TC _(ACK)
0	1	2	3	4	5	6	TC _(ETB)
1	0	0	0	0	0	0	FE _(BS)
1	0	0	1	0	0	0	FE _(HT)
1	0	1	0	0	0	0	FE _(LF)
1	0	1	1	0	0	0	FE _(VT)
1	1	0	0	0	0	0	FE _(FF)
1	1	0	1	0	0	0	FE _(CR)
1	1	1	0	0	0	0	S0
1	1	1	1	0	0	0	SI
1	1	1	1	1	0	0	'
1	1	1	1	1	0	1	`
1	1	1	1	1	0	1	P
1	1	1	1	1	0	1	Q
1	1	1	1	1	0	1	R
1	1	1	1	1	0	1	S
1	1	1	1	1	0	1	T
1	1	1	1	1	0	1	U
1	1	1	1	1	0	1	V
1	1	1	1	1	0	1	W
1	1	1	1	1	0	1	X
1	1	1	1	1	0	1	Y
1	1	1	1	1	0	1	Z
1	1	1	1	1	0	1	[
1	1	1	1	1	0	1	\
1	1	1	1	1	0	1]
1	1	1	1	1	0	1	^
1	1	1	1	1	0	1	_
1	1	1	1	1	0	1	DEL

Official Reference Version — R646 (in OCR-B font)

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	
0	0	0	0	0	0	0	NUL TC _(OLE)
0	0	0	0	1	0	1	TC _(SOH)
0	0	0	1	0	1	0	TC _(STX)
0	1	2	3	4	5	6	TC _(ETX)
0	1	2	3	4	5	6	TC _(EOT)
0	1	2	3	4	5	6	TC _(ENQ)
0	1	2	3	4	5	6	TC _(ACK)
0	1	2	3	4	5	6	TC _(ETB)
1	0	0	0	0	0	0	FE _(BS)
1	0	0	1	0	0	0	FE _(HT)
1	0	1	0	0	0	0	FE _(LF)
1	0	1	1	0	0	0	FE _(VT)
1	1	0	0	0	0	0	FE _(FF)
1	1	0	1	0	0	0	FE _(CR)
1	1	1	0	0	0	0	S0
1	1	1	1	0	0	0	SI
1	1	1	1	1	0	0	'
1	1	1	1	1	0	1	`
1	1	1	1	1	0	1	P
1	1	1	1	1	0	1	Q
1	1	1	1	1	0	1	R
1	1	1	1	1	0	1	S
1	1	1	1	1	0	1	T
1	1	1	1	1	0	1	U
1	1	1	1	1	0	1	V
1	1	1	1	1	0	1	W
1	1	1	1	1	0	1	X
1	1	1	1	1	0	1	Y
1	1	1	1	1	0	1	Z
1	1	1	1	1	0	1	{
1	1	1	1	1	0	1	
1	1	1	1	1	0	1	}
1	1	1	1	1	0	1	-
1	1	1	1	1	0	1	DEL

International Reference Version (in OCR-B font)

THE ROLE OF A COMPUTER IN THE PUBLICATION OF A PRIMARY JOURNAL

by ROBERT W. BEMER

Honeywell Information Systems, Inc.
Phoenix, Arizona, US

INTRODUCTION

The Honeywell Computer Journal has had some acclaim for social responsibility in the computer milieu and for the extensive and pervasive use of a computer in the publishing function. The basic elements of the latter are described here. The Journal is published simultaneously in hardcopy, microfiche, and magnetic tape with embedded text control. Its mixed-media character is accentuated by the fact that not all articles in the microfiche and tape editions appear in the hardcopy edition.

Specifically, the copy that you are now reading has been produced by the identical methods of the Honeywell Computer Journal, as are all of the papers in the Methods and Applications Section of these Proceedings. Thus many of the features can be self-descriptive. The only differences are:

- Video Times Roman font is used here (instead of Optima).
- Column width is 242 points (instead of 228).
- Column height is 57 lines maximum (instead of 60).

To reset this paper for the alternate conditions would cost \$3.50 per page!

MAJOR COMPONENTS OF COMPUTER USE

The computer plays a major role in:

- Subscription fulfillment.
- Entry of text, tables, and figures.
- Production of photocomposed copy, with justification and hyphenation.
- Control of page layout.
- Proofing of copy.
- Control of readability and style.
- Indexing.

All except the first and last functions are covered in this paper. The first is omitted because it is common, and we have made no innovations; the last because we make little use of this admittedly powerful feature for the Journal per se.

Furthermore, we do not use the automatic pagination features that are available to us, because computers can never be more than dull and pedestrian in this role. It may be suitable for a contract specification, or legal documents, but not for a publication that must be artistic, attractive, and readable. Automatic pagination also chews up expensive store and time to keep the total text in core to work with.

SCHEMATIC OF THE SYSTEM

The HIS 6000 system is used for text entry, editing, storage, and running concordances. It is not normally used for the "run-off" function (producing formatted copy on the entry terminal). Even though this feature is available, it is tedious, expensive in line cost, and has little value for final copy.

Formatted copy is produced only by photocomposition. When this is desired, a special postprocessor program converts the text stream and embeds macros for the Page 2 System. This produces a magnetic tape which is (now) transported physically to the facilities of Datagraphics, in Phoenix, and input to a Univac (nee RCA) 2 driving a III Videocomp 830. The resulting copy is laid up in desired page form, and a cycle of editing and further photocomposition begins.

Final copy is waxed on templates in the traditional manner. Special heads are added (in fonts not available to the computer system, and chosen to symbolize article content, where possible), and it's off to the printers.

Basically, we have adjoined two free-standing systems, and in so doing removed from the middle the expensive and non-graphic-quality output of the first, and the somewhat tedious and inflexible input of the second. Jury-rigged as it is, it is nevertheless superior to any method formerly available to us, and points the way to integrated systems for the future. We can live for now with our 2-hour turnaround.

TEXT ENTRY

Text entry is accomplished in the timesharing mode with the standard HIS 6000 Text Editor System,¹ an embedded format system based upon M.I.T. work and similar to the IBM Script. It is *not* a numbered line system like ATS, and eight years of experience has proved *this* wisdom. Searching and alteration are done primarily in the string mode. "Cut and Paste" is limited to operate by number of lines moved, but they are not numbered.

All control actions are signified by embedded "dot" commands. This input convention states that a CR (Carriage Return) character followed by a full stop character (period, dot) signifies a control statement, thus:

```
.begin      .center    .indent n  .subpara n .TAB  
.space (n) .adjust    .indent n  .para      .break
```

These are but a subset of the standard Text Editor, and can be learned by an unskilled person in an hour or so. The editing commands will be explained in the running text of this paper.

CONCLUSION

As the 93rd Congress begins its deliberations, the complex question of limiting Federal expenditures will be a primary subject of concern. It is a fact of life that neither man nor nation can live within available resources without reliable information about needs and expenditures. For this reason it is hoped that among the solutions that are devised will be the granting of top priority to the development of the computer system to support the budget and appropriations cycle. Above all other considerations, this is the most critical need of the Congress. With annual expenditures at the \$250 billion level, even a minor improvement in the budget and appropriation system would save billions.

Computers are the only hope that our Congress has to acquire the basic data needed to control expenditures. Without this data there can be no effective Congress and, ultimately, no democratic system.

REFERENCES

1. "Strengthening Program Planning, Budgeting, and Management in the Federal Government", McKinsey & Co., 1970 Dec.

Selected Readings

American Enterprise Institute for Public Policy Research. "Congress: the first branch of government". AEI, Wash., DC, 1966. 515 pp. (see papers by C.R. Dechert, K.Janda, J.A. Robinson).

Brooks, Jack. "Data-processing techniques to aid Congress". Extension of remarks in the House, Congressional record [Daily edition], 115, 1969 Jan 3, pp. E29-E31.

Califano, J.A., Jr. "Congress has been bypassed in analysis technology", the Washington Post, 1971 Jul 13, p. A-18.

Chartrand, R.L. "Congress, computers, and the cognitive process". In *Planning and politics: uneasy partnership* (T.L. Beve and G.T. Lathrop, eds.), 167-187, the Odyssey Press, New York, 1970.

----, "Redimensioning congressional information support", *Jurimetrics J.*, 11, No. 4, 165-178 (1971 Jun).

Chartrand, R.L., K.Janda, and M.Hugo, eds. "Information support, program budgeting, and the Congress", Spartan Books, New York, 1968. 231 pp. (see papers by Chartrand, Janda, W.T. Knox, R. McClory, F. Schwengel).

Comptroller General of the US. "Budgetary and fiscal information needs of the Congress", US Genl. Acctg. Off., Wash., DC, 1972 Feb 17. 49pp.

"Congress looks to computers for legislative help", *Congressional Quarterly*, 1969 Apr 11, 524-525.

Donham, P., and R.J. Fahey. "Congress needs help". Random House, New York, 1966, 203pp.

"Digitizing Congress", *Electronics*, 1970 Aug 31, pp. 41-42.

Glass, A., CPR report, "Congress moves into computer age but divides on control of new systems", *Natl. J.*, 2, No. 22, 1150-1157 (1970 May 30).

Hopkins, B.R. "Congressional reform: toward a modern Congress", *Notre Dame Lawyer*, 47, No. 3, 442-513 (1972 Feb).

Hunter, K.W. "Toward better information and analysis support services for the Congress", *Proc. Assoc. Comput. Mach.*, 1971, 139-155, ACM, NY.

Information systems: current developments and future expansion. Proceedings of a special seminar held for Congressional Members and staff, 1970 May 20, AFIPS [Amer. Fed. Information Processing Soc.], Montvale, NJ, 88p.

Janda, K. "Information retrieval -- applications to political science", Bobbs-Merrill, New York, 1968, 230pp.

Kravitz, W. "Preparation of committee legislative calendars by computer: description of a working system", Legislative Reference Service, Library of Congress, 1969 Mar 3. 8pp. + appendices.

Little, A.D., Inc. "Management study of the US Congress", commissioned for special NBC news report, "Congress needs help", 1965 Nov 24, 37p.

McClory, R. Congressman McClory suggests computer uses for Congress. Extension of remarks of Honorable Tom Railsback, Congressional record [Daily edition], 114, E275-E277, 1968 Jan 29.

Maiorana, Charles. A computer system to aid in Congressional decision making. The George Washington Univ., Wash., DC, 1966 May 1, 40 pp.

Moorhead, William S. Computers for the Congress: the challenge and the potential. An address before the 15th Annual EDP conference of the American Management Assoc., New York, 1969 Feb. 15pp.

Schloss, L. "Congress needs computers -- but they're many years away", *Government Executive* 2, No. 6, 42-43 (1970 Jun).

Schneier, E. "The intelligence: information and public policy patterns", *Annals of political and social science* 388, 14-24 (1970 Mar).

Semling, H.V., Jr. "Congress and the computer", *Modern Data*, 1969 Sep, 68-71.

Sisk, B.F. "Toward a more effective Congress", *Los Angeles Times*, 1969 Oct 22, part 2, p. 1.

Truman, D.B., ed. "The Congress and America's future", Prentice-Hall, Englewood Cliffs, NJ, 1965, 185pp.

US Congress. House Committee on Government Operations. "Use of computers in the legislative process", Report together with supplemental views of the committee, 91st Congress, 1st session. Report No. 91-258, US Govt. Print. Off., Wash., DC, 1969. 39p.

----, House Subcommittee on Government Activities. "Effective and efficient use of computers in Congress", Hearing before the subcommittee, 91st Congress, 1st session, 1969 Apr 23, US Govt. Print. Off., Wash., DC, 1969, 53pp.

US Congress. House Committee on Administration. First progress report of the special Subcommittee on Electrical and Mechanical Office Equipment. 91st Congress, 1st session, October 1969, 64 pp. (see also the second progress report, 20 pp., and the special report of the same committee, 17 pp., 91st congress, 2nd session), US Govt. Print. Off., Wash., DC.

----, House Committee on Rules. "Legislative reorganization act of 1970", Report of the Committee on Rules on H. R. 17654. 91st Congress, 2nd session, House Report No. 91-1215. US Govt. Print. Off., Wash., DC, 1970, 182pp.

----, Public Law 91-510 [84 STAT. 1140]. An act to improve the operation of the legislative branch of the Federal Government, and for other purposes. 91st Congress. H. R. 17654, 1970 Oct 26. 65pp.

----, Senate Committee on Rules and Administration. "Initial report of the Subcommittee on Computer Services to the Committee on Rules and Administration", 92nd Congress, 1st session, 1971 Jul 21, US Govt. Print. Off., Wash. DC, 11pp.

----, Secretary of the Senate. Administrative reorganization of the offices of the Secretary of the Senate. Based on administrative survey by O.B. Potter. 92nd Congress, 2nd session, US Govt. Print. Off., Wash., DC, 1971. 11pp.

ENTRY FOR UNSKILLED PERSONNEL

The postprocessor program that converts for the Page 2 System is vital for simple text entry. The standard entry methods for the Page 2 System are certainly *not* simple, and require some training and a crib sheet constantly on display to the enterer. Remember that graphic quality output requires a separate font generation for each unique character. It is not sufficient to overprint an umlaut (in its fixed position) for both the upper and lower case "u", for example. The postprocessor does extensive string analysis, much of it based upon backspace and overstrike for entry, which makes it simple for personnel. Examples:

- Characters with diacritical marks - accent acute, accent grave, tilde, umlaut, etc. - are produced by backspace on the terminal and overstrike with the proper character (double quote is used for umlaut).
- Double and single quotes are used as they are for entry. The postprocessor determines whether they are opening or closing quotes. A double quote is two single quotes in photocomposition, and this is called automatically.
- For minor occurrences in text, boldface may be indicated by overstriking single characters three times. This is visible on the terminal when the line is verified. For a longer string of bold characters, the font is altered by a `.bold` command, and turned off by a `.bold end` command. These commands do not force a new line.
- For minor occurrences in text, italics may be indicated by backspacing the length of the word and underlining. This is visible on the terminal when the line is verified. For a longer string of italics, the font is altered by a `.ital` command, and turned off by a `.ital end` command. These commands do not force a new line.
- The bulleting seen here is accomplished by a `.indent 3` followed by a `.undent 3` (which is operative only for the next line), a lower case "oh", 2 blanks, and then the text. The uniqueness of this string permits the convention.
- To the regular Text Editor convention of using the "at" symbol to delete the previous character (guess why our articles never contain this character!), and CAN to delete the entire line of entry, we have added the caret to indicate the "en" space, which is incompressible to the justification process. Thus a new paragraph is caused by a `.break` and an initial line with two carets for indention.
- Normal font sizes for the Journal are:

- 9 point - text
- 8 point - references, some displays as necessary
- 7 point - sub- and superscripts, figure captions

Point size may be changed at any point in the text by inserting the ESCape sequence:

ESC g (7-pt), ESC h (8-pt), ESC i (9-pt)

These override the original settings, and are used for formulas, etc.

CONTROL OF PAGE LAYOUT

It has been a remarkable discovery to us that reader attraction and satisfaction is increased significantly by tight control of page layout. Only in the most exceptional cases will a column start in the middle of a sentence, and then only on the second column of the same page. Usually a column will start with at least a paragraph (not just an arbitrary paragraph, but one that makes sense), and very often with a heading. The appearance of a figure or table will never precede its first mention in text, nor will it often be on a page that is not visible when that mention is made. "Widows" never occur.

Under traditional methods, the editor loses control of page layout after the galley stage; all of the niceties must be left to a composer who has little understanding of the subject matter, and is often less interested in reader satisfaction. With the low cost of text processing taken in conjunction with photocomposition, we do not mind expending many runs to get just what we want.

A quick reading of the first galley copy gives an estimate of the author's redundancy or flowery speech factor, and other ways that compression can be achieved if necessary. Accordingly, the actual film is cut to lay out an approximation of the article. As the last page is always full, we work backward. Whatever is left for the first page we leave for artistic treatment and the "From the Editor" commentary. Great attention is paid to aspects of future readability, left or right page assignment, pleasing placement of tables, figures and photos. Virtually no attention is paid to typos and other mistakes that exist in the copy. Accordingly, the single columns are taped on with more lines than our standard, trusting to editing to cut back to the right number (60).

The beauty of this system is that many things can be changed simultaneously to create correctness, harmony, and interest - point size for certain paragraphs or tables, tab settings, subparagraphing, font style, and text changes and corrections. Imagine a situation where the column copy has to be reduced by two lines, and yet previous editing has taken advantage of all short lines at the end of paragraphs, filler words have been removed, and big words replaced by commoner smaller words with equivalent or clearer meaning. Now you have to get into the guts of the author's meaning and say it shorter and clearer, without altering the flavor or meaning in any way! Being forced to do this by our aesthetic standards for page layout yields a big dividend in increased readability.

Depending upon the content, we may photocompose the text from 2 to 5 times. Do the authors complain about the alterations? Never, in our experience. When it reads well, they just assume that they wrote it that way, never checking their original copy. We have also experimented in putting the author's work in to typeset even when it is only rough draft; results seem to indicate that the visualization of final copy permits him to improve it more than he could by editing from a typed draft.

Obviously, taking this much work for readability means high acceptance standards, and we insist that this is a good thing. Dung coated with 53 layers of Chinese lacquer is still dung, and we do not intend contributing to information pollution.

Hopefully, it is now clear why we do not use the computer for automatic pagination.

PROOFING OF COPY

An optional feature, or byproduct, is the concordance run, usually exercised on what is expected to be the next-to-last photocomposition run. This produces two listings on the high-speed (upper case only) printer. The first listing is a Key Word Out of Context (KWOC) listing; each numeral and word (except for the very small common ones) is listed on the left in collating sequence order, with its entire entry line on the right. The lines are numbered here, for cross-reference to the second listing, which is the consecutive text.

The concordance is now scanned visually, primarily to detect input errors ("typos"). See Figure 1 for some examples. It is our experience that these fairly jump out at one in scanning a concordance, whereas they remain stubbornly glossed over by the eye and mind in traditional proofreading. However, we do read the text - for style and making sense, not for typos. In fact, knowing that you are freed from the typo-hunting task creates a different frame of mind for doing *real* editorial work.

AUTOMHILF	DISPOSITION	MASSACHUSETTS
AUTOMHILE	DISRRUPTION	MASSACHUSETTS
AUTOMHILLES	DISRUPT	MASSACHUSETTS
AVAILABILITY	DISRUPTIONS	MASSES
CERTAINLY	INSTEAD	SOLVED
CERTAINTY	INSTEAD	SOLVED
CERTIFICATION	INSTITUTE	SOLVED
CERTIFICATE	INSTITUTE	SOLVED,
CERTIFICATION	INSTITUTE	SOLVE,
CHARGES	KURT	STEREOSCOPIC
CHARGES	KY,	STEREOTYPED
CHARIMAN	LB	STEREOTYPED
CHARITY	LA	STEREODRAPHIC
CHARLATANS	LAR	STEREOTYPE
		STEWARDS
COLUMBIA	LOCATIONS	SUCCESS
COLUMBIA	LOCATION,	SUCCESS
COMPANY	LOCATION,	SUCCESSING
COMBINATION	LOCATIONS	SUCCESS
COMBINATION	LOGGED	SUCH

Figure 1. Typos Exposed by Concordance

CONTROL OF READABILITY AND STYLE

The concordance produces a histogram of word size distribution as a byproduct, and the average word length may be calculated. We target 5.0 characters per word, and are very suspicious of readability when the author gets above 5.5.

One aspect of style, or rather one of our rules, is that an acronym shall always be given the spelled-out version in parentheses the *first* time it is encountered in text. One has only to spot the first occurrence in the concordance, and look to the corresponding line on the right to see if this has been done. If not, edit.

The Journal has other style rules. Most important is adherence to ISO Standard 1000, or the International System of Units (SI). Check the concordance for inches, feet, yards, miles, pounds, etc. if they occur, and are for measurement, they had better be in parentheses following a metric value. Other examples: \$2 million - not 2 million dollars; 0.5 s - not .5 sec; focused - not focussed.

ECONOMIC CONSIDERATIONS
IN WORKING METHODS

As there is no way to predict the pagination of printed copy when entering text, one could enter it all under a single file name. However, the 6000 Text Editor keeps the entire file in the main store for faster processing (and it is *really* fast), and these facilities must be paid for. Thus original input is made in judiciously separated and named files, breaking at headed sections, for example. These are then adjoined for the photocomposition run.

After page layout is determined, they are adjoined again and resplit by page into files with new names, and the old ones purged. This permits single columns to be reworked into final form. The present rate is \$1.75 per column. Thus a page costs from \$6 to \$10 to compose, comparing rather favorably with the \$70 per page we were paying for linotype setting to our standards before our system was operable. The 6000 cost is not included, as we have been unable to get real figures because we work on an inhouse "exposure" system used for checking out new software releases. We do, however, feel that this cost is compensated by the system doing automatically what we would have to do ourselves otherwise (like proofreading), and the added quality. We do need to modify to set double column on the last run.

Economy dictates that we should process as much text as possible on each photocomposition run. This means linking several files and saving them as a single file. But this increases the risk that something going wrong early will spoil the balance. Care must be taken to separate and insulate each file from any other. Convention starts each file with `.begin` (for a new galley), `.indent 0` (in case the file ahead of it lacked a command to restore indentation to 0), and `.adju` (in case the preceding file had been using tabulation and was not restored to the justification mode).

The power of the Text Editor is of great assistance in checking for correctness of the adjoined file, particularly for closure. Type:

```
fs:/bold/* (meaning "find all occurrences of that string")
```

and you will almost instantly get a message like:

```
end of file - request executed 122 times
```

Hit "b" and CR (for backup to the file beginning, and type:

```
fs:/bold end/*
```

If the message doesn't say 61 times - trouble! A 60 would mean that bold did not get turned off somewhere, and the copy following will be in useless boldface. Do the same for italics, subparagraphs, point size changes, etc.

The files must always be correct for the magnetic tape edition, and identical to the printed copy. Yet it is often wasteful to rerun the entire file for simple patches. A copy is made, and the correct parts wiped out by string replacement, leaving only the changed copy to be reset as a patch (with due consideration to leaving enough text so that paragraphing, etc., is unchanged). These patches are saved under a different name; a number of them are adjoined and run at one time.

INCIDENTAL ADVANTAGES

A number of dividends have shown up that we amateurs did not really foresee:

- Doing our own typesetting permits laying up mechanicals for articles as soon as they are ready, without waiting to group an entire issue for the typesetter to schedule in some time slot. Exclusive of conditions of extreme timeliness, this permits better selection for issue makeup and content.
- Having the feel of the final product, by mockup during the editing and changing stages, affects everyone - author, editor, and reviewer. For the latter, particularly, it gives psychological impetus to hurry up - lest what he dislikes might be in the finished product. All can work simultaneously to correct and improve the copy and make it more readable.
- The Page 2 System hyphenates to English rules and/or custom. Normally we run our French, German, Italian, and Spanish sections in "fill mode" (stretching the spacing between words to fill the line without hyphenation). But if glaring gaps exist we remove them easily by doing a dummy hyphenation, splitting the first word of the next line into two components:

```
rs:/whippersnapper/
ENTER
*whipper- snapper
.
READY
```

This technique can also be used in our English text when Page 2 fails to hyphenate opportunely or (rarely) incorrectly.

On one occasion the entire article was side-by-side in both German and English. Here we could proceed more elaborately, removing Page 2 hyphenation that was incorrect for German, forcing correct hyphenation paragraph by paragraph.

- Page 2 also has the flaw of assuming that a change in font style permits a break for a new line just as hyphenation or a space does:

```
..... Protection A
      gency ...
```

Text Editor can force a correction by replacing sufficient spaces between words by incompressible en spaces.

- We don't have to worry about losing corrected galley in the mails, as the Journal of the ACM did in 1971 October. We also know that the corrections have actually been made in the printer's copy, without waiting for a blue to be returned and show that they were *not* made. This often shortens the production cycle, and certainly cuts costs.
- Secretaries can make very creditable copy inhouse by cutting and pasting galley segments with Scotch Tape, and then using a reproduction method such as Multilith. Interoffice memos are becoming artistic, easier and pleasanter to read, and certainly use less paper.

OUR WISH LIST

A major purpose of the First National Computer Conference and Exposition was to have the end users tell the suppliers the nature of their applications and what they would like to accomplish those applications better, cheaper, and faster. I must follow my own principles. We would like:

- A larger portion of terminals to be equipped with cassettes. Entering text in the timesharing mode is not efficient in line cost.
- Cassettes attachable to office typewriters. If this means new office typewriters, then let them have standard keyboards! By this I mean not only the placement of the printing symbols, but also the placement of the controls, either as separate keys, or in the control position on the regular keys. For example, Control-X is the usual position for CANCEL (deletes the line just typed). Some keyboard designers have not realized that this makes Control-Z a poor place for EOT, because a slip of one position turns off transmission, with resultant loss of all one's work to that point!

With an increased portion of input being generated offline, it would appear that the introduction of the computer at the proper point in the copy production cycle permits entry by *less skilled* people, possibly to the point where the original creator of the text and the enterer are one and the same person. One can imagine an author out in the woods typing his rough copy and getting a cassette record. He would mark up the pages as needed, and send both pages and the cassette to an editing service, which would enter the cassette contents and make online corrections to the author's copy according to his indications.

- Alternatively we would take a CRT display if it corrects certain faults of existing systems in line runaround, etc.
- And perhaps a pointer system that could indicate both the beginning and end of a string to be identified for a working purpose.
- A registry of available digitized symbols, so that one would know where to buy their representations in a transferable form.
- More than any hardware imaginable, we would like to see the development of a common composition language, and its elements, that is, universally-agreed encodings for printed symbols - their graphemes, their placement, and their style. Elements of a proposal follow:

FEASIBILITY OF A COMMON COMPOSITION LANGUAGE

Production of graphic copy from encoded data is an important component for present and future information retrieval systems. Dot matrix characters on a CRT screen will just not be satisfactory for some purposes. Production of graphic hard copy from an information bank may in the future be cheaper than ordering an existing printed reproduction to be invoiced, found, packaged, mailed, and delivered.

Because future information retrieval will consider many more symbols than those of the present ISO Code, existing and future graphic devices must be connectable to the retrieval system.

Equipments that produce hard (or film) copy may be viewed in the same way that we view computer central processors utilizing different instruction sets and object code, and as we view various numerically-controlled machines. There are single programming languages that are common to many central processors. In N/C, the APT language is processed to produce the CL Tape, which is also common to many processing machines. In both cases the common language is processed by computer to produce instruction for specific and multiple equipments. In both cases the translation capability to specific equipment is usually the responsibility of the manufacturer of that equipment. That this is not so in the composition industry is due to the lack of a standard composition language and metarepresentation of text (with associated characteristics of alphabet or other symbol class, font, size, style, weight, and 2-dimensional positioning). If this existed, it would be a high-level language for copy production which is translated, by computer, to instructions for the various hard-copy equipments. The industry suffers from this lack.

To be feasible, the basic functions of copy production must be similar, even if not carried out in the same way. This appears to be so; it has been proved for the Honeywell Computer Journal, which can also be printed from entry terminals. Indention, font change, size change, etc., seem to operate as primitives.

To construct a general text-processing language, of which the composition language is one part, we need to enumerate the functions and then assign standard encodings to them. The provisions to do so exist in the ISO Code and the associated expansion and extension techniques. The most general mechanism is ESCape, although SO and SI exist. Some 2-character ESCape sequences are now virtually standard in the 7-bit code, and will likely be single characters in the 8-bit expanded code. Examples are Half Line Reverse Feed, Cursor Up.

Utilizing code extension procedures, provisions are made to be able to select unambiguously a group of symbols, a font, weight, size, etc. We then use a key device or pressure display panel with single function buttons. The operator would perhaps press "Cyrillic" (to get the GOST Standard encoding), "8" point on "10", "bold". Each key would generate an ESCape sequence in series, inline in the text. He then uses either a special typewriter keyboard, a standard keyboard with a chart of correspondences, or some other device, to enter the Russian text. One can imagine the total set of symbols paged on a microfiche for back projection on a screen.

Computer programs (postprocessors) are created to translate from this standard language into the actual commands and character inputs for the copy device, which could be 6-level Teletypewriter, Monotype, Photon, RCA Page One and Videocomp, Datel typewriter terminals, IBM Selectric Composer, etc.

Until new entry equipment is made available to conform, similar preprocessors could be written to convert from the various entry conventions to the metarepresentation. This would reduce the translations from $N!$ to $2N$. If all entry equipment would eventually conform, a further reduction to N occurs, where N = the number of different composition equipments.

It is expected that this would free the photocomposition industry for expansion in the same way that FORTRAN, COBOL, and ALGOL did so for computational usage. It would provide international standards for alphabet representation, to aid the UNISIST project.

CLASSIFICATION AND GROUPING OF SYMBOLS INTO PAGES

ISO TC46 (International Standards Organization Technical Committee 46), Documentation, has a Subcommittee 4 on Automation in Documentation. This body has responsibility for collecting and/or developing the pages of encoded symbols. Examples of such pages are:

■ Characters to form natural languages (alphabets)

ISO [DIS 646]	Kata Kana [JISCII]
National/accented	Kanji
Cyrillic [GOST 13052-67]	Braille
Greek	Phonetic
Hebrew	Dactylogy [hand signs]
Arabic
Sanskrit	Other punctuation [character augments, bullets, rules, bars, leaders, etc.

■ Symbols of various fields

Aeronautics	Medicine
Astronomy [Astrology]	Meteorology
Biology, Botany	Money
Business [Commerce]	Music
Chemistry	Philately
Ecclesiastic, Fraternal	Pictorial, Ornaments
Electricity, Magnetism	Transportation
Flowcharts	Typography
Games	Welding
Heraldry [flags, insignia, arms]
Logic diagrams	Other Scientific
Mathematics, Geometry, Physics	

■ Controls - for changing point size, weight, slope, font, position relative to the base line, horizontal compression, etc.

An ESCape sequence and prefix character should be proposed for each page of symbols, for registry with ISO TC 97, Computers and Information Processing, which body maintains this registration authority for extension and expansion of the ISO Code.

REFERENCE

1. "TEXT EDITOR Quick Reference Manual, Series 600/6000", Honeywell Information Systems Inc., DB42, 1972 June.



PROFESSIONAL COMMUNICATION

SEPTEMBER 1973

VOLUME PC-16

NUMBER 3

A PUBLICATION OF THE IEEE PROFESSIONAL COMMUNICATION GROUP

SPECIAL ISSUE: RECORD OF THE CONFERENCE ON THE FUTURE OF SCIENTIFIC AND TECHNICAL JOURNALS

PREFACE.....	<i>J. M. Lufkin</i>	49
CONFERENCE SUMMARY.....	<i>A. F. Spilhaus, Jr.</i>	50
SESSION I. EDITORIAL MANAGEMENT		
Problems and Procedures in Editing a Professional Technical Journal.....	<i>G. S. Axelby</i>	54
Editor-Author Relationships: Both Can Win.....	<i>E. Layton</i>	57
The Editor's Role in the Survival of the Broad Technical Journal.....	<i>B. Deacon and D. Hamilton</i>	60
Survival Values in Technical Journals.....	<i>R. Maxwell</i>	64
The Role of Technical Magazine/Journal Hybrids.....	<i>D. H. M. Bowen</i>	66
Problems Related to Journal Subject Coverage, Formats, and Packaging.....	<i>S. W. Terrant</i>	68
LUNCHEON ADDRESS, FIRST DAY		
The Coming Evolution of Scientific Journals.....	<i>P. H. Abelson</i>	69
SESSION II. ECONOMICS AND PRODUCTION		
Financing a Multijournal System in the 1970's.....	<i>J. K. Crum</i>	71
Typewriter Composition Cuts Journal Costs Speeds Publication.....	<i>R. H. Marks and A. W. K. Metzner</i>	73
Impact of Microfilms on Journal Costs.....	<i>J. H. Kuney</i>	80
The Editorial Processing Center.....	<i>H. E. Bamford, Jr.</i>	82
Integrating Primary and Secondary Journals: A Model for the Immediate Future.....	<i>A. W. K. Metzner</i>	84
Integrating Computer Text Processing with Photocomposition.....	<i>R. W. Bemer and A. R. Schriever</i>	92
SESSION III. ALTERNATIVES		
The Engineering Periodical: The Offer You Can't Refuse.....	<i>W. G. Askew</i>	97
AIAA Experiments and Results on SDD, Synoptics, Miniprints, and Related Topics.....	<i>G. L. Dugger, R. F. Bryans, and W. T. Morris, Jr.</i>	100
Separate Article Distribution as an Alternate to Journal Publication.....	<i>D. L. Staiger</i>	107
Scientific and Technical Journals on Microfiche.....	<i>W. G. Bovee</i>	113
The Role of Technical Reports in Scientific and Technical Communication.....	<i>N. Brearley</i>	117
Of Models and Scientific Markets.....	<i>P. W. Hamelman and E. M. Maze</i>	120
LUNCHEON ADDRESS, SECOND DAY		
Irreproducible Results.....	<i>G. H. Scherr</i>	126
SESSION IV. SOCIAL IMPLICATIONS		
The Technical Journal—Who Needs It?.....	<i>H. M. Sarasohn</i>	129
Social Functions of Scientific Communication.....	<i>J. C. Robbins</i>	131
The Publish-or-Perish Syndrome.....	<i>S. R. Searle</i>	136
How Rhetoric Confuses Scientific Issues.....	<i>B. G. Cox and C. G. Roland</i>	140
Some Effects of Delay in Publication of Information in Medical Journals, and Implications for the Future.....	<i>D. A. E. Shephard</i>	143
A Systems Approach to the Management of a Scientific Journal.....	<i>C. W. N. Thompson</i>	148
SESSION V. ROUNDUP AND CRITIQUE.....		
156		
BOOK REVIEW		
<i>Camera-ready</i> , by Kenneth Caird.....	<i>Reviewed by: E. K. Schlesinger</i>	182
CONTRIBUTORS.....		
183		



IEEE PROFESSIONAL COMMUNICATION GROUP

The Professional Communication Group is an organization, within the framework of the IEEE, dedicated to improving the communication of technical information. All members of the IEEE are eligible for membership in the Group and will receive this TRANSACTIONS upon payment of the annual Group membership fee of \$6.00. For information on joining write to the IEEE at the address below.

ADMINISTRATIVE COMMITTEE

THOMAS E. NUNAN, *President*
Burson-Marsteller, Inc., Chicago, Ill.

C. HARKINS, *Secretary*
IBM Corp., Poughkeepsie, N.Y.

JAMES M. LUFKIN, *Vice President*
Honeywell, Inc., Minneapolis, Minn.

ALFRED N. GOLDSMITH, *Adviser*
New York, N.Y.

W. ARROTT, *Treasurer*
Gardner Jones & Co., Chicago, Ill.

Terms Expiring 1973:

R. V. MCGAHEY
Westinghouse Elec. Corp., Pittsburgh, Pa.

E. M. MCELWEE
RCA Commercial Eng., Somerville, N.J.

W. B. DENNEN
RCA Commercial Eng., Somerville, N.J.

R. M. STERN
Publications Eng. Consultants, New York, N.Y.

I. M. SEIDEMAN
RCA Astro-Electron. Div., Princeton, N.J.

J. CHAPLINE
624 West Upsal St., Philadelphia, Pa.

Terms Expiring 1974:

S. F. DIERK
RCA Lab., Princeton, N.J.

W. O. HADLOCK
RCA Corp., Camden, N.J.

C. HARKINS
IBM Corp., Poughkeepsie, N.Y.

I. E. JETER
Bechtel Corp., San Francisco, Calif.

T. E. NUNAN
Burson-Marsteller, Inc., Chicago, Ill.

R. M. WOELFLE
LTV Electrosystems, Inc., Greenville, Tex.

Terms Expiring 1975:

W. ARROTT
Gardner Jones & Co., Chicago, Ill.

D. B. DOBSON
4701 Willard Ave., Chevy Chase, Md.

R. D. EAMES
75 Montvale Rd., Weston, Mass.

T. T. PATTERSON, JR.
170 Newton St., Weston, Mass.

E. K. SCHLESINGER
7 Holmhurst Ave., Baltimore, Md.

M. P. SOUTHWORTH
Pine Mountain Rd., RFD 3, West Redding, Conn.

The Group Committees and the Local Chapter will be found on the inside back cover.

IEEE TRANSACTIONS® ON PROFESSIONAL COMMUNICATION

Editor

I. M. SEIDEMAN
RCA Astro-Electron. Div.
P. O. Box 800
Princeton, N.J. 08540

JOSEPH CHAPLINE
624 W. Upsal St.
Philadelphia, Pa. 19119

Associate Editors

SIEGFRIED DIERK
RCA Lab.
Princeton, N.J. 08540

EDWARD GRAZDA
1524 Brentford Ave.
Westlake Village, Calif. 91360

Associate Editors

GWYN HUNT
IBM Corp.
P. O. Box 12275
Research Triangle Park, N.C. 27709

JAMES LUFKIN
Honeywell, Inc.
2701 Fourth Ave., S.
Minneapolis, Minn. 55408

EMILY SCHLESINGER (Book Reviews)
Baltimore Gas & Elec. Co.
315 Madison St. Bldg.
Baltimore, Md. 21203

Newsletter Editor

RALPH CIAFONE
RCA Lab.
Princeton, N.J. 08540

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Officers

HAROLD CHESTNUT, *President*
J. J. GUARRERA, *Vice President*
SEYMOUR CAMBIAS, JR., *Vice President, Regional Activities*
JOSEPH K. DILLARD, *Vice President, Technical Activities*

ROBERT F. COTELLESA, *Vice President, Publication Activities*
WILLIAM H. THOMPSON, *Secretary*
ARTHUR P. STERN, *Treasurer*

Headquarters Staff

DONALD G. FINK, *Executive Director*

ELWOOD K. GANNETT, *Director, Publishing Services*
WILLIAM J. KEYES, *Director, Administrative Services*
JOHN M. KINN, *Director, Educational Services*
CHARLES F. STEWART, JR., *Director, Member Services*

BETTY J. STILLMAN, *Executive Assistant to the Executive Director*
RICHARD M. EMBERSON, *Director, Technical Services*
EDWIN D. MACDONALD, *Conference Services*
SAVA I. SHERR, *Standards*

Editorial Department

H. JAMES CARTER, *Manager, Editorial Services*

Senior Editors: ANN H. BURGMAYER, CAROLYNE ELENOWITZ, GAIL S. FERENC

Associate Editors: NANCY B. BUDDÉ, HELEN S. GOLDMAN, ROSEMARIE HALVERSON, RUTH MELNICK, NELA RYBOWICZ

IEEE TRANSACTIONS ON PROFESSIONAL COMMUNICATION is published quarterly by The Institute of Electrical and Electronics Engineers, Inc., 345 East 47 Street, New York, N.Y. 10017. Responsibility for the contents rests upon the authors and not upon the IEEE, the Group, or its members. This journal is available in either microfiche or printed form. Individual copies may be purchased at \$3.00 (first copy only) by IEEE members and \$6.00 per copy by nonmembers. Annual subscription price: IEEE members, dues plus Group fee. Price for nonmembers available on request. Abstracting is permitted with mention of source. All rights, including translation, are reserved by the IEEE. Requests for republication permission should be addressed to the IEEE Editorial Department, 345 East 47 Street, New York, N.Y. 10017. Copyright © 1973 by the Institute of Electrical and Electronics Engineers, Inc. Printed in U.S.A. Second-class postage paid at New York, N.Y. and at additional mailing offices.

Integrating Computer Text Processing with Photocomposition

ROBERT W. BEMER and A. RICHARD SHRIVER

[Editor's note: This paper was supplied by the authors in the form of mechanicals for reproduction, photocomposed by the methods described, to match the style of the other papers in this *Transactions*.]

Abstract - Using a computer text processing system as the entry and change vehicle for a photocomposition system affects the publishing function in many ways. Costs are reduced, quality and readability are enhanced, esthetics are more controllable, and entry personnel require little training. Proofreading is almost entirely replaced by a computer-generated concordance. Mechanicals for reproduction are completed at the editor's site, not at the printer's, completing one more step in the movement to the automated office.

The *Honeywell Computer Journal* is published concurrently on hard copy, microfiche, and magnetic tape. The tape can be used to drive other photocomposition systems that differ from our own, just as a computer can translate COBOL programs to the running instructions of a particular computer. Thus our work has shown the way to a common composition language that can describe all formats and identify uniquely the universe of printed symbols.

INTRODUCTION

The introduction of computers to the composition process began in 1961, but not much thinking was applied to the system aspects. This led to some failures and marginal returns. Hyphenation and justification, the earliest uses, are actually trivial. So are text entry and control of character generation.

The challenging functions are page layout, pagination, tabulation, indexing, ruling, proofing, and multiple output from a single file by changing the variables.* However, a danger lies in trying to do these functions automatically by the computer; the amount of difficult programming required often leads to excessive costs, disillusion, and project abandonment - with concomitant prejudice against computers. Shatzkin [1] said:

"The key contribution that the computer can offer the book publisher is very simply this: the predictability of the final result! This may sound very anticlimactic, an absurdly small benefit from such a mighty instrument, but I assure you that predictability can change procedures and even the nature of book publishing in very revolutionary ways."

Our experience in publishing the *Honeywell Computer Journal* has borne this out. As usual, close cooperation between human and computer pays off best. The basic ingredient of our system design is the cost of photocomposition relative to hand or linotype setting, being cheaper by a factor of more than 20. This leads

Manuscript received 1973 June 8. This paper was presented at the 1973 IEEE Conference on the Future of Scientific and Technical Journals, New York, N.Y., May 17-19.

The authors are with Honeywell Information Systems, Inc., Phoenix, AZ 85005.

* For example, this article was photocomposed (by the methods described) in this form and in an alternate form for the 1973 National Computer Conference; the common portions are used with permission of AFIPS.

to the philosophy that we shall always make many photocomposition runs, which governs our procedures from the outset.

SCHEMATIC OF THE SYSTEM

The HIS 6000 system is used for text entry, editing, storage, and running concordances. It is not normally used for the "run-off" function (producing formatted copy on the entry terminal). Even though this feature is available, it is tedious, expensive in line cost, and has little value for final copy.

Formatted copy is produced only by photocomposition. When this is desired, a special postprocessor program converts the text stream and embeds macros for the Page 2 System. This produces a magnetic tape which is (now) transported physically to the facilities of Datagraphics, in Phoenix, AZ, and input to a Univac (nee RCA) 2 driving a III Videocomp 830. The resulting copy is laid up in desired page form, and a cycle of editing and further photocomposition begins.

Final copy is waxed on templates in the traditional manner. Special heads are added (in fonts not available to the computer system, and chosen to symbolize article content, where possible), and it's off to the printers.

Basically, we have adjoined two free-standing systems, and in so doing removed from the middle the expensive and non-graphic-quality output of the first, and the somewhat tedious and inflexible input of the second. Jury-rigged as it is, it is nevertheless superior to any method formerly available to us, and points the way to integrated systems for the future. We can live for now with our 2-hour turnaround.

TEXT ENTRY

Text entry is accomplished in the timesharing mode with the standard HIS 6000 Text Editor System [2], an embedded format system based upon M.I.T. work and similar to the IBM Script. It is *not* a numbered line system like ATS, and eight years of experience has proved *this* wisdom. Searching and alteration are done primarily in the string mode. "Cut and Paste" is limited to operate by the number of lines moved, but they are not themselves numbered.

All control actions are signified by embedded "dot" commands. This input convention states that a CR (Carriage Return) character followed by a full stop character (period, dot) signifies a control statement, thus:

```
.begin      .center    .indent n   .subpara n  .TAB
.space (n) .adjust    .undent n   .para       .break
```

These are but a subset of the standard Text Editor and can be learned by an unskilled person in an hour or so. The editing commands will be explained by examples in the running text of this paper.

PHYSICS AND ASTRONOMY CLASSIFICATION SCHEME - 1973

Summary and Table of Contents

The symbols ▲, ●, ■ in the table below indicate the subject coverage of the three monthly sections of *Current Physics Advance Abstracts* and *Current Physics Titles*, subtitled *Nuclei and Particles* (▲), *Atoms and Waves* (●), and *Solid State* (■).

GENERAL			
1.10*	Education, History, Philosophy and General Interest	▲ ● ■	Page 1
1.20	Metrology	▲ ● ■	1
1.30	Applied Mathematics	▲ ● ■	1
1.40	Classical and Quantum Mechanics	▲ ● ■	2
1.50	Relativity and Gravitation	▲	2
1.60	Statistical Physics and Thermodynamics	▲ ● ■	2
HIGH-ENERGY AND NUCLEAR PHYSICS			
2.10	General Elementary-Particle Theory and Quantum Field Theory	▲	Page 2
2.20	Elementary-Particle Processes	▲	3
2.30	Experiments on Properties of Specific Particles and Resonances, and Status of Fundamental Laws	▲	3
2.40	Cosmic Rays	▲	3
2.50	Nuclear Structure	▲	3
2.55	Nuclear Decay	▲	3
2.60	Nuclear Reactions and Scattering	▲	3
2.70	Properties of Specific Nuclei (Listed by Mass Ranges)	▲	4
2.80	Nuclear Engineering and Nuclear Power Studies	▲	4
2.90	Experimental Methods and Instrumentation for Elementary-Particle and Nuclear Physics	▲	4
ATOMIC, MOLECULAR AND CHEMICAL PHYSICS			
3.10	Atoms	●	Page 4
3.20	Molecules	●	4
3.30	Macromolecules and Polymers	●	5
3.40	Physical Chemistry	●	5
FLUID AND PLASMA PHYSICS			
4.10	Fluid Dynamics	●	Page 5
4.20	Structure and Properties of Fluids	●	6
4.30	Plasma Physics	▲	6
4.35	Quantum Fluids	●	6
4.40	Vacuum Physics, Cryogenics, High-Temperature and High-Pressure Physics	●	7
SOLID-STATE PHYSICS			
5.10	Crystallography and Structural Configuration of Solids	■	Page 7
5.15	Lattice Dynamics	■	7
5.20	Energy States in Solids	■	7
5.25	Mechanical and Acoustical Phenomena	■	7
5.30	Thermal and Thermodynamic Phenomena	■	8
5.35	Transport Phenomena	■	8
5.40	Dielectric and Ferroelectric Phenomena	■	8
5.45	Superconductivity	■	8
5.50	Magnetic Properties of Solids	■	8
5.55	Magnetic Resonance and Relaxation	■	9
5.60	Optical Properties of Solids	● ■	9
5.70	Passage of Particles and Radiation Through Solids	▲ ■	9
5.80	Physics of Surfaces, Thin Films, Interfaces and Small Particles	■	9
5.90	Solid-State Devices	■	10
ACOUSTICS, OPTICS AND CROSS-DISCIPLINARY PHYSICS			
6.10	Acoustics	●	Page 10
6.20	Electromagnetic Technological Applications	●	10
6.30	Optics	●	10
6.40	Geophysics	●	11
6.50	Biophysics	●	11
ASTRONOMY AND ASTROPHYSICS			
7.10	Theoretical Astrophysics	▲	Page 11
7.20	Solar System	▲	11
7.30	Stellar Astronomy	▲	12
7.40	Galactic and Extragalactic Astronomy	▲	12
7.50	Astronomical Measurements (Listed by Type of Observation)	▲	12
7.60	Techniques and Instrumentation for Astronomical Measurements	▲	12

*In the full scheme as printed here beginning in the right-hand column on this page, these classification numbers are suppressed whenever the category is broken down into more detailed subcategories.

1. GENERAL

EDUCATION, HISTORY, PHILOSOPHY AND GENERAL INTEREST

- 1.10.15. Announcements
- 1.10.20. Physics Organizational Activities
- 1.10.30. Reviews of Books, Monographs and Collections
- Education**
 - 1.10.40.20. Curriculum
 - 1.10.40.30. Elementary and Secondary School
 - 1.10.40.40. Teacher Training
 - 1.10.40.60. Teaching Methods and Strategies
 - 1.10.40.90. Other Topics in Education
- Educational Aids**
 - 1.10.50.20. Course Design
 - 1.10.50.30. Audio and Visual Aids
 - 1.10.50.40. Resource Letters
 - 1.10.50.50. Instructional Computer Use
 - 1.10.50.60. Testing Theory and Techniques
 - 1.10.50.65. Demonstration Apparatus
 - 1.10.50.70. Laboratory Apparatus
 - 1.10.50.75. Laboratory Organization, Arts and Techniques
 - 1.10.50.90. Other Topics in Educational Aids
- 1.10.60. Errata
- 1.10.65. Addenda
- 1.10.70. Biographical Notes
- 1.10.75. History of Science
- 1.10.80. Philosophy of Science
- 1.10.85. Science and Society
- 1.10.90. Other Topics of General Interest

METROLOGY

- 1.20.10. General
- 1.20.20. Units
- 1.20.30. Standards
- 1.20.40. Determination and Value of Fundamental Constants
- 1.20.90. Other Topics in Metrology

APPLIED MATHEMATICS

- Algebra**
 - 1.30.10.10. General
 - 1.30.10.20. Algebraic Structures
 - 1.30.10.30. Continuous Groups and Lie Algebras (See also 2.10.25.10, *General Theory of Currents*, and 1.40.50.30, *Group-Theoretical Methods in Quantum Mechanics*)
 - 1.30.10.40. Finite Groups
 - 1.30.10.50. Operator Theory and Matrix Algebra
 - 1.30.10.90. Other Topics in Algebra
- Analysis** (See also 1.30.50, *Numerical Analysis*)
 - 1.30.20.10. General
 - 1.30.20.20. Differential Equations
 - 1.30.20.30. Integral Equations
 - 1.30.20.40. Integrodifferential Equations
 - 1.30.20.50. Special Functions of Physics
 - 1.30.20.60. Potential Theory and Harmonic Functions
 - 1.30.20.70. Complex Analysis
 - 1.30.20.80. Functional Transforms
 - 1.30.20.90. Other Topics in Analysis
- Geometry** (See also 1.50, *Relativity and Gravitation*)
 - 1.30.30.10. General

Fig. 2. First page of AIP's PACS.

ENTRY FOR UNSKILLED PERSONNEL

The postprocessor program that converts for the Page 2 System is vital for simple text entry. The standard entry methods for the Page 2 System are certainly *not* simple and require some training and a crib sheet constantly on display to the enterer. Remember that graphic quality output requires a separate font generation for each unique character. It is not sufficient to overprint an umlaut (in its fixed position) for both the upper and lower case "u", for example. The postprocessor does extensive string analysis, much of it based upon backspace and overstrike for entry, which makes it simple for personnel. Examples:

- Characters with diacritical marks (accent acute, accent grave, tilde, umlaut, etc.) are produced by backspace on the terminal and overstrike with the proper character (double quote is used for umlaut).
- Double and single quotes are used as they are for entry. The postprocessor determines whether they are opening or closing quotes. A double quote is two single quotes in photocomposition, and this is called automatically.
- For minor occurrences in text, boldface may be indicated by overstriking single characters three times. This is visible on the terminal when the line is verified. For a longer string of bold characters, the font is altered by a **.bold** command and turned off by a **.bold end** command. These commands do not force a new line.
- For minor occurrences in text, italics may be indicated by backspacing the length of the word and underlining. This is visible on the terminal when the line is verified. For a longer string of italics, the font is altered by a **.ital** command and turned off by a **.ital end** command. These commands do not force a new line.
- The bulleting seen here is accomplished by a **.indent 3** followed by a **.undent 3** (which is operative only for the next line), a lower case "oh", 2 blanks, and then the text. The uniqueness of this string permits the convention.
- To the regular Text Editor convention of using the "at" symbol to delete the previous character (guess why our articles never contain this character!), and CAN to delete the entire line of entry, we have added the caret to indicate the end space, which is incompressible to the justification process. Thus a new paragraph is caused by a **.break** and an initial line with two carets for indention.
- The normal font sizes for *Honeywell Computer Journal* are:
 - 9 point - text
 - 8 point - references, some displays as necessary
 - 7 point - sub- and superscripts, figure captions

Point size may be changed at any point in the text by inserting the ESCape sequence:

ESC g (7-pt), ESC h (8-pt), ESC i (9-pt)

These override the original settings, and are used for formulas, etc.

CONTROL OF PAGE LAYOUT

It has been a remarkable discovery to us that reader attraction and satisfaction is increased significantly by tight control of page layout. Only in the most exceptional cases will a column start in the middle of a sentence, and then only on the second column of the same page. Usually a column will start with at least a paragraph (not just an arbitrary paragraph, but one that makes sense), and very often with a heading. The appearance of a figure or table will never precede its first mention in text, nor will it often be on a page that is not visible when that mention is made. "Widows" never occur.

Under traditional methods, the editor loses control of page layout after the galley stage; all of the niceties must be left to a composer who has little understanding of the subject matter, and is often less interested in reader satisfaction. With the low cost of text processing taken in conjunction with photocomposition, we do not mind expending many runs to get just what we want.

A quick reading of the first galley copy gives an estimate of the author's redundancy or flowery speech factor and other ways that compression can be achieved if necessary. Accordingly, the actual film is cut to lay out an approximation of the article. As the last page is always full, we work backward. Whatever is left for the first page we leave for artistic treatment and the "From the Editor" commentary. Great attention is paid to aspects of future readability, left or right page assignment, pleasing placement of tables, figures, and photos. Virtually no attention is paid to typos and other mistakes that exist in the copy. Accordingly, the single columns are taped on with more lines than our standard, trusting to judicious editing to cut back to the right number (60).

The beauty of this system is that many things can be changed simultaneously to create correctness, harmony, and interest: point size for certain paragraphs or tables, tab settings, subparagraphing, font style, and text changes and corrections. Imagine a situation where the column copy has to be reduced by two lines, and yet previous editing has taken advantage of all short lines at the end of paragraphs, filler words have been removed, and big words replaced by commoner smaller words with equivalent or clearer meaning. Now you have to get into the guts of the author's meaning and say it shorter and clearer, without altering the flavor or meaning in any way! Being forced to do this by our aesthetic standards for page layout yields a big dividend in increased readability.

Depending upon the content, we may photocompose the text from 2 to 5 times. Do the authors complain about the alterations? Never, in our experience. When it reads well, they just assume that they wrote it that way, never checking their original copy. We have also experimented in putting the author's work in to typeset even when it is only rough draft; results seem to indicate that the visualization of final copy permits him to improve it more than he could by editing from a typed draft.

Obviously, taking this much work for readability means high acceptance standards, and we insist that this is a good thing. Dung coated with 53 layers of Chinese lacquer is still dung, and we do not intend contributing to information pollution.

Hopefully, it is now clear why we do not use the computer for automatic pagination.

PROOFING OF COPY

An optional feature, or byproduct, is the concordance run, usually exercised on what is expected to be the next-to-last photocomposition run. This produces two listings on the high-speed (upper case only) printer. The first listing is a Key Word Out of Context (KWOC) listing; each numeral and word (except for the very small common ones) is listed on the left in collating sequence order, with its entire entry line on the right. The lines are numbered here, for cross-reference to the second listing, which is the consecutive text.

The concordance is now scanned visually, primarily to detect input errors ("typos"). See Figure 1 for some examples. It is our experience that these fairly jump out at one in scanning a concordance, whereas they remain stubbornly glossed over by the eye and mind in traditional proofreading. However, we do read the text - for style and making sense, not for typos. In fact, knowing that you are freed from the typo-hunting task creates a different frame of mind for doing *real* editorial work.

AUTOMOHILE	DISPOSITION	MASSACHUSETS
AUTOMOHILE	DISRRUPTION	MASSACHUSI TTS
AUTOMOHILIES	DISRUPT	MASSACHUSETTS
AVAILABILITY	DISRUPTIONS	MASSES
CERTAINLY	INSTEAD	SOLVED
CERTAINTY	INSTEAD	SOLVEU
CERTIFICATION	INSTITUTE	SOLVEUTO
CERTIFICATE	INSTITUTE	SOLVED,
CERTIFICATION	INSTITUTE	SOLVE,
CHARGES	KURT	STEREOSCOPIC
CHARGES	KY,	STEREOTYPED
CHARIMAN	LO	STEREOTYPED
CHARITY	LA	STERODGRAPHIC
CHARLATANS	LAB	STEROTYPE
		STEWARDS
COLUMHIA	LOCATIONS	SUCCESS
COLUMHIA	LOCATION,	SUCCESS
COHAPNY	LOCATION,	SUCCESSING
COMBINATION	LOCATIOXS	SUCCESSOR
COMBINATION	LOGGED	SUCH

Figure 1. Typos exposed by concordance.

CONTROL OF READABILITY AND STYLE

The concordance produces a histogram of word size distribution as a byproduct, and the average word length may be calculated. We target 5.0 characters per word, and are very suspicious of readability when the author gets above 5.5.

One aspect of style, or rather one of our rules, is that an acronym shall always be given the spelled-out version in parentheses the *first* time it is encountered in text. One has only to spot the first occurrence in the concordance and look to the corresponding line on the right to see if this has been done. If not, edit.

The *Honeywell Computer Journal* has other style rules. Most important is adherence to ISO Standard 1000, or the International System of Units (SI). Check the concordance for inches, feet, yards, miles, pounds, etc. If they occur, and are for measurement, they had better be in parentheses following a metric value. Other examples are: \$2 million - not 2 million dollars; 0.5 s - not .5 sec; focused - not focussed.

ECONOMIC CONSIDERATIONS
IN WORKING METHODS

As there is no way to predict the pagination of printed copy when entering text, one could enter it all under a single file name. However, the 6000 Text Editor keeps the entire file in the main store for faster processing (and it is *really* fast), and these facilities must be paid for. Thus original input is made in judiciously separated and named files, breaking at headed sections, for example. These are then adjoined for the photocomposition run.

After page layout is determined, they are adjoined again and resplit by page into files with new names, and the old ones purged. This permits single columns to be reworked into final form. The present rate is \$1.75 per column. Thus a page costs from \$6 to \$10 to compose, comparing rather favorably with the \$70 per page we were paying for linotype setting to our standards before our system was operable. The 6000 cost is not included, as we have been unable to get real figures because we work on an inhouse "exposure" system used for checking out new software releases. We do, however, feel that this cost is compensated by the system doing automatically what we would have to do ourselves otherwise (like proofreading), and by the added quality. We do need to modify our programs in order to be able to set double column on the last run.

Economy dictates that we should process as much text as possible on each photocomposition run. This means linking several files and saving them as a single file. But this increases the risk that something going wrong early will spoil the balance. Care must be taken to separate and insulate each file from any other. Convention starts each file with `.begin` (for a new galley), `.indent 0` (in case the file ahead of it lacked a command to restore indentation to 0), and `.adju` (in case the preceding file had been using tabulation and was not restored to the justification mode).

The power of the Text Editor is of great assistance in checking for correctness of the adjoined file, particularly for closure. Type:

```
fs:/bold/* (meaning "find all occurrences of that string")
```

and you will almost instantly get a message like:

```
end of file - request executed 122 times
```

Hit "b" and CR (for backup to the file beginning), and type:

```
fs:/bold end/*
```

If the message doesn't say 61 times - trouble! A 60 would mean that bold did not get turned off somewhere, and the copy following will be in useless boldface. Do the same for italics, subparagraphs, point size changes, etc.

The files must always be correct for the magnetic tape edition, and identical to the printed copy. Yet it is often wasteful to rerun the entire file for simple patches. A copy is made, and the correct parts wiped out by string replacement, leaving only the changed copy to be reset as a patch (with due consideration to leaving enough text so that paragraphing, etc., is unchanged). These patches are saved under a different name; a number of them are adjoined and run at one time.

INCIDENTAL ADVANTAGES

A number of dividends have shown up that we amateurs did not really foresee:

- Doing our own typesetting permits laying up mechanicals for articles as soon as they are ready, without waiting to group an entire issue for the typesetter to schedule in some time slot. Exclusive of conditions of extreme timeliness, this permits better selection for issue makeup and content.
- Having the feel of the final product, by mockup during the editing and changing stages, affects everyone - author, editor, and reviewer. For the latter, particularly, it gives psychological impetus to hurry up - lest what he dislikes might be in the finished product. All can work simultaneously to correct and improve the copy and make it more readable.
- The Page 2 System hyphenates to English rules and/or custom. Normally we run our French, German, Italian, and Spanish sections in "fill mode" (stretching the spacing between words to fill the line without hyphenation). But if glaring gaps exist we remove them easily by doing a dummy hyphenation, splitting the first word of the next line into two components:

```
rs:/whippersnapper/
ENTER
*whipper- snapper
*
READY
```

This technique can also be used in our English text when Page 2 fails to hyphenate opportunely or (rarely) incorrectly.

On one occasion the entire article was side-by-side in both German and English. Here we could proceed more elaborately, removing Page 2 hyphenation that was incorrect for German, forcing correct hyphenation paragraph by paragraph.

- Page 2 also has the flaw of assuming that a change in font style permits a break for a new line just as hyphenation or a space does:

```
..... Protection A
agency ...
```

Text Editor can force a correction by replacing sufficient spaces between words by incompressible en spaces.

- We don't have to worry about losing corrected galley in the mails, as the *Journal of the Association for Computing Machinery* did in 1971 October. We also know that the corrections have actually been made in the printer's copy, without waiting for a blue to be returned and show that they were *not* made. This often shortens the production cycle, and certainly cuts costs.
- Secretaries can make very creditable copy inhouse by cutting and pasting galley segments with Scotch Tape, and then using a reproduction method such as Multilith. Interoffice memos are becoming artistic, easier and pleasanter to read, and certainly use less paper.

OUR WISH LIST

End users should tell suppliers the nature of their applications and what they would like to have to do these applications better, cheaper, and faster. We would like:

- A larger portion of terminals to be equipped with cassettes. Entering text in the timesharing mode is not efficient in line cost.
- Cassettes attachable to office typewriters. If this means new office typewriters, then let them have standard keyboards! By this is meant that not only the placement of the printing symbols, but also the placement of the controls, either as separate keys, or in the control position on the regular keys. For example, Control-X is the usual position for CANCEL (deletes the line just typed). Some keyboard designers have not realized that this makes Control-Z a poor place for EOT, because a slip of one position turns off transmission, with resultant loss of all one's work to that point!

With an increased portion of input being generated offline, it would appear that the introduction of the computer at the proper point in the copy production cycle permits entry by *less skilled* people, possibly to the point where the original creator of the text and the enterer are one and the same person. One can imagine an author out in the woods typing his rough copy and getting a cassette record. He would mark up the pages as needed and send both pages and the cassette to an editing service, which would enter the cassette contents and make online corrections to the author's copy according to his indications.

- Alternatively we would take a CRT display if it corrects certain faults of existing systems in line turnaround, etc.
- And perhaps a pointer system that could indicate both the beginning and end of a string to be identified for a working purpose.
- A registry of available digitized symbols, so that one would know where to buy their representations in a transferable form.
- More than any hardware imaginable, we would like to see the development of a common composition language, and its elements, that is, universally-agreed encodings for printed symbols - their graphemes, their placement, and their style. Elements of a proposal follow:

FEASIBILITY OF A COMMON COMPOSITION LANGUAGE

Production of graphic copy from encoded data is an important component for present and future information retrieval systems. Dot matrix characters on a CRT screen will just not be satisfactory for some purposes. Production of graphic hard copy from an information bank may in the future be cheaper than ordering an existing printed reproduction to be invoiced, found, packaged, mailed, and delivered.

Because future information retrieval will consider many more symbols than those of the present ISO Code, existing and future graphic devices must be connectable to the retrieval system.

Equipments that produce hard (or film) copy may be viewed in the same way that we view computer central processors utilizing different instruction sets and object code, and as we view various numerically-controlled machines. There are single programming languages that are common to many central processors. In N/C, the APT language is processed to produce the CL Tape, which is also common to many processing machines. In both cases the common language is processed by computer to produce instruction for specific and multiple equipments. In both cases the translation capability to specific equipment is usually the responsibility of the manufacturer of that equipment. That this is not so in the composition industry is due to the lack of a standard composition language and metarepresentation of text (with associated characteristics of alphabet or other symbol class, font, size, style, weight, and 2-dimensional positioning). If this existed, it would be a high-level language for copy production which is translated, by computer, to instructions for the various hard-copy equipments. The industry suffers from this lack.

To be feasible, the basic functions of copy production must be similar, even if not carried out in the same way. This appears to be so; it has been proved for the Honeywell Computer Journal, which can also be printed from entry terminals. Indentation, font change, size change, etc., seem to operate as primitives.

To construct a general text-processing language, of which the composition language is one part, we need to enumerate the functions and then assign standard encodings to them. The provisions to do so exist in the ISO Code and the associated expansion and extension techniques. The most general mechanism is ESCape, although SO and SI exist. Some 2-character ESCape sequences are now virtually standard in the 7-bit code, and will likely be single characters in the 8-bit expanded code. Examples are Half Line Reverse Feed, Cursor Up.

Utilizing code extension procedures, provisions are made to be able to select unambiguously a group of symbols, a font, weight, size, etc. We then use a key device or pressure display panel with single function buttons. The operator would perhaps press "Cyrillic" (to get the GOST Standard encoding), "8" point on "10", "bold". Each key would generate an ESCape sequence in series, inline in the text. He then uses either a special typewriter keyboard, a standard keyboard with a chart of correspondences, or some other device, to enter the Russian text. One can imagine the total set of symbols paged on a microfiche for back projection on a screen.

Computer programs (postprocessors) are created to translate from this standard language into the actual commands and character inputs for the copy device, which could be 6-level Teletypewriter, Monotype, Photon, RCA Page One and Videocomp, Datel typewriter terminals, IBM Selectric Composer, etc.

Until new entry equipment is made available to conform, similar preprocessors could be written to convert from the various entry conventions to the metarepresentation. This would reduce the translations from $N!$ to $2N$. If all entry equipment would eventually conform, then a further reduction to N occurs, where: N = the number of different composition equipments.

It is expected that this would free the photocomposition industry for expansion in the same way that FORTRAN, COBOL, and ALGOL did so for computational usage. It would provide international standards for alphabet representation.

CLASSIFICATION AND GROUPING OF SYMBOLS INTO PAGES

ISO TC46 (International Standards Organization Technical Committee 46), Documentation, has a Subcommittee 4 on Automation in Documentation. This body has responsibility for collecting and/or developing the pages of encoded symbols. Examples of such pages are:

■ Characters to form natural languages (alphabets)

ISO [DIS 646]	Kata Kana [JISCII]
National/accented	Kanji
Cyrillic [GOST 13052-67]	Phonetic
Greek	Dactylogy [hand signs]
Hebrew
Arabic	Other punctuation [character
Sanskrit	augments, bullets, rules,
Braille	bars, leaders, etc.]

■ Symbols of various fields

Aeronautics	Medicine
Astronomy [Astrology]	Meteorology
Biology, Botany	Money
Business [Commerce]	Music
Chemistry	Philately
Ecclesiastic, Fraternal	Pictorial, Ornaments
Electricity, Magnetism	Transportation
Flowcharts	Typography
Games	Welding
Heraldry [flags, insignia, arms]
Logic diagrams	Other Scientific
Mathematics, Geometry, Physics	

■ Controls for changing point size, weight, slope, font, position relative to the base line, horizontal compression, etc.

An ESCape sequence and prefix character should be proposed for each page of symbols, for registry with ISO TC 97, Computers and Information Processing, which body maintains this registration authority for extension and expansion of the ISO Code.

REFERENCES

- [1] R. W. Bemer, Ed., *Computers and Crisis*, Association for Computing Machinery, New York: (1971) 401 pp.
- [2] *TEXT EDITOR Quick Reference Manual, Series 600/6000*, Honeywell Information Systems, Inc., Waltham, MA, DB42, 1972 June.

The Frictional Interface Between Computers and Society

Dr. Robert W. Berner
Honeywell Information Systems
Phoenix, AZ USA

From

EDMUND C. BERKELEY

815 Washington St.
Newtonville, Mass. 02160

JAN 11 1975

Dear Bob,

Thanks very much.

edB —

Editor

For purposes of this article I propose a simple and perhaps novel classification of computer applications -- in three classes.

The Computer Advises

Applications that do not lead to decisions affecting humans directly -- Examples come largely from the field of numerical computation, the earliest category of usage. Computational results that might tend to prove or lead to a theory; calculations for spaceship or missile design (they don't have to be built or launched); programs for playing games, or associating payoffs with strategies, etc. We may term such computation "advisory."

The Computer Proposes Decisions

Applications with computational results that lead to decisions by humans -- Some of these can get very close to integration into human affairs. For example, someone may be denied credit or refused an employment opportunity. It has turned out, in much practice, that the human decision to be taken may be perfunctory or mindless. Nevertheless there is recourse, no matter how time-consuming and difficult it may be, and regardless of what body of law may need to be enacted to protect people in such circumstances.

The Computer Decides, and if not Countermanded Acts (the Robot)

Applications where the computer has been previously programmed to take a decision and take an action, and will in fact act unless countermanded in time -- Examples are online patient monitoring, control of nuclear power plants, air traffic control and collision avoidance systems, automatic transportation systems (i.e., BART, in San Francisco), and automobile braking and antiskid systems.

Tremendous Increase of Robots

The hardware developments of about the last three years leading to microprocessors on chips, portend a tremendous increase in the third class of application. And this is why we must be on guard as to the propriety and systems aspects of such applications. Applied to automobiles, such applications could be extremely critical. One is reminded of power-steering, a boon when it operates, perhaps, but a definite danger when power fails or is turned off.

A pair of questions indicates a possible dilemma:

- Q: Does technology exist to integrate computer components very closely into human affairs?
- A: Yes. For an example, see the 1974 US automobiles, which will not operate unless seat belts are fastened.
- Q: Are system design and good practice manuals available for such a level of technology, and/or is suitable indoctrination and education available in our educational institutions?
- A: Emphatically NO! This fact is frightening enough to suggest a moratorium on such developments until we understand the tool better.

Inoperable Robots

Consider the announcement of an experimental device which requires matching a certain procedure before you can start your automobile. The intent, and certainly an obvious usage, is to preclude drunken drivers from operating vehicles. But suppose that you are extremely shaken because your wife has just been killed, and your child needs to be taken to the hospital. Could you start the car then?

Or consider the case of online patient-monitoring reported in Datamation magazine of 1972 October. The programming was correct but the computer was not 100% reliable. This, as we know, is taken care of by having a customer engineer to fix it. But nobody remembered to find out whether the customer engineers would always be available over the weekend, and speedily. As reported, a patient died because confusion in the human system caused the computer to remain inoperable.

Space Effort Experience in Overriding Robots

Certainly the US space effort has gathered ample experience in the matter of letting computers decide, when they are capable of it, and of overriding them sensibly when it is shown that they were programmed incorrectly or without consideration for all eventualities and malfunctions. We see many spin-offs from the space effort with respect to products, but very little in methodology which could be so very applicable to computer usage.

Based on a presentation to the NordData Conference, Copenhagen, Denmark, August 1973.

These are examples of the brevity and perspicuity of mathematical language, which is a forerunner of much of the language used for describing algorithms to be used on computers.

Mathematics is very largely a language that has three special properties. First, its chief subject matter is certain kinds of abstract elements -- numbers, lines, points, etc. -- and the relations of these elements. Second, its chief property is that it makes use of very efficient symbols, symbols that one can calculate with to a vast extent. Third, its chief purpose is to determine necessary consequences from given assumptions.

All these properties are of course present to some extent in ordinary language. In ordinary language, you can talk to some extent about numbers, lines, points, etc., and their relations. In ordinary language, you can even calculate to some extent. And, using ordinary language, you are often interested in deducing the necessary consequences of some suppositions. Accordingly the difference between mathematics and ordinary language is partly a difference of quality, but also a difference of degree. It is reasonable to believe that if you have a desirable property in mathematical language, you may be able to arrange the same desirable property in language.

We have mentioned calculation. What do we mean by it? We mean that we can manipulate symbols according to rules that pay little attention to meanings, and that when we come out with results at the end of the process of manipulation, then our symbols often apply truthfully to real situations. People can and do calculate with ordinary words, even when they are not sure of the meanings of these words. Here is an example taken from "Language and Communication" by George A. Miller:

Suppose we learn on good authority that all mantelops are lespeads and that all lespeads hile. We can conclude immediately that all mantelops hile, and that any grimpet that does not hile certainly is not a mantelop. There may of course be lespeads that aren't mantelops, so hiling is not a sure sign of manteloplicity.

Then he says:

The fact that we have no idea what we are talking about does not stop us from talking. We simply operate on names and properties according to the rules governing the use of logical terms.

And he might very well add, though he did not say so, that the reasoning is perfect, and that all we have to do to get truthful uses of the statements about "mantelops, lespeads, hiling, grimpets" is to interpret the words in suitable ways.

5. How to Improve the Designation of Meaning

How shall we improve the designation of meaning? and the description of reality so that we can more faithfully describe it and deal with it?

Of course this is an old problem. Many people work on it both consciously and unconsciously from time to time. A person who coins a new and useful expression is helping in this process. In the United States some time in the 1830's the expression "OK" was coined. No authority is quite sure of the derivation. Now the word is internationally used, and

everywhere understood and accepted. It filled a need. And it is much clearer than "yes" and "all right."

There is continual competition in language between different expressions for closely related ideas. The competition often produces two words where there was just one before. "Of" and "off" for example are both derived from the same word. Differentiation is the name given to the linguistic process.

The movement for the liberation of women has produced the word "Ms." (pronounced "miz") so that a woman can be addressed politely without the designation of whether or not she is married. This is obviously a useful and democratic improvement and probably is permanent. Correspondingly, "spokesman" and "chairman" have sprouted the words "spokesperson" and "chairperson." Whether these three-syllable words offered in place of two-syllable words will survive is a question; but the words might survive.

Perhaps the most important process for improving the designation of meaning is paraphrasing, translating from one set of words, often with highly colored overtones, into words that are much more neutral and avoid those overtones. "Shit" becomes "excrement". "Pigheaded fool" becomes "stubborn and foolish person". "The morgue" of a hospital becomes "Ward X".

Paraphrasing -- translating expressions into other expressions that "say the same thing but say it better" -- is a key process.

A computer can do paraphrasing. When a program is written in FORTRAN, the computer by means of what is called a FORTRAN compiler can paraphrase that program into a machine language program that will enable that computer to solve that type or problem.

The program "DWIM," "Do What I Mean," mentioned above can do paraphrasing. It will accept many variations of computer input from a human being and convert them all into a single standard computer input. This is the early seed of a great development in computer applications.

We are like fish swimming in an ocean of natural language. We are thoroughly immersed in that ocean, and often blind and ignorant about what lies outside of the part of the ocean that we happen to know.

The development of computer paraphrasing of expressions, statements and discourse in ordinary natural language will take us a long way. Like lung fish we shall climb out of the ocean, and begin to obtain a much greater and more correct picture of the world, eventually including land, atmosphere, and space.

REFERENCES

1. Barnhart, Clarence L., "The Barnhart Dictionary of New English since 1963", Harper & Row, New York, NY, 1973, 510 pp.
2. Berkeley, Edmund C., "The Personality of the Interactive Programmed Computer," in "Computers and Automation," Dec., 1969
3. Berkeley, Edmund C., Andy Langer, and Casper Otten, "Computer Programming Using Natural Language," in "Computers and Automation" or "Computers and People": Part 1, Jun, 1973, p.10; Part 2, Jul, 1973, p.18; Part 3, Aug, 1973, p.28; Part 4, Aug, 1974, p.2; Part 5, Oct, 1974, p.30

(please turn to page 19)

Computers as a Boon

Now, I like computers. I believe that they are presently more beneficial than harmful to society, and that this ratio can be increased if we take careful consideration and plan for their best and proper usage.

If I were fatalistic, I should feel that they have arrived just in time to save us from our enemies, who are ourselves. In 25 years as a programmer I have never faced a day of working with computers without pleasant anticipation.

I also like a fire in the fireplace, but not arson. Both fire and computers are tools accessible to all of society in some form, and society uses such basic tools in many ways, some deemed good and some bad.

Fire was an early tool, useful for hollowing out logs to make vessels, to make transformations in food, and to heat enclosed air. It was also used to burn vegetation and trees, sometimes accidentally (which was thought bad) and sometimes deliberately, to clear for planting (which was thought good).

Side-Effects, Unanticipated

A major difficulty in analyzing the contribution of a tool is the inability to categorize, in an absolute way, its uses as being good or bad. This is not philosophical, but only to remind us that we make these judgments of good and bad in the narrow context of our mores and morals, which are in turn conditioned by our accumulated knowledge and analysis of the workings of our world. We have learned a little more of those workings lately, not because we sought the knowledge so much as because it has been made painfully evident to us that there is more or less coupling between all the elements of our world.

I quote from an interview with Dr. Carl Hammer of Univac, regarding a conversation with V. A. Trapeznikov, acting Co-Chairman of the United States and Russian Joint Commission on Scientific and Technical Cooperation: "He told me, as he told President Nixon one day earlier, that 'we all must cease to make wrong decisions on a large scale because mankind can no longer afford it. Mankind's resources are highly limited, and we can no longer squander them' ... we must develop not only national but international models for improving our decisionmaking processes. Decisions which at this time are made on a political or emotional base, neither way will produce optimal results."

So I touch on some bad uses of computers only to illustrate the problems to overcome by legislation, education, and professionalism to make computers serve us better.

Mind-Amplifying Factor

Carl Hammer says "We have already built into our society a mind-amplifying factor of 2000 to one. Behind every man, woman, and child in this country (the US), there stands the power of 2000 human beings. The responsibility of any data processing manager of today, of the computer scientists ... is so enormous that even I cannot envision it. It is the greatest challenge that has ever faced mankind."

Power it is, in elemental form. IBM's recent advertising stresses "think of the computer as energy." Theoretically, the computer is vast power at the

service of people, to be used as the imagination of the people leads it, subject of course to limiting legislation.

Knowledge Power

But let us not be lulled by any advertising into thinking that the energy is just like electricity. Computer power is work power, but it is also knowledge power, of the kind that has been used throughout history for aggrandisement as well as the good of the people. In a time when technology stands at bay, it will be well to consider the dangers of computer misuse in prejudicing the population against a valuable tool, and of misuse by corrupt or ignorant officials.

There are no known instances of computers voluntarily stopping normal work to perform illegal acts without direction by humans. Consider the science fiction capability of walking through matter; we have seen it in the cinema, usually used to get into the bank vault or perform some other evil deed. But in the cinema it was a power accorded only to a few, being so technically difficult. Computer power is available widely, and we must not be surprised that some people should turn it to their own ends in disregard of the general benefit of society.

Jerry Schneider: Crook, Then Security Consultant

Consider the case of Jerry Schneider. There is no problem with mentioning his activities. He sent an abstract of a paper that he wanted to present at the 1973 National Computer Conference, telling about how he tapped into a computerized ordering system and stole something like \$1 million of telephone equipment by having it delivered to a telephone company van bought at auction. The computer program, not knowing how to bill and get payment, ignored it as being within loss limits. When turned in by an employee, Schneider spent two months in jail and was back in business as a computer security consultant!

There is no question but that computer power may be abused by individuals. It may be so used by larger entities, such as corporations, to fool or defraud. It may be so used even by governments, however wittingly.

Redressing the Balance

Dr. Henry Bruck of M.I.T. spoke of this at ACM 70, in a talk entitled "To Redress the Balance." His thesis was that computers, because of cost and training investment, were more likely to become the tools of government and big business than the general public. Countering the argument that minicomputers, microcomputers, and hand calculators are available to individuals at low cost, he said that it was a fallacy to assume that this meant that computer power was available to the general public for this reason. Shovels for a penny are useless unless one knows how to dig, and has arms. It is the usage skill that is important.

He thought that modifying education so that imparting basic computer skills (and problem-solving techniques) would be given as much emphasis as learning one's own language would be unnecessary overspecialization. Nor would the answer be to reduce usage by government and business, for we have ever more need for decisionmaking information that is more likely to be accurate and complete, taking into account the overall advantage to people. However, he saw no reason why computer services could not be provided to the citizenry through public institutions.

I agree. There are many opportunities for computer services to be provided by municipalities and/or private ventures. One can imagine data banks that could serve as advisories for human action and choices. There is an experiment in Los Angeles where the computer serves as a general counselor for a multitude of services. Consumerism could be served in a great many ways -- product safety and efficiency, comparative shopping, financing aid for major purchases, reminders for preventive maintenance, etc.

Thus there are many ways to redress the balance by making computer power really available to everyone in a direct manner and without having to learn how to program. There is a need, however, for a certain amount of "computer literacy" in order to feel comfortable with such usage.

Mystique of Computer Authority

As a tool, the computer has become commonplace with a rapidity exceeded by no other, even the automobile. This has caused some disallocation and unease, which the practitioners have not been able to avoid. Most major tools, when introduced, have had their custodians, and then their guilds or professions that, from gradual experience, added to the body of law and practice those safeguards for usage that appeared necessary from gradual occurrences of misuse.

This did not occur with computers, and perhaps we did not even use the time that was available to us, so caught up were we with the mystique and power. Certainly we did not familiarize people generally with computers; instead, they were publicized as "giant brains," and the mystique grew into "authoritativeness."

One of the main problems with authority is that it can be blamed. Surely you all know many examples, but I shall add a few to your knowledge.

Perhaps it is a worldwide phenomenon. One calls the store that has made a mistake in the bill, the bank that has not returned the cancelled checks, the association that has blacklisted your credit -- and the voice replies "I'm sorry, sir, but we have a computer now ...".

The Allen Piano Co. Lie

The Allen Piano and Organ Company of Phoenix advertised by radio that its computer had made a mistake in ordering inventory; they were now overstocked and were therefore holding a sale. I wrote the company a letter, on behalf of the Association for Computing Machinery, offering to fix the computer or program so it would not make such a mistake anymore, on condition that "if it developed that a human was at fault," and not a computer, they would so acknowledge this in their subsequent broadcast advertising. Datamation magazine followed the story -- it turned out that the Allen Piano and Organ Company DID NOT HAVE a computer, nor did they use any computer facilities.

Note the convictions of the advertisers that a computer would give authority to their spurious claim of overstocking.

The No-Apostrophe Lie

One Mr. D'Unger, not of the computer community, wrote to several companies maintaining mailing lists containing his name, either for billing or solicitation, asking them to please spell it correctly. Not

DUNGER, and not D UNGER, and not Dunger (for those with lower case capability). He received several replies, all saying that it was unfortunately impossible with their computer equipment. Learning of this from his letter to Computerworld, I called several of these data processing departments, to find in each case that the print chain was in fact an IBM chain that did have the apostrophe on it, but that they had not bothered to use it! It seems to me that a man's name is a dear possession, and not one to be treated cavalierly under cloak of computer authority.

The Bank Lie

I once visited a home where four elderly women were playing bridge. When they found out that I was in the computer profession there was a chorus of horror stories. Then one brought out a letter from her bank, with a handwritten apology from the teller for the shortcomings of the computer. I was on the spot. To save face I called the bank vice president to see what could be done. They didn't have a computer either!

The Authority of the Computer as an Accomplice

The computer is a convenient means of implicitly or explicitly covering activities that run from illegal to self-serving, intentional or unintentional.

The notorious Equity Funding scandal will certainly become a classic, even though the exact ways that it was perpetrated will take some time to discover. We know, even now, that it was a pyramiding operation, and that computers were used to give authority and extra layers of protection from discovery. Many corrective actions could arise from the case, such as new emphasis on EDP auditing. It appears that perhaps as many as 200 people were involved in collusion.

The University of Michigan has a research service that projects the effect of various decisions and actions upon the GNP (Gross National Product) and its growth, with respect to the State of Michigan. The results could easily be given in regular typewritten (or typeset) reports, but they are not! A computer printout accompanies the report to give it AUTHORITY. The set of results that I saw seemed both spurious and misleading, and perhaps others could have detected this had they been as unawed by computers as I am.

Perhaps there may come a day when the US augments its Environmental Protection Agency with a Human Protection Agency. Then, taking the lead from the present requirement to make notification on cigarette packages that "cigarette smoking is dangerous to your health," it could order that each computer-printed page be preceded by:

"WARNING -- these answers were produced by a computer, and could be hazardous to your health!"

Of course I am being facetious about the overkill which does not seem to diminish smoking anyway, but I do recall the following case.

"Sorry General, Three Years of Wrong Answers"

Univac was attempting to sell the US Army an 1107. The benchmark process included a compilation and run of a certain FORTRAN program. The 1107 compiler printed a diagnostic indicating an entry into the middle of a DO loop. The General in charge indicated that

this was impossible, as they had been running that same program for three years, and asked a programmer to examine the situation. He returned in a short while and said "Sorry, General. Three years of long answers."

The Computer as a Sewage System

A well-known truism of computer usage is "Garbage in, Garbage Out." But what happens when we put perfectly valid data in? Can we get it out again? Can someone else do so? If it does come out, is it legible?

We still live in the computer era where 90% or more of the data depends entirely upon the associated program to be turned into information. The data description of COBOL is a start to improve this, but why should the description be appended to the program rather than to the data itself?

Do you need a program to read a book in the library? At ACM 70, Dr. John Richardson of the US Dept. of Commerce said "Information Conserves Resources Through Better Decisions," but some of the valuable data that we need to make those better decisions is not, in fact, retrievable, exchangeable, or digestible. It cannot be turned into information. Indeed, one of the major findings in the various studies of data banks is that the sum of many small data banks is not a large data bank, at least not yet, contrary to the fears of many. And yet there are good as well as harmful reasons to consolidate data. If, for example, the US Congress had two reliable pieces of information -- 1) how much it was costing to not grow cotton, and 2) how much it was costing to promote the use of cotton -- the very juxtaposition might give rise to some better decisions. The organizing power of the computer depends completely upon legibility and interchangeability of data.

Incompatibility of Data Banks

A classic example is the situation that arose when the EPA (US Environmental Protection Agency) was formed by consolidation of several diverse groups, each with its own information systems. When they tried to consolidate the data as well, surely one of the main reasons for the coalescence, they found out that data could not only not be exchanged between various components, but not even between the several computer systems in the subdivisions of the agencies! And, of course, the air masses travel over many states, each state with its own computers and monitoring systems, and each computer incapable of making decisions that would optimize for the entire country, much less the world -- if that possibility were permitted.

Illegibility of Computer Data without the Program

Examples of the illegibility of computer data without the program are countless. Dr. Fred Whipple, the astronomer, once mentioned that only 1% of his information from satellite and probe vehicles was being processed. I corrected him slightly to say "data," and he reiterated "information." I asked if anyone could process the tapes if the program were destroyed? He admitted that it would be impossible. "Data" it was.

The Las Vegas city police and county sheriff's department recently consolidated to form a "Metropolitan" Force. It will be many years before their computerized data files can also be consolidated to be of efficient use.

Of course this particular manifestation of swallowing of data and not giving it back to anyone else could be largely solved by using labels and data description on data media, so that the data can be self-descriptive. Congressman Brooks of the US has called for a "declaration of independence for data."

Computer Failure

Another way of not being able to get data out is to have the computer system fail. Integration of computer systems into human affairs demands extreme reliability. We all know this, yet there are many times when one is tempted by the power of the computer to entrust to it a function that has some deadlines. I am guilty of this myself. We use a computer for text processing and publication. The problem is that we are forced to share a computer that is used for software experimentation and new system software validation, or for benchmarking in various configurations. While the hardware may be very reliable, newly-developed software is, unfortunately, not -- and we have entrusted our total text to the disk files. When difficulties occur, no manual methods, however desperate and strenuous, can be employed to do a makeshift job. It is the ultimately perfect job or none at all; we are at the mercy of a system that must be fully operational.

The point of my story is that it is a human failing to be optimistic that the computer will be up! So one does not plan for back-up, duplicating files on another system, or batch methods that work even when timesharing is down. Now we cannot even re-process the sewage. We have given the computer valid and useful data and cannot get it back until too late.

Some Actions to be Taken for Human Protection

Society long ago learned to impose minimum restrictions and educational or training requirements upon classes of workers whose operations affected the public safety or welfare. These constraints led to professions, with codes of ethics and a store of recommended practice often embodied in local law, such as building codes. Examinations by peers is a prerequisite to practice -- for doctors, lawyers, engineers, accountants, ad infinitum. Until now such restrictions have not been imposed upon the computer community; one can only suppose that the professions just mentioned did not materialize so abruptly before the social consciousness.

Some public exposure of malfeasances moved the legislature of the State of California to consider, in 1971, the certification of computer programmers as a class. This was given attention by the press and, together with the fact that the legislature was in a quandary, it was sufficient for assistance to be asked of AFIPS (American Federation of Information Processing Societies). AFIPS convened a System Certification Committee in 1972 February.

Certification of Programmers and Handbooks of Good Practice

The committee arrived very quickly at the conclusion that there seemed to be no authoritative way to achieve certification. I proposed that a series of books of good practice should be conceived and constructed through AFIPS. This project is now underway. The first such book of good practice is on confidentiality and security, due to the very strong and justifiable interest in this topic at the moment, and is about to be field-tested. It is

largely in checklist form. As a minor note, the committee has changed its name to "Systems Improvement," to emphasize the fact that it does not feel that any form of certification is feasible yet.

Reliability of Computers

"Reliability for Integration into Human Affairs" was the title of one of the sessions of the 1973 National Computer Conference in the US. The session had a certain distinction. The other sessions were, by design, to reflect a "vertical" or "end use" orientation. Here I deliberately chose, in planning the program, to take a further step, to see what aspects of computer systems design were common to many end uses for the specific reason that they were directly integrated into human affairs.

The panel included representatives from air collision avoidance systems, online patient monitoring, online power plant control, credit systems, ground transportation, and merchandising. Many of these applications are of Type 3; power control against blackout, for example, requires a response faster than a human can achieve. Air traffic control is another; in the 1980's there are expected to be 5000 people always in the air above Los Angeles, in 700 craft! The representative gave two major requirements:

- Predictable reliability should be astronomical.
- There should be "bail-out" capability for whenever the system fails unpredictably.

"Fire-Drills"

This second point created much discussion. Many of the builders of complex computer-controlled systems found that the people that ran such systems were seldom able to practice fixing them. When they did fail, they were not properly capable of coping. It was suggested that holding "fire drills" for such systems was a basic element of good practice.

Searching for other elements of good practice, it appeared that none of the panelists or their design teams knew of any source or reference book to use for reliability aspects of computer usage, even though there was much commonality in their applications. There are some specialists in this field, such as Bob Patrick, but no body of knowledge is available generally. Patrick gives some examples of bad design:

- One computer installation had back-up tapes in a fire-proof vault, and "grandfather" tapes inside a mountain. But there was only one copy of the "run book" that told the operator how to read the tapes, and that was in the machine room, and would be lost in a fire.
- A military installation had high security, and was very protective of the data. To ensure good readability, the tapes periodically had the first 20 metres or so clipped. The problem was that these tape strips were thrown away, under custody of garbage men without clearances, and they had not been erased!

\$300 Million a Year of Computer Crime

Donn Parker, who chaired the above-titled session, is an authority on computer-related crime. His estimate is that this now amounts to \$300 million a year, and will reach \$2000 million in the 1980's!

Dick Mills of the First National City Bank says that the bank has \$8000 million per day in its interchange "pipeline," so that even a small leak drains a lot. It would seem that we are not being overcautious in insisting upon reliability in such "people-sensitive" applications.

Protective Measures Against Misuse: Legal Measures: Safety Standards

There are many examples of laws for involuntary personal protection. Construction workers must wear hard hats; cyclists must wear leather and helmets. These are occupational protections enforced upon the individual presumably because he represents an investment by society.

The US Government has imposed certain requirements upon the manufacture of automobiles, i.e., to be constructed so as to withstand collision of X km/h without sustaining more than \$Y in damage, or the like. The Government has stated that requiring such action is within its right to protect the safety of its citizens. It seems certain that the computer has a direct effect upon not only the safety of our citizens, but also upon other rights. It might thus be reasonable to demand that software and hardware should also be built to certain standards to protect these rights.

We are certainly going to have to build computer systems with facilities for confidentiality and security. Although there is no law on this, there is little doubt that US Government users will be demanding these features.

This area is covered comprehensively in "Legal Aspects of Computerized Information Systems," a US Govt. Report. See the Honeywell Computer Journal, Vol. 7, No. 1.

Voluntary Measures: Ombudsman

Dr. Harold Sackman, Chairman of the AFIPS Committee on Social Implications of Computers, called recently for a "computer user society of America." This was to be a computer citizen's group active in social reliability, for the reason that the computer community really gets to see the problems first, and has the responsibility to expose the problems to those who can treat them. The ACM owes much to the Scandinavian creation of the ombudsman; its ombudsman program has solved many problems of bad computer usage.

Power to Audit

There is a growing class of auditors versed in data processing, but we may have to take drastic measures to aid them. There are many current efforts for better methods for software construction. One hopes that increased simplicity will lead to more direct legibility and auditability of computer programs. Most programs are documented poorly, and I see only one hope of solution -- the program specifications, narrative documentation, and operating instructions must be integral! Using a block-structured language is vital to constructing auditable software. It also enables programmed devices to detect tampering with the running programs.

Handbooks of Design and Practice

Handbooks of design and practice are required to be available before computing can truly be a profession. Many computer societies are in various stages of using codes of practice and certification

of practitioners. One hopes that they will not stop short of general certification but will also adopt application-oriented certification in joint action with the professions of those applications.

We will have to equip our systems with performance measuring and evaluation capabilities. Wastage of resources has been considered an evil in other fields before this.

Recommendations

As custodians of the power source we have many responsibilities. When I planned the ACM 70 Conference, it was as a model for a National Computer Year, which could possibly be followed by an International Computer Year. A possible list of goals for such a Year could be:

- To consciously put computers in service for international goals; to increase public understanding of computers, their role and potential; and to emphasize the computer as servant by more humanized use and applications.
- To develop strategies for the best future use of computer systems (technological, social, educational, political, and legislative).
- To conserve, and maximize utility of, those existing and future intellectual resources known as data and programs, by finding how to utilize them on multiple equipment and in multiple applications.
- To aid government, business, and private decisionmaking by opening up new and more complete data for those decisions, and to facilitate the making of those decisions by reducing the information volume required (as opposed to data volume).
- To plan a closed cycle for redistributing work assignments between people and computers, for re-education prior to change of assignment, so that people can best fulfill their potential.
- To ensure that public safety and welfare are considered adequately when computers are integrated directly into human activity.
- To set up new and broad interdisciplinary paths for exchange of information among hitherto segregated organizations, and to foster their maximum involvement on an international scale.
- To plan the most economical and effective interaction between computing systems and other systems such as communications.

Interaction Between Computers and Society

It is not too soon for a comprehensive examination of the interaction between computers and our society. Two papers from the 1973 National Computer Conference support this view -- "The Social Implications of the Use of Computers Across National Boundaries" and "A New NSF Thrust -- Computer Impact on Society." NSF is the National Science Foundation of the US.

Easier than We Think?

I believe that it won't be so difficult for computers and society to adjust to each other if we really put our minds to making it happen.

In 1970 an Assistant Postmaster General of the US observed that a third of all first class mail is machine-addressed, but only 6% arrives on the post office docks in Zipcode order. He asked why the computerized address files could not be ordered by

Zipcode as well as any other way? So I asked many data processing departments the same question. The answer was that they had not thought about it, and would just as soon do it that way.

It may be as simple as that.

Integrating Computer Text Processing with PHOTOCOMPOSITION

R. W. Bemer and A. R. Shriver
Advanced Systems and Technology Operation
Phoenix, AZ, US

INTRODUCTION

The introduction of computers to the composition process began in 1961, but not much thinking was applied to the system aspects. This led to some failures and marginal returns. Hyphenation and justification, the earliest uses, are actually trivial. So are text entry and control of character generation.

The challenging functions are page layout, pagination, tabulation, indexing, ruling, proofing, and multiple output from a single file by changing the variables.* However, a danger lies in trying to do these functions automatically by the computer; the amount of difficult programming required often leads to excessive costs, disillusion, and project abandonment - with concomitant prejudice against computers. Shatzkin [1] said:

"The key contribution that the computer can offer the book publisher is very simply this: the predictability of the final result! This may sound very anticlimactic, an absurdly small benefit from such a mighty instrument, but I assure you that predictability can change procedures and even the nature of book publishing in very revolutionary ways."

Our experience in publishing the *Honeywell Computer Journal* has borne this out. As usual, close cooperation between human and computer pays off best. The basic ingredient of our system design is the cost of photocomposition relative to hand or linotype setting, being cheaper by a factor of more than 20. This leads to the philosophy that we shall always make many photocomposition runs, and this governs our procedures from the outset.

* For example, this article was photocomposed (by the methods described) in this form and in an alternate form for the 1973 National Computer Conference; the common portions are used with permission of AFIPS.

Using a computer text processing system as the entry and change vehicle for a photocomposition system affects the publishing function in many ways. Costs are reduced, quality and readability are enhanced, esthetics are more controllable, and entry personnel require little training. Proofreading is almost entirely replaced by a computer-generated concordance. Mechanicals for reproduction are completed at the editor's site, not at the printer's, completing one more step in the movement to the automated office.

The Honeywell Computer Journal is published concurrently on hard copy, microfiche, and magnetic tape. The tape can be used to drive other photocomposition systems that differ from our own, just as a computer can translate COBOL programs to the running instructions of a particular computer. Thus our work has shown the way to a common composition language that can describe all formats and identify uniquely the universe of printed symbols.

SCHEMATIC OF THE SYSTEM

The HIS 6000 system is used for text entry, editing, storage, and running concordances. It is not normally used for the "runoff" function (that is, producing formatted copy on the entry terminal). Even though this feature is available, it is tedious, expensive in line cost, and has little value for final copy.

Formatted copy is produced only by photocomposition. When this is desired, a special postprocessor program converts the text stream and embeds macros for the Page 2 System. This produces a magnetic tape which is (now) transported physically to the facilities of Datagraphics, in Phoenix, AZ, and input to a Univac (nee RCA) 70, Model 2, driving a Videocomp 830 from Information International, Inc. The resulting copy is laid up in desired page form, and a cycle of editing and further photocomposition begins.

Final copy is waxed on templates in the traditional manner. Special heads are added (in fonts not available to the computer system, and chosen to symbolize article content, where possible), and it's off to the printers.

Basically, we have adjoined two free-standing systems, and in so doing removed from the middle the expensive and non-graphic-quality output of the first, and the somewhat tedious and inflexible input of the second. Jury-rigged as it is, it is nevertheless superior to any method formerly available to us, and points the way to integrated systems for the future. We can live for now with our 2-hour minimum for turnaround, from input to galleys.

TEXT ENTRY

Until recently, when we obtained a Teletype Model 38 with tape cassette, text entry had been accomplished in the time-sharing mode with the standard HIS 6000 Text Editor System [2], an embedded format system based upon M.I.T. work and similar to the IBM Script. It is *not* a numbered line system like ATS, and eight years of experience has proved *this* wisdom. Searching and alteration are done primarily in the string mode. "Cut and Paste" is limited to operate by the number of lines moved, but they are not themselves numbered.

All control actions are signified by embedded "dot" commands. This input convention states that a CR (Carriage Return) character followed by a full stop character (period, dot) signifies a control statement. This imposes the minor restriction that a line of entered text may not begin with the full stop. Some basic "dot" commands are:

```
.begin      .center    .indent n  .subpara n .TAB
.space (n) .adjust   .undent n  .para      .break
```

These are but a subset of the standard Text Editor and can be learned by an unskilled person in an hour or so. The editing commands will be explained by examples in the running text of this paper.

ENTRY BY UNSKILLED PERSONNEL

The postprocessor program that converts for the Page 2 System is vital for simple text entry. The standard entry methods for the Page 2 System are certainly *not* simple and do require some training and a crib sheet constantly on display to the enterer. Remember that graphic quality output requires a separate font generation for each unique character. It is not sufficient to overprint an umlaut (in its fixed position) for both the upper and lower case "u", for example. The postprocessor does extensive string analysis, much of which is based upon backspace and overstrike for entry, thus making it simple for unskilled personnel. Examples:

- Characters with diacritical marks (accent acute, accent grave, tilde, umlaut, etc.) are produced by backspace on the terminal and overstrike with the proper character (double quote is used for umlaut).
- Double and single quotes are used as they are for entry. The postprocessor determines whether they are opening or closing quotes. A double quote is two single quotes in photocomposition, and this is called automatically.
- For minor occurrences in text, boldface may be indicated by overstriking single characters three times. This is visible on the terminal when the line is verified. For a longer string of bold characters, the font is altered by a **.bold** command and turned off by a **.bold end** command. These commands do not force a new line.
- For minor occurrences in text, italics may be indicated by backspacing the length of the word and underlining. This is visible on the terminal when the line is verified. For a longer string of italics, the font is altered by a **.ital** command and turned off by a **.ital end** command. These commands do not force a new line.
- The bulleting seen here is accomplished by a **.indent 3** followed by a **.undent 3** (which is operative only for the next line), a lower case "oh", 2 blanks, and then the text. The uniqueness of this string permits the convention.
- To the regular Text Editor convention of using the "at" symbol to delete the previous character (guess why our articles never contain this character!), and CAN to delete the entire line of entry, we have added the caret to indicate the en space, which is incompressible to the justification process. Thus a new paragraph is caused by a **.break** followed by the initial line with two carets for indentation.
- The normal font sizes for *Honeywell Computer Journal* are:

9 point - text
8 point - references, some displays as necessary
7 point - sub- and superscripts, figure captions

Point size may be changed at any point in the text by inserting the ESCape sequence:

ESC g (7-pt), ESC h (8-pt), ESC i (9-pt)

These override the original settings, and are used for formulas, etc.

CONTROL OF PAGE LAYOUT

It has been a remarkable discovery to us that reader attraction and satisfaction is increased significantly by tight control of page layout. Only in the most exceptional cases will a column start in the middle of a sentence, and then only on the second column of the same page. Usually a column will start with at least a paragraph (not just an arbitrary paragraph, but one that makes sense), and often with a heading. The appearance of a figure or table will never precede its first mention in text, nor will it be on a page that is not visible when that mention is made. "Widows" never occur.

Under traditional methods, the editor loses control of page layout after the galley stage; all of the niceties must be left to a composer who has little understanding of the subject matter, and is often less interested in reader satisfaction. With the low cost of text processing taken in conjunction with photocomposition, we do not mind expending many runs to get just what we want.

A quick reading of the first galley copy gives an estimate of the author's redundancy or flowery speech factor and other ways that compression can be achieved if necessary. Accordingly, the actual film is cut to lay out an approximation of the article. As the last page is always full, we work backward. Whatever is left for the first page we leave for artistic treatment and the "From the Editor" commentary. Great attention is paid to aspects of future readability, left or right page assignment, pleasing placement of tables, figures, and photos. Virtually no attention is paid to typos and other mistakes that exist in the copy. Accordingly, the single columns are taped on with more lines than our standard, trusting to judicious editing to cut back to the right number (60).

The beauty of this system is that many things can be changed simultaneously to create correctness, harmony, and interest: point size for certain paragraphs or tables, tab settings, subparagraphing, font style, and text changes and corrections. Imagine a situation where column copy must be reduced by two lines, and yet previous editing has taken advantage of all short lines at paragraph end, removed filler words, and replaced big words by commoner smaller words with equivalent or clearer meaning. Now you have to get into the guts of the author's meaning and say it shorter and clearer, without altering the flavor or meaning in any way! Being forced to do this by our aesthetic standards for page layout yields a big dividend in increased readability.

Depending upon the content, we may photocompose the text from 2 to 5 times. Do the authors complain about the alterations? Never, in our experience. When it reads well, they just assume that they wrote it that way, never checking their original copy. We have also experimented in putting the author's work in to typeset even when it is only rough draft; results seem to indicate that the visualization of final copy permits him to improve it more than he could by editing from a typed draft.

Obviously, doing this much work for readability means that we insist upon high acceptance standards. Dung coated with 53 layers of Chinese lacquer is still dung, and we do not intend contributing to information pollution.

Hopefully, it is now clear why we do not use the computer for automatic pagination.

PROOFING OF COPY

An optional feature, or byproduct, is the concordance run, usually exercised on what is expected to be the next-to-last photocomposition run. This produces two listings on the high-speed (upper case only) printer. The first listing is a Key Word Out of Context (KWOC) listing; each numeral and word (except for the very small common ones) is listed (first appearance only) on the left in collating sequence order, with its entire entry line on the right. The lines are numbered here, for cross-reference to the second listing, which is the consecutive text.

The concordance is now scanned visually, primarily to detect input errors ("typos"). See Figure 1 for some examples. It is our experience that these fairly jump out at one in scanning a concordance, whereas they remain stubbornly glossed over by the eye and mind in traditional proofreading. However, we do read the text - for style and making sense, not for typos. In fact, knowing that you are freed from the typo-hunting task creates a different frame of mind for doing *real* editorial work.

AUTOMOBILE	DISPOSITION	MASSACHUSETTS
AUTOMOBILE	DISHRUPTION	MASSACHUSETTS
AUTOMOBILIES	DISRUPT	MASSACHUSETTS
AVAILABILITY	DISRUPTIONS	MASSSES
CERTAINLY	INSTEAD	SULVED
CERTAINLY	INSTEAD	SULVED
CERTIFICATION	INSTITUTE	SULVEDTU
CERTIFICATE	INSTITUTE	SULVED,
CERTIFICATION	INSTITUTE	SOLVE,
CHARGES	KURT	STEREOSCOPIC
CHARGES	KY,	STEREOTYPED
CHARIMAN	LU	STEREOTYPED
CHARITY	LA	STEROGRAPHICC
CHARLATANS	LAR	STEROTYPE
		STEWARDS
COLUMBIA	LUCATIONS	SUCCESS
COLUMBIA	LUCATION,	SUCCESS
COMPANY	LUCATION,	SUCCESSING
COMBINATION	LUCATIONS	SUCCESSPP
COMBINATION	LUDGE	SUCH

Figure 1. Typos exposed by concordance.

CONTROL OF READABILITY AND STYLE

The concordance produces a histogram of word size distribution as a byproduct, and the average word length is calculated. We target 5.0 characters per word, and are very suspicious of readability when this gets above 5.5.

One aspect of style, or rather one of our rules, is that an acronym shall always be given the spelled-out version in parentheses the *first* time it is encountered in text. One has only to spot the first occurrence in the concordance and look to the corresponding line on the right to see if this has been done. If not, edit.

The *Honeywell Computer Journal* has other style rules. Most important is adherence to ISO Standard 1000, or the International System of Units (SI). Check the concordance for inches, feet, yards, miles, pounds, etc. If they occur, and are for measurement, they had better be in parentheses following a metric value. Other examples: \$2 million - not 2 million dollars; 0.5 s - not .5 sec; focused - not focussed.

ECONOMIC CONSIDERATIONS IN WORKING

As one cannot predict pagination of printed copy when entering text, one could enter it all as a single file. However, the 6000 Text Editor keeps the entire file in the main store for faster processing (it is *really* fast), and these facilities must be paid for. Thus input is made in judiciously separated and named files, e.g., breaking at headed sections. These are then adjoined for the photocomposition run.

After page layout is determined, they are adjoined again and resplit by page into files with new names, and the old ones purged. This permits single columns to be reworked into final form. The present rate is \$1.75 per column. Thus a page costs from \$6 to \$10 to compose, comparing rather favorably with the \$70 per page we were paying for linotype setting to our standards before our system was operable. The 6000 cost is not included, as we have been unable to get real figures because we work on an inhouse "exposure" system used for checking out new software releases. We do, however, feel that this cost is compensated by the system doing automatically what we would have to do ourselves otherwise (like proofreading), and by the added quality. We do need to modify our programs in order to be able to set double column on the last run.

Economy dictates that we should process as much text as possible on each photocomposition run. This means linking several files and saving them as a single file. But this increases the risk that something going wrong early will spoil the balance. Care must be taken to separate and insulate each file from any other. Convention starts each file with `.begin` (for a new galley), `.indent 0` (in case the file ahead of it lacked a command to restore indentation to 0), and `.adju` (in case the preceding file had been using tabulation and was not restored to the justification mode).

The power of the Text Editor is of great assistance in checking for correctness of the adjoined file, particularly for closure. Type:

```
fs:/bold/* (for "find all occurrences of that string")
```

and you will almost instantly get a message like:

```
end of file - request executed 122 times
```

Hit "b" and CR (for backup to the file beginning), and type:

```
fs:/bold end/*
```

If the message doesn't say 61 times - trouble! A 60 would mean that bold did not get turned off somewhere, and the copy following will be in useless boldface. Do the same for italics, subparagraphs, point size changes, etc.

Files must always be correct for the magnetic tape edition, and identical to the printed copy. Yet it is wasteful to rerun the entire file for simple patches. A copy is made, and the correct parts wiped out by string replacement, leaving only the changed copy to be reset as a patch (with due consideration to leaving enough text so that paragraphing, etc., is unchanged). Patches are saved under a different name; several are adjoined and run at one time.

INCIDENTAL ADVANTAGES

A number of dividends have shown up that we amateurs did not really foresee:

- Doing our own typesetting permits laying up mechanicals for articles as soon as they are ready, without waiting to group an entire issue for the typesetter to schedule in some time slot. Exclusive of conditions of extreme timeliness, this permits better selection for issue makeup and content.
- Having the feel of the final product, by mockup during the editing and changing stages, affects everyone - author, editor, and reviewer. For the latter, particularly, it gives psychological impetus to hurry up - lest what he dislikes might be in the finished product. All can work simultaneously to correct and improve the copy and make it more readable.
- The Page 2 System hyphenates to English rules and/or custom. Normally we run our French, German, Italian, and Spanish sections in "fill mode" (stretching the spacing between words to fill the line without hyphenation). But if glaring gaps exist we remove them easily by doing a dummy hyphenation, splitting the first word of the next line into two components:

```
rs:/whippersnapper/  
ENTER  
*whipper- snapper  
*  
READY
```

This method can be used for English text when Page 2 fails to hyphenate opportunely or (rarely) incorrectly.

On one occasion the entire article was side-by-side in both German and English. Here we could proceed more elaborately, removing Page 2 hyphenation that was incorrect for German, forcing correct hyphenation paragraph by paragraph.

- Page 2 also has the flaw of assuming that a change in font style permits a break for a new line just as hyphenation or a space does:

```
..... Protection A  
gency ...
```

Text Editor can force a correction by replacing sufficient spaces between words by incompressible en spaces.

- We don't have to worry about losing corrected galleys in the mails, as the *Journal of the Association for Computing Machinery* did in 1971 October. We also know that the corrections have actually been made in the printer's copy, without waiting for a blue to be returned and show that they were *not* made. This often shortens the production cycle, and certainly cuts costs.
- Secretaries can make very creditable copy inhouse by cutting and pasting galley segments with Scotch Tape, and then using a reproduction method such as Multilith. Interoffice memos are becoming artistic, easier and pleasanter to read, and certainly use less paper.

A WISH LIST

End users should tell suppliers the nature of their applications and what they would like to have to do these applications better, cheaper, and faster. We would like:

- A larger portion of terminals to be equipped with cassettes. Entering text in the timesharing mode is not efficient in line cost.
- Cassettes attachable to office typewriters. If this means new office typewriters, then let them have standard keyboards! By this is meant that not only the placement of the printing symbols, but also the placement of the controls, either as separate keys, or in the control position on the regular keys. For example, Control-X is the usual position for CANcel (deletes the line just typed). Some keyboard designers have not realized that this makes Control-Z a poor place for EOT, because a slip of one position turns off transmission, with resultant loss of all one's work to that point!

With an increased portion of input being generated offline, it would appear that the introduction of the computer at the proper point in the copy production cycle permits entry by *less skilled* people, possibly to the point where the original creator of the text and the enterer are one and the same. Imagine an author out in the woods typing his rough copy and getting a cassette record; he would mark up the pages and send both pages and cassette to an editing service, which would enter the cassette contents and make online corrections to the author's copy according to his indications.

- Alternatively we would take a CRT display if it corrects certain faults of existing systems in line runaround, etc.
- And perhaps a pointer system that could indicate both the beginning and end of a string to be identified for a working purpose.
- A registry of available digitized symbols, so that one would know where to buy their representations in a transferable form.
- More than any hardware, we would like to see the development of a common composition language, and its elements, that is, universally-agreed encodings for printed symbols - their graphemes, their placement, and their style. Elements of a proposal follow:

FEASIBILITY OF A COMMON COMPOSITION LANGUAGE

Production of graphic copy from encoded data is an important component for present and future information retrieval systems. Dot matrix characters on a CRT screen will just not be satisfactory for some purposes. Production of graphic hard copy from an information bank may in the future be cheaper than ordering an existing printed reproduction to be invoiced, found, packaged, mailed, and delivered.

Because future information retrieval will consider many more symbols than those of the present ISO Code, existing and future graphic devices must be connectable to the retrieval system.

Equipments that produce hard (or film) copy may be viewed in the same way that we view computer central processors utilizing different instruction sets and object code, and as we view various numerically-controlled machines. There are single programming languages that are common to many central processors. In N/C, the APT language is processed to produce the CL Tape, which is also common to many processing machines. In both cases the common language is processed by computer to produce instruction for specific and multiple equipments. In both cases the translation capability to specific equipment is usually the responsibility of the manufacturer of that equipment. That this is not so in the composition industry is due to the lack of a standard composition language and metarepresentation of text (with associated characteristics of alphabet or other symbol class, font, size, style, weight, and 2-dimensional positioning). If this existed, it would be a high-level language for copy production which is translated, by computer, to instructions for the various hard-copy equipments. The industry suffers from this lack.

To be feasible, the basic functions of copy production must be similar, even if not carried out in the same way. This appears to be so; it has been proved for the Honeywell Computer Journal, which can also be printed from entry terminals. Indentation, font change, size change, etc., seem to operate as primitives.

To construct a general text-processing language, of which the composition language is one part, we need to enumerate the functions and then assign standard encodings to them. The provisions to do so exist in the ISO Code and the associated expansion and extension techniques. The most general mechanism is ESCape. Some 2-character ESCape sequences are now virtually standard in the 7-bit code, and will likely be single characters in the 8-bit expanded code. Examples are Half Line Reverse Feed, Cursor Up.

Utilizing code extension procedures, provisions are made to be able to select unambiguously a group of symbols, a font, weight, size, etc. We then use a key device or pressure display panel with single function buttons. The operator would perhaps press "Cyrillic" (to get the GOST Standard encoding), "8" point on "10", "bold". Each key would generate an ESCape sequence in series, inline in the text. He then uses either a special typewriter keyboard, a standard keyboard with a chart of correspondences, or some other device, to enter the Russian text. One can imagine the total set of symbols paged on a microfiche for back projection on a screen.

Computer programs (postprocessors) are created to translate from this standard language into the actual commands and character inputs for the copy device, which could be 6-level Teletypesetter, Monotype, Photon, RCA Page One and Videocomp, Datel typewriter terminals, IBM Selectric Composer, etc.

Until new entry equipment is made available to conform, similar preprocessors could be written to convert from the various entry conventions to the metarepresentation. This would reduce the translations from $N!$ to $2N$. If all entry equipment would eventually conform, then a further reduction to N occurs, where: N = the number of different composition equipments.

It is expected that this would free the photocomposition industry for expansion in the same way that FORTRAN, COBOL, and ALGOL did so for computational usage. It would provide international standards for alphabet representation.

CLASSIFICATION AND GROUPING OF SYMBOLS INTO PAGES

ISO TC46 (International Standards Organization Technical Committee 46), Documentation, has a Subcommittee 4 on Automation in Documentation. This body has responsibility for collecting and/or developing the pages of encoded symbols. Examples of such pages are:

■ Characters to form natural languages (alphabets)

ISO [DIS 646]	Kata Kana [JISCII]
National/accented	Kanji
Cyrillic [GOST 13052-67]	Phonetic
Greek	Dactylogy [hand signs]
Hebrew
Arabic	Other punctuation [character
Sanskrit	augments, bullets, rules,
Braille	bars, leaders, etc.]

■ Symbols of various fields

Aeronautics	Medicine
Astronomy [Astrology]	Meteorology
Biology, Botany	Money
Business [Commerce]	Music
Chemistry	Philately
Ecclesiastic, Fraternal	Pictorial, Ornaments
Electricity, Magnetism	Transportation
Flowcharts	Typography
Games	Welding
Heraldry [flags, insignia, arms]
Logic diagrams	Other Scientific
Mathematics, Geometry, Physics	

- Controls for changing point size, weight, slope, font, position relative to the base line, horizontal compression, etc.

An ESCape sequence and prefix character should be proposed for each page of symbols, for registry with ISO TC 97, Computers and Information Processing, which body maintains this registration authority for extension and expansion of the ISO Code.

REFERENCES

1. R. W. Bemer, Ed., *Computers and Crisis*, Association for Computing Machinery, New York: (1971) 401 pp.
2. *TEXT EDITOR Quick Reference Manual, Series 600/6000*, Honeywell Information Systems, Inc., Waltham, MA, DB42, 1972 June.

FROM THE EDITOR

Caution: Premature implementation of the substance of the following article may not be a wise business move, but it does give examples of sets of additional characters necessary to construct the common composition language we have just discussed.

We are particularly pleased to be able to give broad distribution to such forward-thinking work. As an old standards man, I have always been discouraged by the slow processes and concurrent lack of dissemination of knowledge about what might eventually become standards. Meanwhile the hundreds of hardware designers, unaware of the standardization work and trends in progress, choose their own arbitrary methods of operation. This time there will be no excuses. All designers are alerted that serious work is in progress. Although it nears completion, it is still possible to join the effort. When the standard is published it will be difficult to change and may conflict with existing equipment.

My solution is to publish sufficient extracts from work in progress so that the designers can be alerted and consult with the standards committees whenever they have questions. Show this article to two or three hardware or software designers of your acquaintance, and witness their shock!



An Analysis of the Spelling Arguments

ABOUT SPELLING OF (SI) UNITS

R.W. Bemer, Editor - Honeywell Computer Journal
Honeywell Information Systems, Phoenix, AZ

As editor of a publication of a multinational company (in ISO A4 size, and utilizing SI units exclusively for more than two years) I take a very firm international position in the current furor over the spelling of certain SI units. To document how I have arrived at this position, the arguments for and against the spelling of metre and litre (in particular) are listed and analyzed here. Some rebuttal is expected, but past indications are that it will be largely emotional. As an American citizen I am fairly dispassionate about my prerogatives to tell the rest of the world what it should do, as compared to my strong concern about what a negative balance of payments does to the well-being of my family. I also do not think that this is a "zero-sum" game, where the other countries must be less prosperous according to the United States being more prosperous. Our firm adherence to the SI, as it is internationally understood and agreed, will help all.

1. FOR "er" (as in meter and liter)

- *It's phonetic*

True. "meter" is pronounced exactly the same as "meeter". But is this a valid basis for choosing a spelling? Particularly for common words? Many people pronounce "their" and "there" identically, and these are the 53rd and 67th most common words, respectively. Yet we seem to find the difference in spelling to be a useful distinction that many people seem to be able to make.

The phoneticists, however, don't seem to argue for "leter"!

- *It's American*

Perhaps. But if one takes this attitude, he must be prepared to swallow the hard fact that, of the derived units, only two are named after Americans (tesla and henry). The others will probably be Un-American, in his view.

- *It's German*

Yes, the Germans apparently use the "er" form. They have a need to do this because of the fairly phonetic structure of the language. English cannot depend upon phonetics.

- *It's Webster's Third International Dictionary*

Yes, in the Third Edition, "re" has been downgraded, but *only* for "meter". Note that "re" still has equal status for "litre", "mitre", "nitre", etc.

The Merriam staff states that it only reflects practice; it does not define it logically. Occurrences are counted in such publications as *Science*, but not General Motors manuals.

Note also that Webster's Third contains many usages in direct contradiction to the SI, and this is somewhat understandable inasmuch as they reflect usage in the old and less coherent metric system (See box for some examples). One can safely predict that the "authority" of the dictionaries will make difficulties for the US conversion to SI units, unless the educational and technical materials specifically mention this problem and say "We're right".

Moreover, appearance in Webster's Third does not necessarily indicate American usage. See for example the definition of "date", wherein the order is given as day, month, and year.

continued on page 19

Spelling Argument (Cont.)

- *Some US societies have chosen "er"*

See Mr. Sokol's summary.* It is interesting to note that most of these societies deal basically with "soft" products for export; i.e., the dollar volume of sales of literature abroad is likely to be somewhat lower than the sales of chemicals, wheat, computers, automobiles, etc.

2. AGAINST "re"

- *It's foreign.*

Our whole language is foreign; we got it from the English. However, if one says that English is *not* foreign, then neither is the "re". Webster's Third says, for example, that "theatre" was a Middle English word. It says the same about "mitre", "nitre", and "centre".

I see "4.2 litre" on the Grem-lin, and our local restaurants have menus listing wine by the "litre". Why haven't the common people rejected this foreignness?

- *The US has been using "er"*

True. But we have been using the CGS system in this context, not SI. We used it that way when there *was* no international standard. Now there is.

3. AGAINST "er"

- *It's not the International Standard*

Some argue that there is no definite place where international standards say that it should be "re" rather than "er". However -

- *"re" is in conformity with the definitions given by the authoritative body on this question - the General Conference on Weights and Measures.*
- *ISO 31 recommends the "re" spelling for the English language.*
- *All ISO and IEC standards and papers use "re" for documents in English.*
- *All British Standards use "re", which is certainly the common spelling in the UK (this seems to make "metre" and "litre" English words).*

- *The Canadian Metric Association has chosen "re".*

- *It gives a dual system*

Imagine the confusion in the United States if the proposal of the National Bureau of Standards were to be followed - that the scientists use "re" and the lay people use "er"! It is commonly accepted that the strongest force behind US conversion to SI is the need to redress the balance of trade payments by more export sales (and the President's Export Council has just warned that the balance will get much worse because we are paying so much more in oil import prices). In addition to value available, sales also depend upon goodwill. Conformity to the usage of prospective customers will increase goodwill and understanding.

It is also quite possible that dual usage could lead to ambiguity and unenforceability in the laws.

The dual system would lead to increased production costs. GM would have to print two sets of manuals. And can you imagine the last man on the American Motors assembly line having to select the "4.2 litre" name hardware or "4.2 liter", depending upon whether the car was destined for foreign or domestic sale?

4. FOR "re"

- *Some US societies have chosen it*
- See Mr. Sokol's summary. Note that these are the societies related

to "hard" products.

- *Many major US companies have chosen it*

Again see Mr. Sokol's summary. Do you think that GM, Ford, IBM, Honeywell, etc. would have made this decision if it weren't in the best interests of international trade, or if it would have been unpalatable to their employees?

- *It's British and Canadian, and therefore just possibly "English" Self-explanatory.*

- *It gives a valuable distinction between "unit" and "instrument"*

The distinction between "metre" and "meter" should not be lost, for it will lead to *less* confusion in learning the new system. The first is a unit of measure, the second is a measuring device. This distinction can carry through for verbal understanding as well. One would teach it in this way:

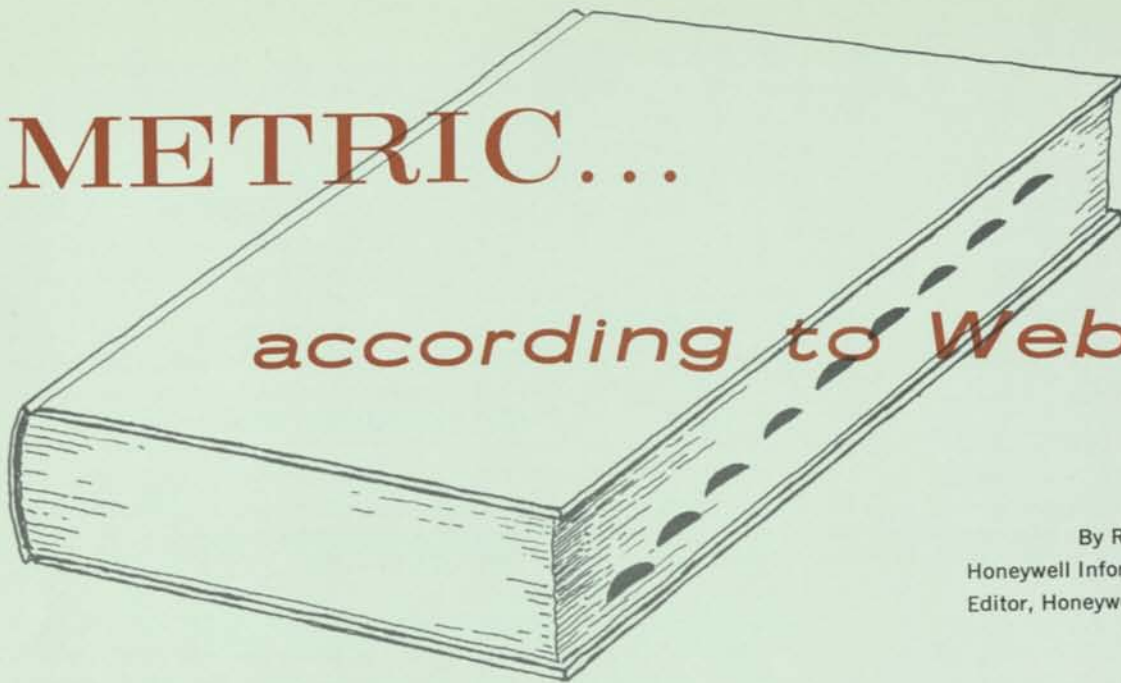
Meter - this is an instrument to measure with. You can touch it. You can pronounce compound words (with meter in them) correctly by noticing the "er" ending. A thermom'eter measures temperature, a manom'eter measures pressure, a speedom'eter measures velocity, etc.

Metre - this is a unit of measurement. You cannot touch it. You can pronounce compound words, (with metre in them) correctly by noticing the "re" ending. A kil'ometre is the unit

* See page 59

METRIC...

according to Webster



By R.W. Bemer

Honeywell Information Systems, Inc.
Editor, Honeywell Computer Journal

SOME ASPECTS OF WEBSTER'S THIRD NEW INTERNATIONAL DICTIONARY (1971) THAT QUESTION ITS AUTHORITY AND RELIABILITY IN MATTERS PERTAINING TO THE (SI) METRIC SYSTEM.

1. p. 1181 - no entry for "international system (of units)"
- p. 1181 - "international unit" defined solely as "a quantity of a biological . . ."
- p. 2322 - no entry for "system international"
- p. 2110 - "SI" is the abbreviation for either "short interest" or "staff inspector"

Conclusion: Not authoritative for ISO R 1000, dated 1969 February.

2. With respect to adequate definitions for SI units:
 - p. 1236 - *kelvin* is defined only as "relating to", not as a unit per se.
 - p. 325 - *candela* is defined only as "candle, 4b", although the term was adopted by 1948.
 - p. 1221 - *joule* is defined in terms of *ergs*, which are defined in terms of *dynes*. Neither of these are deprecated, per the

agreement of 1960.

- p. 2361 - *tesla* does not have any definition at all (although in force since at least 1960).
- p. 1345 - *lumen* is defined via "candle" and "unit solid angle", not via "candela" and "steradian".
- p. 1424 - *metre* is defined as "chiefly Brit var of METER", and on
- p. 1422 - *metre* is no longer given as alternate spelling to meter (the unit), as it was on p. 1547 of the Second Edition, which carefully used only *meter* for the instrument. Yet,
- p. 1321 - *liter* and *litre* are both still equivalent, as are p. 1448 - *miter* and *mitre*, and p. 1530 - *niter* and *nitre*.

Conclusion: It is clear that this edition is oriented basically to the old (less coherent) metric system, and not to the rigorous

SI. A substantial amount of nationwide confusion is sure to occur in the U.S.

3. With respect to the prefixes:
 - p. 141 - no entry for *atto-* (adopted 1961)
 - p. 837 - no entry for *femto-* (adopted 1961)
 - p. 2357 - no entry for *tera-* (adopted 1960)
 - p. 1 - the prefix *ab-* is given as valid for certain CGS units, e.g., *abampere*, in direct contradiction to the official prefix system of the SI.

Conclusion: Attempts to understand the prefix system of the SI by studying Webster's Third will be unsuccessful and lead to invalid usage. It is neither correct nor adequate for reference in U.S. conversion and for teaching the metric system in schools, where it will cause confusion by being contradictory to the textbooks.

continued on page 59

Dictionary Metric (Cont.)

4. With respect to the symbols for the units and the prefixes:
- The symbols are not to be found with the definitions.
 - Under the definitions of the symbols themselves, the units and the following prefixes are catered for (although with some confusion as to capitalization): kilo- hecto- deci- centi-
 - The symbols for the following do not define their use for: giga- deca- nano- pico-
 - And unfortunately the symbol "m" is defined to stand for: mega- micro- milli-
 - The symbol "a" is defined to stand for angstrom unit (definition 8, example 29), and neither the symbol nor the unit have the diacritical marks.

Conclusion: Much improvement is possible.

5. To cap the confusion, one is always referred to the metric table on page 1424, where the following errors exist:
- *deca* has "dk" as its symbol — but should be "da".
 - *myriameter* is defined as equal to 10 km, abbreviated *mym*. No such prefix or abbreviation exists in the SI (see p. 1495 for definition of *myria* - "especially in terms belonging to the metric system").
 - *quintal* is defined as equal to 100 kg, abbreviated *q*. No such unit or abbreviation exists in the SI, and even the abbreviation is incorrectly formed, as it does not contain *g* for gram. (See also an additional definition for *quintal* as between 100 and 130 pounds.)
 - *gram* is abbreviated as either *g* or *gm*, the latter being illegal in the SI.
 - *metric ton* has an alternate abbreviation of "MT" (can a megatesla really equal a tonne?).

Editors Note: Do not confuse metric symbols with abbreviations. Symbols do not use periods except to end a sentence.

Spelling (Cont.)

The *metre* spelling is well established by the majority of US industry now using the International System of Units (SI). There is much logic to support this spelling and virtually none to substantiate a departure. ISO-1000-73 uses the "re," ISO Recommendation (ruling) #31 and the General Conference of Weights and Measures conform to the "re." IEC documents all use the same spelling, the British Standards Institute, Canadian Standards Association, and those standards organizations of Australia, South Africa (English) New Zealand, India, Zambia and other English speaking nations use the "re" spelling. These national associations follow the same: American Society of Mechanical Engineers (ASME), American Society for Testing Materials (ASTM), Society of Automotive Engineers (SAE), and the Society of Manufacturing Engineers (SME) - - - there are a number of others.

A translation of "Le Systeme International d' Unites," approved by the International Bureau of Weights and Measures, published by the National Bureau of Standards and the National Physical Laboratory (U.K.), called NBS 330, contains the following statement;

"The spellings "metre" and "kilogram" are used in this USA /UK translation in the hope of securing worldwide uniformity in the English spelling of the names of the units of the International System."

A *meter* is a measuring device, *metre* is a unit of measure. A micrometre is measured with a micrometer. The spellings denote the difference. The same is true of the two words "their" and "there." Their meanings are very different and can only be determined by the spelling. No emotional uprising ever attempted to eliminate "there" because of the "e."

Those groups departing from the logical spelling of *metre* are going to find they represent a small minority. Our only interest in the settlement of this issue is to help eliminate confusion and avoid the re-education of students. Uniformity is very essential when an industry, a community or an entire nation of people switch from a customary to a new system of measure.

The following is a letter received by the Journal from Mr. Louis Sokol, President, Metric Association.

Following the issuance of NBS SP330, I as editor of the *Metric Association Newsletter* published the August 1971 issue using the "re" spelling in all articles in the belief that the U.S. had officially adopted that form. This elicited a strong objection from Dr. John N. Howard, editor of *Applied Optics* and at that time a vice president of the Metric Association. Two other readers also objected to the new form of spelling. Dr. Howard's argument for the continued use of the "er" spelling was that it was the "American English" form and considerably more phonetic than "British English." In deference to his objection, I reverted back to the "er" spelling in subsequent issues of the *Newsletter* but continued to use the "re" form in my personal articles and slides.

The international standardization of the English spelling of *metre* is highly desirable. Britain, Australia, India, South Africa (English Language), New Zealand, Canada, and the English speaking developing nations will not change to the "er" form from what they consider to be the international spelling form. Thus if we are to achieve a universal English language spelling of the two units, it will have to be *metre* and *litre*.

Louis F. Sokol

METRIC PAPER SIZES...

There Are None!

R.W. Bemer, Editor - Honeywell Computer Journal

Much has been discussed on the subject in recent weeks. Moore Business Forms has published a small booklet which opposes ISO paper sizes. Several schools and industries have expressed views on the topic and want questions answered. This article may answer a few of the objections and questions.

There have been many public statements in opposition to converting United States usage to the ISO A- and B- series of paper sizes. These are quite often called metric sizes, a most erroneous appellation. There is no such thing as a "metric" sizing system for paper sheets. Those making the argument that we should change our paper sizes as an integral part of a metric conversion plan are in error. SI and paper sizes are only under concurrent study.

There are, however, adequate reasons for change. A table in Webster's Dictionary illustrates the incredible hodge podge of our present sizing system (153 sizes/tradenames are listed). I am one that argues for the U.S. to change as soon as possible to the international standard paper sizes, of which the most representative sheet is the A4 size of 8.27" x 11.69". Never mind that this sheet is basically defined as 210 x 297 mm. The point is that this ratio of $1:\sqrt{2}$ would remain so even if the U.S. never did adopt the SI as ordinary usage. And this ratio is the key, as each

smaller size can be derived with only a cut, and no trim.

Objections to using the international sizes appear to derive from conversion economics in at least three major areas — printing presses, computer printers, and storage — and some comments on these follow:

1. Printing Presses

The customary argument is that it will cost more to make runs with "A" sizes, due to available paper stock or width limitations. For the former, don't worry; if the demand is there, rolls will be supplied at proper widths. For the latter:

- a. Beds of sufficient widths can be found on many equipments, either because they are presses of foreign design or of U.S. design with forethought.
- b. If the bed is of insufficient width (even allowing for better utilization of minimum trim), signatures need to be laid up at 90° rotation. Here they will run 6% slower; on the other hand, they will run 3% faster in bed of sufficient width.

The Honeywell Computer Journal, of which I am Editor, has been in A4 size for almost three years. It was apparently the first US publication to appear in this size; as far as I know, yours is the second. and my congratulations are offered. Our trouble and added expense to do so is zero. Anyone who does not know how to do it is referred to our printer, Richard Wilen, Seal Send Systems, 1383 Pine Court, East Meadow, NY 11554.

2. Computer Printers

Computer output printers usually produce from 300 to 1600 lines per minute, often in multiple copies, which is a volume worthy of economic consideration in this question. A substantial portion of this paper is supplied by Moore Business Forms, whose retired Vice President Richard J. O'Brien has produced an extensive study in a booklet entitled "Metric Business Letter Sizes for North America". It is a good compendium of this topic, despite the fact that he himself disclaims the "metric" part, it deals with more than letter sizes,

and I cannot support the logic of some of his conclusions.

Most current computer printers feed paper by the tractor method, not pinch drive. Most tractors, worldwide, are Kidder tractors, produced by a subsidiary of Moore Business Forms. The fingers on the drive chain, and consequently the perforations on both sides of the forms, are at $\frac{1}{2}$ " spacing, and the driving is by stepping motors that hiccup to produce 6 (and rarely 8) to the inch spacing. As a consequence, even the ECMA (European Computer Manufacturers Association) document on "Recommended Sizes for Continuous Sprocket-Punched Stationery-Part I" conforms to this inch basis. Thus if continuation to a new page is achieved by continuous line printing, rather than a slew to the first line, the fanfold must be located in fractional-inch multiples.

However, there are some arguments that this limitation will either not always obtain or may be overcome. Some are:

- a. The newer and higher speed computers are nonimpact type, feed by pinch drive, use continuous (not fanfold) paper, and cut at arbitrary lengths. My employer Honeywell Information Systems produces a printer of this class, capable of 18 000 lines per minute.
- b. Paper in shorter supply at higher cost and mailing rates has spurred conversion to COM (Computer Output Microfilm). No dimensional problem here, although in SI units.
- c. The ECMA document describes fanfold paper at $11-2\frac{2}{3}$ " length, which maintains the 6 per inch spacing, while producing a paper length of 296.34 mm. One assumes the tolerance of 0.67 mm would not be noticed or present any binding problems.
- d. Conversely, the $\frac{1}{2}$ " hole spacing can be maintained for any fanfold dimension. This would

mean that successive sheets, or pairs of sheets, would not have side holes in the same relative position. This would only affect certain of today's binding methods where the perforations are not removed and the fold is at the top of the sheet due to printing a row at a time. Yet this is no more limiting than using a printing method where the fold is on the left side, as is normal for all other types of printing.

3. Storage

By a strange coincidence, my standard file folders are 297 mm long; if they fit in files, then so does A4 size paper. I have no trouble in book cabinets, as US notebooks are about 292 to 295 mm deep.

In a recent interview, Gilbert Jones (of IBM's ultimate management group) said that IBM's Office Products Division was the fastest growing division of all, reflecting the great increase in Word Processing. He also predicted that it may not be too long before there are no office files in paper form.

That should weaken the storage argument against ISO sizes.

mm

TOOL and DIE ASSOCIATION CONTINUES TO SUPPORT METRIC

Tool and die members assert that the recent defeat of the metric bill in Congress won't slow the pace of conversion, and some contend it doesn't matter what Congress does at all on the issue.

An informal poll of attendees at the National Tool, Die and Precision Machining Association meeting last week, showed the group disappointed at the defeat but not unchanged in their opinions.

Rudolph Bredenbeck, president of Efficient Industries, Inc., Cleveland, commented he is finding his customers already changing over, and "wouldn't be surprised to find the country on the metric system within 10 years even if Congress does nothing.

"The defeat of the bill was disappointing," he continued, "there's no doubt about that. But we feel it was due more to the way it was brought on the floor, barring amendments, than to the provisions of the bill itself. We feel that the lack of backing from organized labor was critical. They obviously wanted some amendments, and had enough backing in Congress to defeat the bill as it stood.

"But it probably will be brought up again, possibly this session, and we feel it will pass."

DEPARTMENT OF DEFENSE IN ALABAMA

Roy Trowbridge, President of American National Standards Institute will be the dinner speaker at a DOD meeting July 10 to 12th in Huntsville, Ala. The 3 day conference is the first major metric planning and orientation program the department has held since metric conversion was announced in a Washington D.C. meeting in April 1974.

CHRYSLER CORP. TO ANNOUNCE METRIC PLANS SOON

Informed sources in Washington recently reported to the *Journal* that Chrysler will soon make a public statement regarding the metric conversion now underway.

The book, "The International (SI) Metric System and How It Works" by R. A. Hopkins is the most valuable reference text to hit the educational system in Ontario, Canada.

C. Richardson,
Technical Coordinator,
London Board of Education

Standards in Performance Evaluation and Measurement

R. W. Bemer

Honeywell Information Systems, Phoenix, Arizona 85005

Giving "evaluation" equal billing with "measurement" opens the door to discussion of performance that is good or bad, as opposed to fast or slow. Through this opening come considerations of security and confidentiality, validation of software and hardware means for performing arithmetic operations and evaluating mathematical functions (to varying degrees of precision and accuracy), code independency, auditing and warranty, optional optimization in compilation of running programs in high-level languages, and retention of statistics of every aspect of operation—for later analysis and reduction of duplicate work.

Key words: Accuracy; audit; certification; code-independent; documentation; optimization; precision; run statistics; security; terminology; validation; warranty.

1. Justification

The United States Government has imposed certain requirements upon the manufacture of automobiles, i.e., to be constructed so as to withstand collision at X kph without sustaining more than \$Y in damage, or the like. The Government has stated that requiring such action is within its right to protect the safety of its citizens.

Perhaps the reason that analogy of automobiles to computers is so facile is that computers are also a major restructurer of society. The newer computer uses have a greater than ever proportion of integration into human activities (even into the automobile). It seems certain that the computer has a direct effect upon not only the safety of our citizens, but also upon other rights. It might thus be reasonable to demand that software and hardware should also be built to certain standards to protect these rights.

Giving "evaluation" equal billing with "measurement" in the discussion of performance of computer systems is a major step, for it permits us to subsume good and bad performance as well as fast and slow performance. It enables us to view the need for confidentiality and security concurrently with performance measurement. There is probably much commonality in the requirements for both.

2. Nomenclature

The present intense efforts on performance evaluation and measurement indicate a movement toward professionalism in the computing field. Yet inspection of successful professions shows the basic need for standard nomenclature, and this is lacking in our field. In particular, the American National Standard Vocabulary is to be renamed as a dictionary; this is quite proper, for it is only a list of defined usage in alphabetical order of the terms. It has no structure, whereas the IFIP/ICC Vocabulary did. Imagine a dictionary for the botanist!

And did you ever see such a sloppy term as "overhead"?

We might start with the primitive of:

- Work—Answer-producing
 - Answer-validating
- Not Work—Scheduling
 - Monitoring
 - Allocating Resources
 - Reporting
 - etc.

Another partitioning includes people as well—in a time sequence of software preparation, testing and validation, production runs, and modification. All of these need to have subactivities named and defined more rigorously than at present. The jargon of JCL is incomprehensible to those that use other systems, and vice versa in many cases.

We need standard terminology for the operating system functions—resource management, data management, core compaction, incomplete allocation attempts, waiting, swapping, saving for restart or protection against crash, user validation, etc., etc., so that the smaller functions and program kernels can be assigned to their proper place in the classification structure.

These are the working functions, which would go on whether or not the performance was measured. Similarly, we need good definitions of the monitoring and measuring functions.

3. Reporting

A distinction should be made between the two types of reporting—online for operator intervention and change, and offline (later) for accounting and analysis. Both provide opportunities for performance improvement. The most improvement is likely to be available through providing the operator with sufficient tools, once the operating system has been shaken down somewhat. (I would prefer to see operators of higher caliber than programmers, at least for complex systems, with this reflected in the promotion scale.)

ANSI X3 is very unlikely to achieve a standard for operating systems. There could be some standardization in the subset of reporting activities and their appearance to operators. This might seem unnecessary in the present situation, where programmers change installations with a basic knowledge of some standard programming language, whereas operators scarcely ever do so. But wait until management finds out that some operators have skills, and a feel for tuning a system, that make them far more valuable than any programmer who knows COBOL only.

Accordingly, it is not too early to seek some standards for reporting, by both printed message and analog displays, of resources allocated and used with respect to the individual jobs or batches of jobs. From the crude manometer display on up, more than resource consumption must be reported; contention must also be reported and identified to specific tasks, i.e., resource wastage as well as resource consumption.

4. Software Construction

4.1. Code Independency

All software, whether it be written in high-level or assembly language, should be code-independent from the native character code of the CPU and/or any other code such as the ISO Code (ASCII) and EBCDIC.

The importance of this condition may be judged by the fact that the original 360 software, written without control over such code dependencies, has never been able to be converted to run the 360 as an ASCII-based machine—a feat that the hardware is fully capable of doing.

It may also be judged by an example program in the benchmark tests for the WWMCCS procurement. The source program, although written in COBOL, utilized conditional statements that were operative based upon knowledge of the collating sequence of the EBCDIC (in order to provide these benchmark programs, they were first written for the IBM 360/50, and so tested). The HIS 6000 programmers assumed from the terms of the specifications that ASCII was to be used throughout, and at first could not get correct answers. When a subroutine was inserted to mimic the EBCDIC sequence, there was an 8 percent penalty in running time.

The class of statements that can operate improperly due to code dependency is definable. Source programs may be searched mechanically (by program) for such occurrences, and offending statements at least printed out for manual inspection, if not automatic.

Alternatively, input data to program testing should be given in up to three codes—ASCII, EBCDIC, and the native CPU code if it differs. Such testing should all fall under the Quality Assurance function.

As to public warranty, all software should be certified to auditors, and in advertising, satisfactorily tested for code independency, whenever there is any possibility of portability.

4.2. Frequency of Usage

Software should be so constructed that a frequency count of execution is obtainable, upon demand, for all components. This requires a standard way of identifying such components, and conformance to standards for call and linkage (in hierarchical form, by function).

There should also be provision for count of actual machine instructions during execution of a working program (for the program itself, however, distinct from the operating system, which should have its own count). This provides a "signature" analysis of generated code. In the WWMCCS procurement, a high frequency of single-character moves indicated improper generation of object code. Rewrite resulted in a great improvement in running time.

Frequency of program component execution is quite a different thing from frequency of instruction usage. Both are useful. The latter may be accomplished satisfactorily in a Monte Carlo sense by trapping the instruction in operation at fixed intervals of time. In 600 FORTRAN, this showed that a 4-instruction linkage took up 7 percent of all running time during compilation. Two instructions were cut easily, thus improving 3.5 percent. Over the lifetime of the system, this amounts to several million dollars.

4.3. Computational Accuracy

Results, or answers, are commonly not as accurate as the programmer expects them to be. This is often due to successive operations, truncation, roundoff, basic precision used for both fixed and floating point operations. Use of greater precision should be not only under the control of the programmer, but also as a handle to the operating system. It is conceivable that the programmer should be required to state a value of expected or required accuracy for answers from a computational program segment. The operating system could randomly switch to multiple precision and rerun that segment, with an error message if the difference from the single precision answers exceeds the stated bound.

There should be a standard for floating point computation (in either hardware, firmware, or software) that says: When addition or subtraction of two floating point numbers results in an effective zero because they are of equal magnitude to the precision used, the result shall have a fixed point part of zero, with an exponent part diminished only by the precision of the fixed point part—the exponent shall not be the minimum representable. For old CPU's that do not operate in this manner, all such computations should be interrupted for logging and/or notice to the operator/programmer.

There are many studies in the literature (and the number is accelerating) that show inaccuracies in the

common mathematical and business functions that exceed by far the inaccuracies in the normal arithmetic functions. This calls for certification of such functions for specific accuracy within a specific range, with public notice given—for either free or product software, arithmetic, mathematical, or business.

There should be a standard for such programmed function that requires the accuracy, execution time, and storage use to be integral with the function. Then the programmer could call for certain accuracies for general computation, and one of multiple forms for a specific function could be selected to meet (but not overmeet) that requirement.

4.4. The Compilation Process

We take the premise that programs of any significance will be compiled many times prior to successful operation, and many times later for update and modification, and that this process will move to the jurisdiction of other than the originating programmer.

Optimization is often a substantial component of running time, sometimes up to half. Therefore compilers should be constructed so that optimization is selectable.

Virtual storage or not, breaking up a large program into several components for compilation and testing is still good practice.

The compiler should have facility to flag identifiers of fewer than enough characters to make good documentation for other users. Uniqueness is not enough.

Compilers should always produce an updated source program! This should contain at least:

- An imprimatur identifying the compiler used, language features required (or not used), level, and time.
- A statement of the facilities and resources used, running time (either demanded or assigned), etc., for later analysis.
- A concordance of identifiers and statement types used (this may be in hard copy at option).
- A reblocked source program, indented to show nested levels.
- Appended list of mistake messages, if any, or an indicator of successful compilation, as far as the compiler can tell.

5. Documentation

All data on media should be self-descriptive as to format and content, regardless of whether or not it is to be used for interchange. Present labeling standards are insufficient.

It is presently difficult to associate program documentation and run instructions with the program itself, because many programs are kept in punch card form. However, with the full-scale advent of cassettes this condition should be mandatory.

Local documentation, i.e., that associated with the individual operating statements or groups of statements, may be subject to a certain minimum amount

3. Workshop Discussion

Many participants felt that "standards" could not be set because performance evaluation ideas have not matured adequately. Instead, the term "guideline" was adopted by most people. One of the areas for potential guidelines was accounting data.

Browne: It should be possible to have some guidelines, even if not standards, saying that all systems shall put out the following things on an accounting basis. If it's done right, there should be some minimum guidelines for main-frame vendors and software vendors that solve some of our problems. I think this is a "must." I think we should put some guidelines down suggesting that this is a minimum kind of thing that we ought to be looking for; we'll do better later.

Bell: It seems that the epitome of what we're stranded for is for accounting data, when the systems collect essentially the same data and put it in different formats with slightly different definitions. It's apparently trivially easy to make them coincident. They ought to be coincident so that things can be done in a consistent manner. It's like having tape drives with different size reels.

Browne: There are two points to the problem. They should be receptacles for linear transformation and be consistent.

of verbiage, else the program may not pass Quality Assurance.

6. Hardware

It is difficult to make many standards for hardware design, for the technology is at a time when virtually anything is possible at a reasonable price, due to microprogramming and chips.

One definite requirement is that all CPU's should have at least two clocks—one continuous and one resettable—both fully available to software.

Bell: I second it.

Boehm: Ok, would somebody state precisely what it is that we're saying ought to be "musts."

Browne: I think we should write some guidelines—we must write some guidelines for minimum content in the accounting system and for a common format for accounting data.

While the need for such guidelines was clear, potential problems were noted by other participants.

Kolence: I'd like to recommend two points that I think are important. One is that along with the type and format of data to be obtained, the capability for the user of such data to obtain other new data is important. In other words, I don't think we could expect our suggestions to serve a fixed set of data that's going to be given for everything. I think it's imperative that we make a resolution open ended to permit other types of data to be collected. In other words, the facilities must be there to collect other data than what we anticipate now. That's point one. Point two is what we were talking about earlier: That integrated instrumentation systems include a minimum set of accounting data and report it well.

Jeffery: You want also to be absolutely sure that what goes into a guideline can use results from a research environment.



Robert W. Bemmer is Senior Consulting Engineer with Honeywell Information Systems. He has published numerous technical articles, is a Fellow of the British Computer Society, and served as Program Chairman for ACM 70. He was editor of *Computers and Crises* and the *Honeywell Computer Journal*.

SOCIAL ASPECTS OF COMPUTERS

"...Noticeable social aspects are acquired by a tool when there is sufficient irresponsible usage of it...computational correctness, program correctness, operational correctness and safety are very important...equally as serious as security and privacy..."

My title may suggest a vast survey; actually, I intend to discuss only the nature of "social aspects" in a generic sense, not what they all are. Noticeable social aspects are acquired by any tool when there is sufficient irresponsible usage of it. Guns are an example. Used for target shooting or hunting when food is needed, there was no particular social aspect or problem. Now restrictive legislation is being proposed everywhere. Fire is another example -- it's not the heating but the arson that causes difficulties. So this talk is restricted in scope to responsible usage of computers.

I shall give some examples of poor computer usage. This does not ignore the numerous examples of good and beneficial usage, but serves to emphasize that the current accent on security should not blind us to other aspects that will soon come into sharp focus. Some types of computer usage are bound to become less permissible or tolerable.

Responsible Computer Usage

Security is only one aspect of responsible computer usage. Donn Parker talks about computer abuse that is intentional. I have for 25 years seen computer abuse that was unintentional. The difference is that formerly nobody would bother you about it. Now unintentional and careless abuse needs to be guarded against because it is no longer free from the displeasure of the public; and, because specialists in protecting the public against such abuse may begin to operate.

Malpractice insurance in the medical field is in a crisis condition right now. The great jumps in both

awards and insurance fees show how protection of the public can at the same time be lucrative for the protectors. Consider microcomputers in automobiles. Suppose some manufacturer put a chip in his cars that had a faulty sine/cosine routine that steered you off a cliff, or say a thousand drivers died and the chip was suspected. You might find a lawyer filing a case against the chip designer and the manufacturer.

Are manufacturers responsible? Although those making ballpoint pens do not take any responsibility for their usage in public restrooms, it is generally conceded that computers are a rather more powerful tool, and equally pervasive. I have here an excellent quote by Parry Rogers, IBM Director of External Affairs.¹ He says that the manufacturer must not get into any arrogant position of deciding what is best for society. The user has a major responsibility, too. And that is why I am addressing you today.

But Rogers does say that the manufacturer has a couple of responsibilities:

"First of all we must help the user to understand the limitations, dangers and exposures of this new tool. Secondly,... it is our responsibility to provide the user with the means to use the equipment in ways acceptable to society."

I think that our meeting here has followed this direction admirably, but to see the change in thinking over a span of years, I went back to Ike Auerbach's opening speech in the 1962 IFIP

¹ Interview in *Computing*, June 7, 1973, pp. 14-15.

Congress in Munich.² His title was "The Impact of Information Processing on Mankind." First he said that there were some potential dangers, as well as benefits, in the use of computers, and that they were hidden. He then shelved the dangers for some time and went on and on about the benefits. When he finally came back to the dangers visible in 1962, they were not too specific. He said that usage in government could be a two-edged sword, and that there can be less desirable results in the optimization of business, with automation. He then asked:

"Who is going to determine the criteria for a satisfactory solution to the problems that will be solved by the computer? Is efficient performance to be the sole criterion, or will human considerations also be taken in account? Who will be interpreting the computer outputs? Whose value judgments will be the controlling ones?...In other words, who is going to control the computer that controls the economy that eventually controls us?"

Computational Accuracy

Now this is all good sound stuff, but by that time the wings had already been falling off the Electra for six years. You may recall that small oscillations built up to big ones, and the vibration eventually overcame the structural design. And when wings fall off airplanes, people get killed.

I'll remind you that security was not a factor here. It was the calculating algorithms and programs that were wrong. Here it takes three to tango -- the program dancing with the data to the computational music. Don't forget that the security considerations for data manipulation are a lot more recent than computational method, and security has no meaning for incorrect programs. The AFIPS System Improvement Committee (Dick Mills and I are among the members) recognizes this. It just seemed top priority to attack the security problem first. But program correctness and computational accuracy are being worked on now, although perhaps too much at only the academic level so far.

Although the story in the Electra case has been somewhat disguised, the lesson to be taken is that floating point computation is fraught with hazards. It permits one to define the order of computation to be anything one wishes, say multiplying a by b and by c and by d, and then dividing consecutively by e, f, and g. When you have built up a huge number and then come down to a number of reasonable size, the precision of the floating point computation is destroyed, and some of those little factors that cause oscillation in the wings have disappeared. Not from the wings themselves, but from the computation.

Any of you who may have used IBM 1401 in the days of software floating point would have noticed that when a result came to zero, the exponent was

reduced only by the amount of the precision, not to the smallest exponent possible. That was a direct result of my moving to IBM from Lockheed, but I wasn't the programmer involved, just there at the time.

There were also some troubles with the UNIVAC 1107. When I started to work, there were seven computers in the field. I asked someone to check the range of floating point answers on both the IBM 7094 and the 1107. The 7094 reached down to numbers with exponents of 10^{-35} , but the 1107 produced zeros after 10^{-22} . It appears that the hardware man threw out the result upon underflow for the more significant part as well as the least. You can imagine what would have happened if some user ran programs interchangeably on both machines.

Dante's Blacksmith

Yet IBM wasn't really successful with its floating point arithmetic. W. Kahan, in a masterful survey,³ tells a horror story prefaced by the quote, "The bad blacksmith blames the iron."

"Mr. Z. was despondent when I first saw him. A graduate student of aeronautical engineering, he was trying to augment boundary layer flow past wings...if his idea worked, his reward would be a Ph.D. thesis and a job with a local aircraft firm. He was testing his idea on our university's computer, then an IBM 7090,...finally producing a graph from which he could read Success or Failure. He had just read Failure."

Dr. Kahan was testing a new logarithm routine, and used Mr. Z's program among several in his test. The graph moved to Success. Yet Z didn't know he had used this logarithm in his program, but it was; the basic exponentiation function was accomplished in this way. So Z had his correct answer in single precision, reconfirmed when IBM issued a revision to the double precision floating point software. But shortly thereafter the 7094 was installed; its built-in double-precision hardware pointed to Failure again. Erroneously, of course.

But Kahan asks:

"How often are engineers baffled by subtly wrong computations, thwarted in otherwise exemplary endeavors, and unable to uncover what went wrong? And how often is an engineer who expresses doubts about the computing system...regarded as if he were Dante's bad blacksmith?"

² I. L. Auerbach, "The Impact of Information Processing on Mankind," *Proceedings of IFIP Congress 1962* (Amsterdam: North Holland Publishing Co., 1963), pp. 9-13.

³ W. Kahan, "A Survey of Error Analysis," *Proceedings of IFIP Congress 1971* (Amsterdam: North Holland Publishing Co., 1972), pp. 1214-1239.

Risking the loss of a Ph.D. may not be so serious, but I have for my own purposes grouped computer usage to be:

1. Advisory -- not leading to decisions that directly affect humans.
2. Recommendatory -- offering possible decisions that humans can take.
3. Directive -- where the computer has been pre-programmed to take a decision and action, and will in fact so act unless countermanded in time.

Bear in Mind, Too

The last class represents the closest integration into human affairs. Here we come to the cases of computer-monitored patients dying in the hospital because the field engineer was unavailable -- the nuclear power plants where humans cannot possibly react fast enough to control emergencies -- the 7,000 airplanes that will be in the air at any one time over Los Angeles in 1980. For the latter, remember that the current FAA system is designed to handle 500 planes, but in fact has been saturating the computer at 150.

Many faults lie not in the computational methods and algorithms, but in the languages and compilers we use. My classic case is when Univac was trying to sell an 1107 to the U.S. Army. In the compilation of a FORTRAN program appeared a diagnostic indicating an entry into the middle of a DO loop. The General in charge said that was impossible; they had been running that program for three years. A programmer was detailed to examine the situation. He returned in a short while and said, "Sorry, General, three years of wrong answers."

You can see that these are not fictitious dangers that I am pointing out. There is a tendency to blind acceptance of everything the computer prints out. I remember in particular a friend who got economic statistics from the University of Michigan about

the way in which the national GNP affected Michigan industry. The report was loaded with computer printouts. I studied it and saw that the conclusions were spurious. So would he, if the results had been typeset, but that computer listing reeked with authority.

Some of you may have heard of the time that a local piano dealer here in Phoenix advertised that they were holding a sale on pianos because their computer fouled up and ordered too much inventory. I wrote a letter, on behalf of the ACM, saying that I would be glad to fix the program for them on the condition that if it turned out to be a human error, they would so acknowledge in their next advertising. A copy of the letter was sent to *Datamation*, because I received no answer. The reason, of course as *Datamation* discovered, was that the piano dealer *didn't have* a computer. They just depended upon the public's propensity to believe blaming the computer.

Conclusion

To conclude, although the manufacturers of ballpoint pens can't control the graffiti, many computer manufacturers feel an obligation to be helpful and point out pitfalls in future usage so that you can make adequate preparation against them. Security is important but computational correctness, program correctness, operational correctness, and safety, etc., are also very important. I ask that you pass the word, for lawyers in computer malpractice can make more than in the medical field. One doctor services only so many patients, but a computer program can service millions; 35 million in the case of a credit system studied by Dr. Robert Goldstein in his book *The Cost of Privacy*.

I have given you no more than a flavor of other social aspects of computers, but I hope that you can agree that others are equally as serious as security and privacy.

$$Z = \sum_{k=1,2}^3 \sum_{l=1,2}^3 \sum_{m=1,2}^3 \dots 10^{3+3k} + 0^{3+3k}$$

Although an expression containing a summation operator must be provided by one or more FOR specifications in order to be meaningful, FOR variables may also be used in expressions that do not contain the & operator. For example:

7 FOR A = 3:1:2, FOR B = 1:2, X = A*B

In these cases, the expression will be evaluated separately for each possible combination of FOR values for all zones in FORTRAN. The output from the example expression list above would appear as:

```

X
  3  6  9  12  15  18  21  24  27  30
  6  12  18  24  30  36  42  48  54  60
  9  18  27  36  45  54  63  72  81  90
  12  24  36  48  60  72  84  96  108  120
  15  30  45  60  75  90  105  120  135  150
  18  36  54  72  90  108  126  144  162  180
  21  42  63  84  105  126  147  168  189  210
  24  48  72  96  120  144  168  192  216  240
  27  54  81  108  135  162  189  216  243  270
  30  60  90  120  150  180  210  240  270  300
  
```

If a label variable is used, such as (X) in the above example, the last letter of the value is remembered for the variable.

Making Microfiche Irresistible

by Robert W. Bemer
Honeywell Information Systems



Robert W. Bemer is a senior consulting engineer at Honeywell Information Systems, Phoenix, AZ. He was formerly a staff consultant and editor of the Honeywell Computer Journal. Before joining Honeywell he was with General Electric as manager

of systems and software engineering integration, and had previously held positions with the Univac Division of Sperry Rand, IBM Corporation, Lockheed Missiles and Space Division, Marquardt Aircraft and the RAND Corporation.

Mr. Bemer holds an A.B. degree in mathematics from Albion College and a certificate in aeronautical engineering from Curtiss-Wright Tech. He is a fellow of the British Computer Society, and has written numerous articles for computer publications.

ABSTRACT

Designing hardcopy and microfiche formats to exist interchangeably and coequally brings multiple benefits. An unusual feature of this new method is that the fiche copy as viewed in the reader is larger, not smaller, than the original copy! A single fiche may contain either 392 or 420 pages, each with 3/4 of the capacity of the usual A4 page. Thus the information density is tripled, without impairing (and in fact enhancing) readability.

Although I became one of the pioneers in 1971 by issuing a journal with an integral microfiche copy (in a pocket inside the back cover of the Honeywell Computer Journal — no longer in production), there still remains the feeling of being a novice. For a bullfighting parallel — although a novice, I am nevertheless a microficianado!

I sense that microfiche is retarded from full acceptance (except by students) because too much of it is 1-to-1 copy of either a typewritten page (perhaps of insufficient quality) or a printed page designed without consideration for fiche. Readability must be enhanced before acceptance will be complete, particularly for those of us with bifocal lenses.

The computer world is also bending to the forces of increased costs for paper and mail. Indeed, a recent article by Rogoff [1] argues that it is cheaper to send letters in encoded form, by packet switching, than in typed form by mail. I have for some time followed the practice of encoding by entering my letters into a computer and photo-composing the text. Gilbert Jones of IBM warns us that the day of the paper file cabinet is nearly over.

MOTIVATION AND SERENDIPITY

I think that I have stumbled onto an advantageous way of using microfiche copy. It might not have worked, had not several different processes become cost-effective at about the same time. As it is, there seems to be a promise of (as the title says) making the use of microfiche irresistible.

The method came from a study of reducing the consumption of 11" by 14" printer paper — not for the usual COM applications where multiple copies are disseminated for use and archival storage; Honeywell Information Systems was already doing that extensively. Our problem was with programmers and their huge dumps and other listings — thrown away after a glance or two. Would it be practical to replace them by COM masters that would never be copied, just discarded after diagnosis? The answer was *yes*, with a better than 2:1 cost advantage over printer paper.

A Self-Descriptive Page: This page shows the method by example. Anyone can verify it by filming the page at 24X and reading the fiche in a COM reader. The upper (or lower) pairs of "little pages" will fill the screen. (Columns are slightly shortened because J. Micrographics doesn't use A4 paper). The type is the actual Helvetica Medium font that we use.

As our study showed that 95% of the cost would be in fiche production, and only 5% for the fiche readers, it became evident that the programmer work station is likely to consist of an online terminal and an offline fiche reader for the 11" x 14" paper images. But with readers amortized so effectively and cheaply by the diagnostic function, it became irresistible to consider putting programming manuals and other documentation on fiche also.

We have always had special problems with manuals. Software is revised and corrected quite often, but the economics of hardcopy production and paper-oriented methods work against complete reissuance of manuals conforming to a specific software release.

Once we envision a programmer work station consisting of a terminal and microfiche reader, it is seen desirable to have all program manuals and useful documentation available in microfiche form at that station, for reasons of efficiency and storage. It has been argued that once we get to this point it may be more economical and practical, providing video terminals are used, to store manuals in the computer itself for display of portions as needed. However, there are advantages for using fiche in conjunction with terminals:

- Fiche produced photographically can contain diagrams, tables, photographs, and graphic-quality distinctions in text (i.e., variation in type size, weight (bold), slope (italic)) that enhance the extraction of meaning. Admittedly some such distinctions are possible in a more limited way on softcopy terminals.
- The manual may be in constant display while using the terminal in a working (not study) mode. For example, a procedure outlined in the fiche manual may be followed step by step. This argues favorably for using nothing but dual-carriage readers.

A question now arises on the mechanics of producing a manual on fiche. Certainly no one wishes to see a reproduction of a typewritten page. We

N

Choosing the Page Size

Because we wish to read manuals (with text, diagrams, and pictures) using the same reader and magnification with which we read COM output, the design problem was this:

- What size of printed area, photographed at 24X by step-and-repeat camera, will permit a left- and right-hand page pair to occupy approximately the same screen area as the printed area of a 11" x 14" printer page (COM version) produced at 48X?
- The aspect ratio (width to height) should be equally suitable for the interior area of a standard 6" by 9" book, for the fiche reader, and for A4 (or 8.5" x 11") paper.
- Four pages should be photographed as a single frame for the standard 98-image fiche.

To get sufficient print quality we used the photocomposition methods developed for the Honeywell Computer Journal. Here our computer text editor was modified to produce input to the Page 1 System, which then carried out the hyphenation, justification, and photocomposition.

To derive the printed area (column width by column length plus page number) we examine the physical grid of the 24X fiche. It is 10 mm wide by 12.5 mm high. At 24X the corresponding full-size grid is 240 mm wide by 300 mm high (A4 paper is 210 by 297). Putting 4 pages in this area means that each would be in a grid 120 mm wide by 150 mm high. Alternately, each pair of pages has a grid 240 mm wide by 150 mm high. Figure 1 shows the actual dimensions chosen for the printed area.

Although these dimensions work, so might other choices. The actual choice was also influenced by hardcopy considerations. The 4.5" x 7" interior of a 6" x 9" book page has an aspect ratio of 1.55 (a basic consideration because a 37% photographic enlargement of our original copy will provide for a hard copy book of this size).

The page size must also serve for hard copy directly as it comes from the photocomposing unit. It turns out that the 83.5 mm x 128 mm size fits very nicely on A6 paper. We print these, add Bristol board covers, and punch two holes for individual rings. These permit the manual to fold flat at any place, for convenience at a terminal (see Figure 2).

are fortunate that our manuals have, for many years, been produced by a computer text editor system. The drawbacks have been:

- They are attuned to 8.5" x 11" paper, which is too big to handle when working at a terminal.
- Output was by IBM or DATEL typewriter-style terminals having constant (pica) spacing that is more difficult to read than graphic quality type. Such readability problems are worsened when fiche is used.

MECHANICS OF TEXT AND IMAGE PRODUCTION

A reasonable and economic goal is to use the same fiche reader to view both the manual and the COM output of printer pages without the added expense of a second lens for different magnification, and without the irksome task of changing the lens setting. In short, everything should be read at a constant 48, 42, or 36X.

One's first inclination is to produce the manuals via the COM units that produce the printer page images, but these drawbacks are encountered:

- The COM units now in operation cannot produce a full spectrum of characters and symbols, bold and italics, tables, etc. Even an upper and lower case alphabet is difficult to come by.
- Only pica (constant) spacing is available now.
- Fonts, in only one size, are not of full graphic quality.
- Although our new graphic COM equipment may solve all of these problems, photographs are still difficult, and a method is still needed for those locations not having this equipment.

Future equipment to meet all these requirements can be expensive, so the present situation may be fortunate. It forces us to traditional camera methods producing 24X fiche from 98 hardcopy images of A4 size.

The happy result is that we are now committed to produce fiche at 24X that will be read at 36X or 48X. Thus the image that will be viewed is 1.5 to 2 times as large as the copy that was photographed! This is a fine feature, for one of the hindrances to widespread fiche usage has always been poor or unacceptable legibility.

N + 1

Choosing Font and Size

The fiche that is produced by this method will now have, in the usual row of 14 images, 28 images in the top half and 28 images in the bottom half, for a total of 392 pages per fiche. (If the heading area of the 48X format is suitable, we can get 420 pages per fiche.) What then is the most text that could or should be placed upon such pages? This is a function of point size. Our 83.5 mm-wide column has a 236-point capacity, quite standard for double-column printing. In fact, an optional A4 layout is possible by not breaking the columns vertically. However, this method is only suitable for hard copy.

I have found that 7-point type with a 2-point lead is very satisfactory. At the full 48X magnification this appears as 14-point with a 4-point lead, much like a child's primer. At 36X, or 75%, it is nearly 11-point, with a 3-point lead, and still very readable. In 6 x 9 book form this becomes 9.6-point -- again very suitable for reading from hardcopy text. I chose Helvetica Medium as the font, for its fairly thick strokes give good readability on microfiche. No serif fonts are acceptable, to my mind. Of course, the possibility of 8-point has not been ruled out, but we note that 7-point type is still very readable in our hardcopy manuals for terminal usage. This is due to both the graphic quality and the fact that they contain reference material, not novels. Perhaps you have reached this point without undue annoyance, even though it is in 7-point for the express purpose of illustrating the method.

Choosing the Image

The remaining option was whether the fiche should be a positive or a negative. Our COM customarily comes out negative for computer output, and this is the easiest photographic process. Our expert likes the negative form because foreign material, which fiche unavoidably attracts, does not show up so much. Despite all this, some programmers prefer the positive form, and we let them specify their choice. When you can get, for a dime, a fiche containing 2 or 3 complete manuals -- I feel we can afford to give the programmer a new copy whenever too much foreign material collects!

Figure 3 shows the entire timesharing manual for the HIS Level 66 computer system (index at bottom). A black matte template is used for shooting. I feel that it enhances the layout by clean demarcation of images.

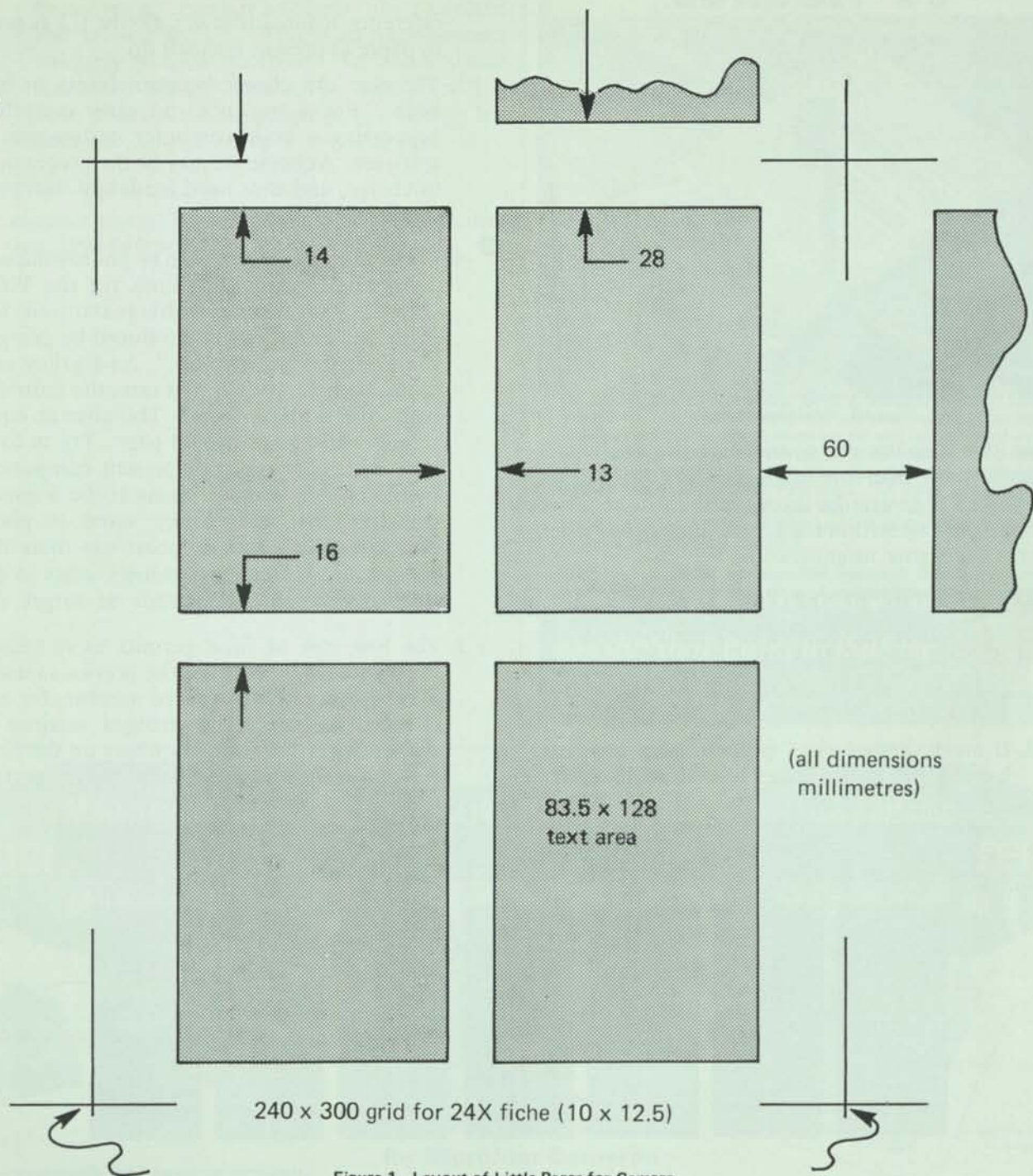


Figure 1. Layout of Little Pages for Camera.

ESSENCE OF THE METHOD

We now have a useful, effective and economical method, by the simple expedient of assaulting two accepted conventions:

- That the fiche process is subordinate to hard-copy production (sort of an afterthought, if you will), and that either COM or step-process filming must take the copy as it finds it.

- That the film will always be read at a magnification equivalent to, or less than, the reduction in production.

These conventions are now abrogated. Copy and layout are designed with both fiche and hardcopy considerations in mind. They are compatible, and in many cases the hardcopy production is 1:1 offset from the same master used for the fiche. Now the fiche process is not subordinate; it is coequal. Be-

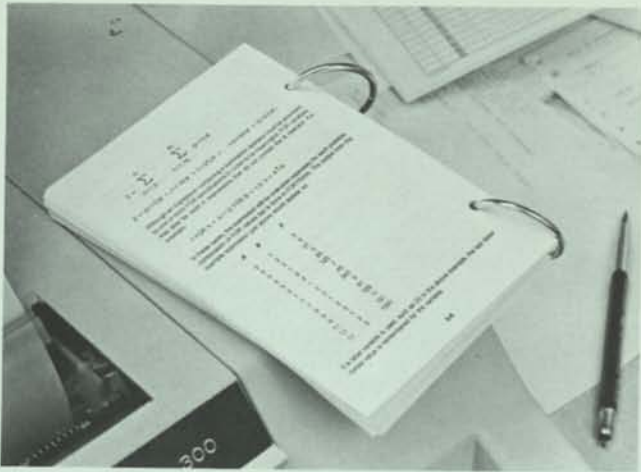


Figure 2. Hardcopy Manual.

cause this is so, we can consider reading at a higher magnification than the original reduction. All we had to do was design the layout so that the required portion *did* fall within the screen dimensions when read at the higher magnification.

WHAT ARE THE BENEFITS?

1. Removal of the usual 8-foot shelf of manuals that the programmer keeps to consume office and filing cabinet space. Not to mention the pile of listings on his table or floor.
2. If much desired, the 7-point original copy can be used by direct offset to make a hardcopy

reference manual that costs only 1/3 as much in paper as present manuals do.

3. The user can choose between forms, or have both. For a trip, one can carry everything supporting a large computer system and its software. At home, he may be the programmer in charge, and thus need hardcopy for annotation.
4. Fortuitously, two pages can be photocomposed on a single galley (12" limit for the Video-comp). With column width guaranteed, four pages per galley can be produced by going, as the printers say, "two-up." As a galley costs us less than \$2.40 each, flat rate, the individual page cost is about \$0.60! This gives an equivalent rate of \$0.80 per A4 page. Try to touch that with your typist or present composition methods! In fact, it seems to be a general principle that the money saved in photocomposing at a smaller point size more than offsets any incremental camera costs in producing stripped mechanicals at target size.
5. The low cost of fiche permits us to reshoot the manual in the form of the previous version, except annotated by circled numbers for each changed section. The changed sections are shown in revised form elsewhere on the fiche. As soon as the user comes to a changed section

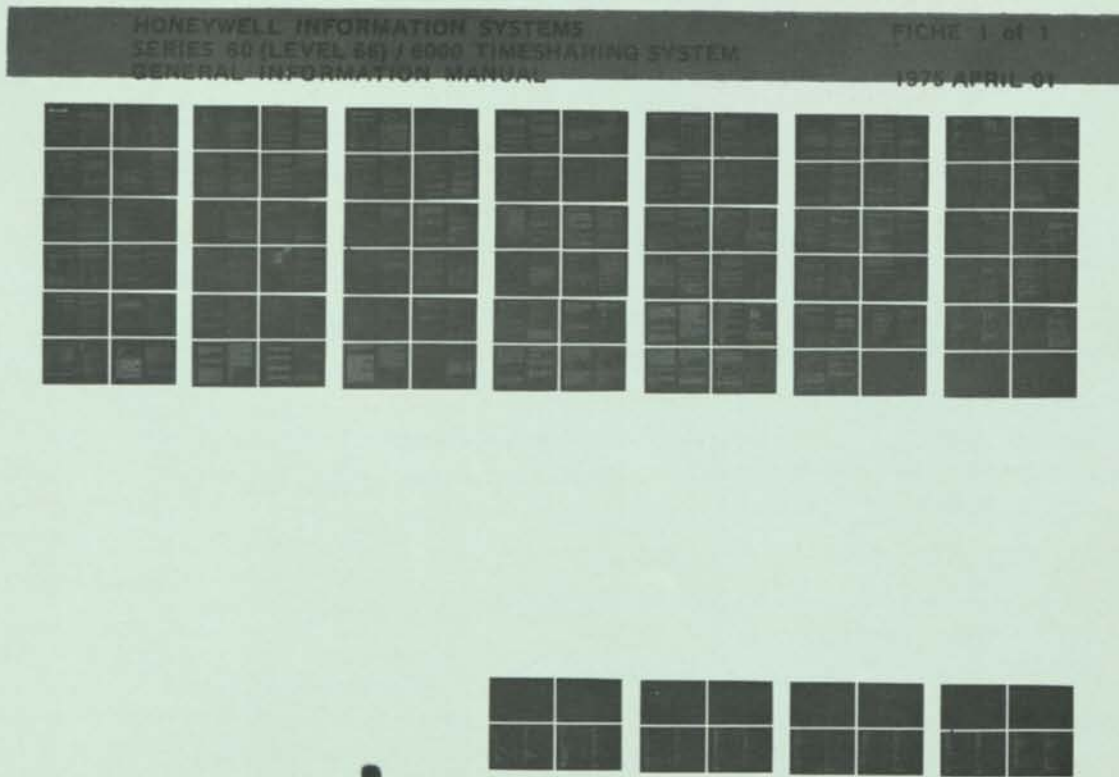



Figure 3. Fiche Version of Extensive Computer Manual.

he moves the carriage to display the corrected and revised section. This is an improvement uniquely possible to fiche. To know simultaneously what it used to be, and what it is now, conveys real intelligence. A change bar does not.

REFERENCES

1. Mortimer Rogoff, "The case for electronic mail," Modern Data, 1974 December, 34-37. 

GORDON 35 Portable Microfilmer



The Gordon 35 is an outstanding, portable, planetary microfilm camera which is ideal for location filming where original documents cannot be removed from their source. It serves equally well for permanent installation and is designed for a wide variety of applications. The portable, compact, precise planetary single-lens reflex microfilm camera has a vacuum platen for maximum sharpness. The entire unit, including case, weighs only 27 pounds and can be set up in less than two minutes on only 14"x20" of desk space. A 16mm Conversion Kit is available. Write for data and price.

A-G-E Inc. can fill all your microfilm equipment requirements. Write your specific needs and send for our free microfilm equipment catalog.

SERVING THE WORLD

alan gordon enterprises inc. 

5362 Cahuenga Blvd., North Hollywood, CA 91601

Telephone: (213) 985-5500

TWX: 910-321-4526 • Cable: GORDENT

The Most Dependable DENSITOMETER of them all!



Model MR-2
by Fotomatic

A high-sensitivity light measuring instrument, employing the cadmium sulfide photo-resistor. For use in Microfilming, Graphic Arts, Industrial Laboratories and Medical and Research Laboratories. Determines exposure times, aperture settings, density values, contrast ranges and maintains light constants. Not for color. Entire circuit is voltage stabilized. No amplifying tubes or transistors which can drift with age. Covers density range of 0 to 3.00 in increments of .02. Reads diazo film directly.

Meter-Power Supply	\$266.00
Transmission Attachment	157.00
Incident Light Probe	33.00

See your dealer or write . . .

FC FOTOMATIC COMPANY
A Division of Diagnostic, Inc.

4030 West 10th Street • Indianapolis, Indiana 46222

EXPOSURE CONTROL



The PLANATROL Automatic Exposure Control for Microfilm Cameras

Planatrol controls automatically set the proper lighting to compensate for variations in surface brightness caused by different shades, colors and textures of documents . . . even correct for line voltage changes. Every exposure is correct. Every negative has correct density. No more manual light measurements, no more setting light intensities or shutter speeds. No wiring necessary. Just plug it in. For incandescent-lighted planetary cameras up to 2000 watts, including Recordak MRD-1 and MRD-2 cameras.

\$633.00 F.O.B. Indianapolis

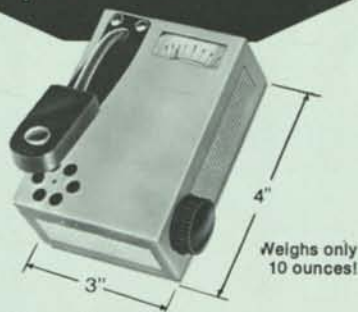
ALSO Automatic exposure control kits available for rotary cameras.

See your dealer or write . . .

FC FOTOMATIC CORPORATION
A Division of Diagnostic, Inc.

4030 West 10th Street • Indianapolis, Indiana 46222

A portable DENSITOMETER you can count on!



The Romelco by Fotomatic Photoelectric Transmission Densitometer

ONLY \$97.00 F.O.B. INDIANAPOLIS

Small, accurate, easy to use and read • Measures densities of microfilm from 0.60 to 1.40 . . . below 0.60 with extra setting • Reads diazo film directly • Scale divided into 0.05 increments that may be interpolated accurately to 0.02 • Factory calibrated to ± 0.02 • Solid state for long life • Not for color • Uses two AA mercury batteries for maximum life and stability • Instructions, calibrated film strip and batteries included.

See your dealer or write . . .

FC FOTOMATIC COMPANY
A Division of Diagnostic, Inc.

4030 West 10th Street • Indianapolis, Indiana 46222

ASCII - The Data Alphabet That Will Endure

Robert W. Bemer
Honeywell Information Systems, Inc.
Phoenix, Arizona, US

A standard data alphabet is indispensable to understanding communication and reading data in machine-encoded form (not spoken, not written or printed). ASCII (the ISO Code) has, by design, capabilities for expansion and extension not inherent in any other code. The many billions of dollars worth of ASCII-based communication and computation equipment is the best prepared for the coming fields of networking, electronic funds transfer, text processing and photocomposition, the automated office, etc.

The status and prospects of this healthy 12-year-old are explored.

Keywords: Alphabet; ASCII; character; code; ISO; symbol

1. ASCII is a True Alphabet

Because ASCII [1] and its international identical twin, the ISO Code [2], are actually called "coded character sets" in the formal standards, I must begin by explaining why I use the term "alphabet" instead. One reason is that another international identical twin is called the C.C.I.T.T. Working Alphabet No. 5. For the others, some definitions are necessary. Even though I dislike Webster's Third International (sic) Dictionary intensely, here are some of the things it says that "alphabet" means:

- 1a. Any particular set of letters with which one or more languages are written, especially such a set of letters arranged in customary order.
- 1b. Any set of characters with which one or more languages are written, whether these characters are letters (sense 1a), signs of a syllabary, or other basic units of writing.
- 1f. The alphabetic system of writing, as distinguished from syllabic, ideographic, and other systems.
- 1h. Any system of signs or signals, visual, auditory, or tactile, that serve as equivalents for the usual written letters of the alphabet.
- 1i. A particular set of names used to designate the various letters in the alphabet (the pronouncing alphabet used in civil aviation).
- 1j. In cryptology, a set of one-to-one equivalences between a sequence of plaintext letters and the sequence of their cipher substitutes.

Are not ASCII and the ISO Code actually alphabets in every such sense?

In sense **1a**, it contains letters, and they are arranged in the customary order (but not collating sequence, because of the dual case representations of the letters) by the numerical order of their bit encodings. More than one language can be written with it; as the international code, the most prevalent languages using the Roman alphabet can be written. Note particularly, in the official reference version, that provision is made for the extra Scandinavian letters, located in the proper position (although the usage is not the same in Denmark, Norway, and Sweden).

In sense **1b**, it contains other units of writing. Punctuation is there, as are underscore and other common symbols. Diacritical elements exist for forming compound and accented letters, thus bringing more Roman-based languages within its capability.

In sense **1f**, it certainly does not have syllabic or ideographic characteristics. So it is not excluded from being called an alphabet for these reasons.

In sense **1h**, the encoded representations are the equivalents. In fact, this is the definition of most interest to us. Note that after 90 years of encoding (starting with punch cards for the 1890 Census), Webster's Dictionary fails to give specific status to this manifestation.

Senses **1i** and **1j** may not appear to be pertinent now, but they are there for a reason, and we shall return to them.

2. Why Alphabets?

Alphabets comprise a class of methods to record knowledge for transmittal to others. To transfer knowledge we must transfer information; to transfer information we must transfer data.

There are, of course, other methods using basic elements at higher levels of complexity -- such as syllables, ideographs, etc. However, the primitive ASCII (ISO Code) is *the* worldwide standard for exchange of data and information. It continues to be the standard because the primitives are representable in alternate but related ways.

It is interesting to speculate what might have happened if our forebears had developed the phonograph or tape recorder *before* writing with alphabets. Would we be as deeply into databanks as we are now? Would standards have been developed for speech sounds in analog form, using the computer to discriminate and remove differences in people's voices? Would Confucius have said that one analog picture was worth a thousand digital words?

One suspects it might be difficult to search such a databank, however. At least it appears that we do not have the methods yet. For example, in reply to the Senate Committee asking for information from the Nixon tapes, the White House says that they have not been classified as to content, and it would take listening to them in their entirety. In other words, a linear realtime search -- and we know how inefficient that can get as databases grow larger.

3. Why ASCII Survives and Grows

My personal history or view of the development of the ISO Code and ASCII [3] tells of millions of dollars of careful international effort and planning spent in its creation. But that is minor, amortized by many billions now invested in communications and computer equipment that operate via ASCII. It also tells of the IBM code called EBCDIC, a result (according to Fred Brooks, one of the chief designers of the 360) of forced announcement before ASCII peripheral equipment could be completed. Although the 360 was *said* to have ASCII capability, it was never realized in the software.

If computers, in substantial portion, do not operate in ASCII as native mode, then why will EBCDIC not be the survivor? Many people, both in and out of government, have blithely assumed that EBCDIC will -- and continue to invest money in software and operations based upon EBCDIC.[4] They are going to be very surprised, because *IBM knows* that EBCDIC will eventually be subordinate! [5]

Why? Because EBCDIC is not, like ASCII, the result of meticulous design.

"... it would appear that no single 'computer code' can be completely adequate, and that insistence on a single code for all purposes would be counterproductive. Rather, the Federal Government should maximize the benefits to be accrued from taking advantage of our growing technological ability to live in a multi-code world ..." [5]

In ASCII, the controls are all located in the leftmost two columns. It is compact, extensible, expandable, and even subsettable. It can grow easily into an 8-bit code (expandability), or into 9-bits, 10- , or anything. At any level of byte size, it can be extended to encompass alternate sets of characters; keeping the same control columns, various pages can be substituted for the other columns. The methods for expansion and extension are also standardized.[6,7] Sets having sufficient utility may be registered for international usage, via the French standards body AFNOR, which holds the secretariat for international code standardization within ISO TC97. The vehicle for doing this is the ESCape character.[8] Various pages are registered with unique ESCape sequences.[9,10]

4. Code Extension

In the extension procedures, the existing 7-bit ASCII is divided into control and graphic portions. The first two columns of code -- the controls -- comprise the C0 set; the other six columns --the graphics -- comprise the G0 set. The extended set first removed from basic ASCII is similarly divided into the C1 and G1 sets. Obviously such sets could be adjoined in the 8-bit form, and the USSR [11] and Japanese [12] standards are excellent examples of so doing.

4.1 Extended Control Sets

C1 sets can be, and have been, designed for many purposes. The one furthest progressed to agreement is that for softcopy controls, for CRT display screens [13,14].

"... The major difficulties at present are in using the established 'control' characters with devices that had not been invented at the time the code was, or in extending the 'graphic' symbol set to meet new application requirements ..." [5]

Work is continuing in both ECMA (the European Computer Manufacturers Association) and ANSI X3 to get agreement sufficient for final registration. The original work of X3L2 was for softcopy controls to be in an *expanded* set (8-bit code), but that is presumptuous.

				b.	0	0	0	0	1	1	1	1
				b.	0	0	1	1	0	0	1	1
				b.	0	1	0	1	0	1	0	1
				0	1	2	3	4	5	6	7	
b.	b.	b.	b.	0		SP	0		P			
0	0	0	1	1			1	A	Q			
0	0	1	0	2		"	2	B	R			
0	0	1	1	3			3	C	S			
0	1	0	0	4		\$	4	D	T			
0	1	0	1	5			5	E	U			
0	1	1	0	6			6	F	V			
0	1	1	1	7			7	G	W			
1	0	0	0	8		(8	H	X			
1	0	0	1	9)	9	I	Y			
1	0	1	0	10		*		J	Z			
1	0	1	1	11		+	;	K				
1	1	0	0	12		/	<	L				
1	1	0	1	13		-	=	M				
1	1	1	0	14		.	>	N				
1	1	1	1	15		/		O				

Figure 1. COBOL Character Set

ISO Technical Committee 46 (Documentation), in its Subcommittee 4 (Automated Documentation) has a working group on bibliographic codes. Its first candidate for registration as a C1 control set is a set for bibliographic controls [15] to be embedded in text to delimit certain special data. This C1 set contains four classes of characters -- annotation controls, filing controls, reference controls, and subject designators. Major credit should be given here to Dr. Ernst Kohl of the Bavarian State Library in Munich.

Although little work has been done, other C1 control sets are envisioned for typographic control -- to vary the font, weight, slope, size, and spacing, etc. of the graphic characters. Other sets could be envisioned for fields such as process control, animation and other graphics applications, sewing machines, etc. Do you think the last one far-fetched? Singer has already announced a machine with a microcomputer, and is there any reason to think that future models won't use ASCII characters, in a hand calculator type of display, to give instructions and options available?

				b.	0	0	0	0	1	1	1	1
				b.	0	0	1	1	0	0	1	1
				b.	0	1	0	1	0	1	0	1
				0	1	2	3	4	5	6	7	
b.	b.	b.	b.	0		SP	0		P			
0	0	0	1	1			1	A	Q			
0	0	1	0	2			2	B	R			
0	0	1	1	3			3	C	S			
0	1	0	0	4		\$	4	D	T			
0	1	0	1	5			5	E	U			
0	1	1	0	6			6	F	V			
0	1	1	1	7		'	7	G	W			
1	0	0	0	8		(8	H	X			
1	0	0	1	9)	9	I	Y			
1	0	1	0	10		*	:	J	Z			
1	0	1	1	11		+		K				
1	1	0	0	12		/		L				
1	1	0	1	13		-	=	M				
1	1	1	0	14		.		N				
1	1	1	1	15		/		O				

Figure 2. Fortran Character Set

4.2 Extended Graphic Sets

G1 sets are further along. ISO TC46/4/1 has tabled Draft International Standards for Latin (DIS-5426), Greek (DIS-5427), and Cyrillic (DIS-5428). We may presume the latter is in harmony with [11]. Under study are sets for mathematical characters and the African languages. Proposals have been solicited for such languages as Arabic, Kata Kana, Kanji, etc. It is permissible for a G1 set to be a partial replication of the basic G0 set of ASCII; indeed, many are very similar, with the lower case being replaced by the new alphabet.

Although we have seen the ISO assignments for natural languages to be in the jurisdiction of TC46, TC97 (Computers and Information Processing) has retained authority to make assignments for programming languages. The work has been concentrated in G1 sets for COBOL (figure 1), Fortran (figure 2), Basic (figure 3), and PL/I (figure 4). A table for APL is being constructed. ALGOL presents different problems.[16]

				b. 0 0 0 0 1 1 1 1									
				b. 0 0 1 1 0 0 1 1									
				b. 0 1 0 1 0 1 0 1									
				0 1 2 3 4 5 6 7									
b.	b.	b.	b.										
0	0	0	0			SP	0			P			
0	0	0	1			!	1			A		Q	
0	0	1	0			"	2			B		R	
0	0	1	1			#	3			C		S	
0	1	0	0			\$	4			D		T	
0	1	0	1			%	5			E		U	
0	1	1	0			&	6			F		V	
0	1	1	1			'	7			G		W	
1	0	0	0			(8			H		X	
1	0	0	1)	9			I		Y	
1	0	1	0			*	:			J		Z	
1	0	1	1			+	;			K			
1	1	0	0			,	<			L			
1	1	0	1			-	=			M			
1	1	1	0			.	>			N		^	
1	1	1	1			/	?			0		_	

Figure 3. BASIC Character Set

One may be tempted to think of these not as G1 sets but rather as subsets of the G0 set, standard ASCII. But note that they are incompatible in minor ways, particularly for PL/I, which was the cause of considerable difficulty in stabilizing ASCII. So perhaps the G1 status is an easy solution.

4.3 The Registry Method

A responsible standardizing body with a specific proposal makes application to AFNOR, acting as agent for ISO TC97/SC2. Applications may be for graphic sets (G1, etc.), C0 or C1 control sets, a single control character, or a code requiring special interpretation. The approval procedure is defined in [10]. A unique ESCape sequence is assigned. It is here that definitions *1i* and *1j* for alphabet become applicable. The ESCape sequence, as adjoined to any following character before termination, becomes a name for the alternate characters and alphabet, in one-to-one equivalence. Thus all of the world's symbol and alphabets may be represented uniquely for interchange.

				b. 0 0 0 0 1 1 1 1									
				b. 0 0 1 1 0 0 1 1									
				b. 0 1 0 1 0 1 0 1									
				0 1 2 3 4 5 6 7									
b.	b.	b.	b.										
0	0	0	0			␣	0			P			
0	0	0	1			!	1			A		Q	
0	0	1	0			"	2			B		R	
0	0	1	1			#	3			C		S	
0	1	0	0			\$	4			D		T	
0	1	0	1			%	5			E		U	
0	1	1	0			&	6			F		V	
0	1	1	1			'	7			G		W	
1	0	0	0			(8			H		X	
1	0	0	1)	9			I		Y	
1	0	1	0			*	:			J		Z	
1	0	1	1			+	;			K			
1	1	0	0			,	<			L			
1	1	0	1			-	=			M			
1	1	1	0			.	>			N		␣	
1	1	1	1			/	?			0		_	

Figure 4. PL/I Character Set

5. How IBM Can and Will Use ASCII

EBCDIC is a sparsely settled code that utilizes the 8-bit capability of 256 characters ineffectively. The collating sequence(s) are not easily derivable from the numerical values of the coded representations. The controls are intermingled with the other characters, so that it can not be extended by paging, as ASCII can. It has only one redeeming virtue -- one-to-one correspondence with ASCII via a common character set as represented in punch cards! (See figures 5, 6)

"The interesting observation is that if two character codes each have the same symbol set, and if each meet the requirement of no symbol ambiguities for the same bit pattern (no duals), then automatic context-free translation between the two character codes is a trivial task ... The operating cost of translation (between two such character codes) concurrently with preparing or accepting an interchange message is trivial in today's systems and will be more so in tomorrow's LSI machines." [5]

				b ₈	b ₇	b ₆	b ₅												
				0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
b ₄	b ₃	b ₂	b ₁	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	0	0	00															
0	0	0	1	01															
0	0	1	0	02															
0	0	1	1	03															
0	1	0	0	04															
0	1	0	1	05															
0	1	1	0	06															
0	1	1	1	07															
1	0	0	0	08															
1	0	0	1	09															
1	0	1	0	10															
1	0	1	1	11															
1	1	0	0	12															
1	1	0	1	13															
1	1	1	0	14															
1	1	1	1	15															

Figure 5. Hole Patterns Assigned By Code

	12	11	10	09	08	07	06	05	04	03	02	01	00	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
	02/08	02/13	03/00	02/00	07/11	07/12	07/13	11/10	10/08	11/01	11/09	06/00	12/03	12/10	13/01	13/08	8	1											
1	04/01	04/10	02/15	03/01	06/01	06/10	07/14	13/09	00/01	01/01	08/01	09/01	10/00	10/09	09/15	11/11	9	1											
2	04/02	04/11	05/03	03/02	06/02	06/11	07/03	13/10	00/02	01/02	08/02	01/06	10/01	10/10	11/02	11/12	9	2											
3	04/03	04/12	05/04	03/03	06/03	06/12	07/04	13/11	00/03	01/03	08/03	09/03	10/02	10/11	11/03	11/13	9	3											
4	04/04	04/13	05/05	03/04	06/04	06/13	07/05	13/12	09/12	09/13	08/04	09/04	10/03	10/12	11/04	11/14	9	4											
5	04/05	04/14	05/06	03/05	06/05	06/14	07/06	13/13	00/09	08/05	00/10	09/05	10/04	10/13	11/05	11/15	9	5											
6	04/06	04/15	05/07	03/06	06/06	06/15	07/07	13/14	08/06	00/08	01/07	09/06	10/05	10/14	11/06	12/00	9	6											
7	04/07	05/00	05/08	03/07	06/07	07/00	07/08	13/15	07/15	08/07	01/11	00/04	10/06	10/15	11/07	12/01	9	7											
8	04/08	05/01	05/09	03/08	06/08	07/01	07/09	14/00	09/07	01/08	08/08	09/08	10/07	11/00	11/08	12/02	8	8											
9	04/09	05/02	05/10	03/09	06/09	07/02	07/10	14/01	08/13	01/09	08/09	09/09	00/00	01/00	08/00	09/00	9	8	1										
10	05/11	05/13	05/12	03/10	12/04	12/11	13/02	14/02	08/14	09/02	08/10	09/10	14/08	14/14	15/04	15/10	8	8	2										
11	02/14	02/04	02/12	02/03	12/05	12/12	13/03	14/03	00/11	08/15	08/11	09/11	14/09	14/15	15/05	15/11	5	8	2										
12	03/12	02/10	02/05	04/00	12/06	12/13	13/04	14/04	00/12	01/12	08/12	01/04	14/10	15/00	15/06	15/12	5	8	4										
13	02/08	02/09	05/15	02/07	12/07	12/14	13/05	14/05	00/13	01/13	00/05	01/05	14/11	15/01	15/07	15/13	9	8	5										
14	02/11	03/11	03/14	03/13	12/08	12/15	13/06	14/06	00/14	01/14	00/06	09/14	14/12	15/02	15/08	15/14	9	8	5										
15	02/01	05/14	03/15	02/02	12/09	13/00	13/07	14/07	00/15	01/15	00/07	01/10	14/13	15/03	15/09	15/15	9	8	7										

Figure 6. Code Assigned To Hole Patterns

Anyone with a \$15 hand calculator realizes how cheap a microelectronic chip must be, for his calculator does a more complex job than the job of converting back and forth between the ASCII and EBCDIC encodings for the same character.

So let us postulate a very inexpensive chip inside IBM computers. It converts from EBCDIC to ASCII or from ASCII to EBCDIC without any delay as input-output or other operations are executed. Two questions must be asked:

- Is the data EBCDIC or ASCII?
- Does the program expect EBCDIC data or ASCII data?

Imagine subscribing the comparison instruction by EBCDIC or ASCII tags. I have a master file in EBCDIC, against which I run an update tape in ASCII. My program says "Compare the keys on an ASCII basis". The CPU, noting that the key of the master file is in EBCDIC, routes it through the chip before attempting to compare it to the key from the update record. My program can also give instruction to convert the entire updated file to ASCII automatically as it is being stored.

Thus the feasible technique. How about some signs that it will be so?

- Note IBM's commitment to word processing and photocomposition. According to B. O. Evans:

"There are similar requirements for the ability to use differing codes in differing contexts to represent different graphic needs. In particular, the development of 'end user' devices, such as photocomposers and interactive displays, requires greatly expanded symbol sets to be developed for some applications -- more symbols than can be contained directly in a 7- or 8-bit code. Thus, specific codes (of however many bits are required to represent a symbol) might well be developed for data interchange in certain application areas utilizing such devices." [5]

- Note IBM's 6250 cpi magnetic tape, which departs from cross-tape parity checking and recognizability of code without programmed knowledge.
- Note IBM's firm and continuing insistence that data control procedures be bit-transparent and not byte-oriented -- particularly not 8-bit-byte-oriented.

"We expect to see machine architectures having the flexibility to adapt efficiently to as-yet-undefined code structures without disruption of existing applications at any time even after the system was installed." [5]

- Note SDLC and SNA. When you are going to be communications-oriented, and even run a satellite system, why object to using the ASCII code that *all* communications is based on?

6. Conclusions

ASCII was well-designed, and is flexible to adapt to usage that may take any turn of development. There is no reason, with presently available technology, to use any other encoded alphabet. As a single standard, it enables private data to become public whenever that is desirable (i.e., privacy may be protected or maintained in ways other than unintelligibility). It is the alphabet of all communications networks, of all minicomputers, and of some larger computers such as the NCR Century series.

After twelve years it is still healthy, and when IBM puts the seal of approval on it (viz. virtual memory and APL) it will be the undisputed universal interchange medium and linguist.

7. References

1. ANSI X3.4 - 1974, "American national standard code for information interchange"
2. ISO 646 - 1973, "6 and 7 bit coded character sets for information processing interchange"
3. R.W.Bemer, "A view of the history of the ISO code", Honeywell Comput. J., 6, No. 4, 274-286 (1972)
4. Federal Information Processing Standard Publication 7, "Implementation of the code for information interchange and related media standards"
5. B.O.Evans, letter of 1973 June 28 to R.R.Johnson, Chairman Federal Information Processing Standards Task Group 12.
6. ANSI X3.41 - 1974, "Code extension techniques for use with the 7-bit coded character set of (ANS) code for information interchange"
7. ISO 2022 - 1973, Code extension procedures for ISO 7 bit code"
8. R.W.Bemer, "Escape - a proposal for character code compatibility", Commun. ACM 3, No. 2, 71-72 (1960 Feb)
9. R.W.Bemer, Working Paper, "Registry - A new feature for ISO recommendations", 1970 May 28
10. ISO 2375 - 1974, "Procedures for the registration of escape sequences"
11. USSR State Standard, GOST 13052-67, "Computers and data transmission equipment, alpha-numeric codes", 1967 July 10
12. Japanese Industrial Standard Code for Information Interchange, "JISCII", C6220-1969
13. T.O.Holtey and E.H.Ciamons, "Soft-copy controls", Honeywell Comput. J. 7, No. 4, 267-269 (1973)
14. ANSI X3L2/1499, "Proposed American national standard - control characters for an 8-bit code", 1975 Oct 10
15. ISO/TC46/4/1 N54, "The set of specific bibliographic control characters", 1975-08-12
16. ISO Third Draft Proposal 1672, "Hardware representation of ALGOL basic symbols in the ISO 6 and 7 bit coded character sets"

COMNET

COMPUTER NETWORK CORPORATION

November 12, 1975

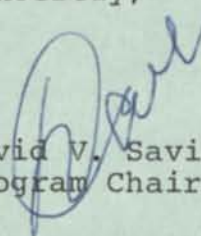
Mr. Robert W. Bemer, Sr.
Consulting Engineer
Honeywell Information Systems
Room B106
P.O. Box 6000
Phoenix, Arizona 85005

Dear Bob:

Your delineation of ASCII, today and forever, was excellent. All these pieces are beginning to fit together. You should feel proud of the footprints you've been making in these sands of time. ASCII forever.

My sincere thanks for your contribution to the success of the Second National Symposium on the Management of Data Elements in Information Processing.

Sincerely,



David V. Savidge
Program Chairman

DVS/em

ALTERNATIVES TO THE GENERALIZED DBMS

R W Bemer

Honeywell Information Systems

ABSTRACT

The abstract of this report is intended to provide a brief, general overview of the main ideas and concepts of the report. It is not intended to be a substitute for the full report, which contains the detailed analysis and discussion of the various alternatives and their relative merits.

The report is intended to provide a brief, general overview of the main ideas and concepts of the report. It is not intended to be a substitute for the full report, which contains the detailed analysis and discussion of the various alternatives and their relative merits.

The report is intended to provide a brief, general overview of the main ideas and concepts of the report. It is not intended to be a substitute for the full report, which contains the detailed analysis and discussion of the various alternatives and their relative merits.

1. INTRODUCTION

The purpose of this report is to provide a brief, general overview of the main ideas and concepts of the report. It is not intended to be a substitute for the full report, which contains the detailed analysis and discussion of the various alternatives and their relative merits.

The purpose of this report is to provide a brief, general overview of the main ideas and concepts of the report. It is not intended to be a substitute for the full report, which contains the detailed analysis and discussion of the various alternatives and their relative merits.

The purpose of this report is to provide a brief, general overview of the main ideas and concepts of the report. It is not intended to be a substitute for the full report, which contains the detailed analysis and discussion of the various alternatives and their relative merits.

82



R W Bemer obtained a degree in Mathematics at Albion College, Michigan. His distinguished computing career began with RAND Corporation, and he has since obtained experience with a variety of firms including IBM, Univac, Bull and General Electric. In 1970 he joined Honeywell Informations Systems as Staff Consultant and Editor of the Honeywell Computer Journal. In 1974 he became Senior Consulting Engineer with Honeywell, and in recent years he has headed the development within the company of an interactive enquiry system based on the relational model. Mr Bemer is a member of the British Computer Society and is author of numerous publications.

ABSTRACT

The opportunity exists to improve data base methods and usage, especially for smaller data bases and more direct usage by non-experts of the office world. They will be content-addressable and relational, to avoid expensive programming and restructuring as requirements shift.

There must be tools to ensure retrievability of clean data, and to expand the utility and value of the data in return for the cost of recording and storage.

For more stable data bases, where display use exceeds change, interactive searching may give way to pre-planned substructures created on a single scan through the data base, thence to indexed microform.

DATA BASE TYPES

I begin with a reminder that the data base milieu shows considerable variety, and that no single method is likely to be optimum, or even standard. This should have become particularly evident in the last three years. The advent of microprocessors and quite large store chips has created a drastic imbalance in hardware/software cost ratios, much in favour of hardware. Software must follow if it can; perhaps parallel searching and content-addressable or associative stores will become practical twenty years after introduction of the principles.

With the greater freedom in method these hardware advances permit, we can consider adapting more flexibly to the variety of data bases, particularly as their usage extends downwards to smaller businesses. Some data bases are constructed for a single purpose. An example might be the software and documentation for a computer system. Some are constructed for single purposes and found adaptable to others. For example, the public telephone directory, which one might usually construct as a linear file, might be structured differently if mail were addressed by telephone number (an idea from M Longworth of HIS). Others are multi-purpose from the start - for ordering, transactions, reservations, records, etc. And now people are considering distributed data bases as well as central. I wish them luck - it will be a very difficult problem.

DATA BASE USAGE

Not enough attention has been given to the usage purposes of data bases, particularly on the human side. We are accustomed to automatic update and change, to specific enquiries, and to automatic end products such as addressing, order fulfilment, etc. But businesses (and governments) need more utility from their data bases. They can profit from browsing - to discern relationships, groupings, patterns, and trends - all of which can indicate better decisions.

A data base is a collection of information whose elements have relationships with each other ranging from null to weak to strong. The tendency for the last ten years has been to utilize a single physical representation of any element, even though it may have memberships in several sets within the data base. This was/is to minimize search time and storage requirements.

The change/display ratio (C/D) should be a more critical factor in choice of data base usage methods. For applications with high C/D ratios the conventional integrated data store, highly structured with multiple pointers, may be quite appropriate. But many business applications fall in a much lower C/D range. The need to manipulate the data for display demands flexibility to adapt to new (and unforeseen) usage. The executive can change his mind faster than a data base can be restructured, so the thing to do may be to avoid building relations into the structure. Display requirements sometimes demand that the data base exist physically in report or table form. Practical experience in usage often causes a change in manipulation of the data base, or in what is required from it. And usage is not necessarily interactive query. Many banks have found it satisfactory to capture a COM picture of the data base at 24-hour intervals, reproduce for all users, and query from microform. This is usually less expensive than interactive enquiry at a terminal, often by as much as 3:1.

Data bases with low C/D can be stabilized in some time interval, and subjected to a single search to extract simultaneously the pre-planned substructures that will be of interest. Certainly the use of parallel microprocessors would speed this method. The substructures then go to COM, with perhaps a hardcopy index. One would look at the index to see if there was a substructure (table), for example, that showed the freshwater bodies in the US, ordered by state, with their mean volumes from 1966 to 1975. It might exist. Then one might ask for the same, replacing volumes by amounts of each type of fish stocked. If it did not exist, it would be put on the list for extraction the next time through.

DATA BASE UTILITY

There are methods to ensure getting one's money's worth from a data base. Display is an essential. To illustrate, in profiling all of the people in my consulting data base on a major of 'government', I noticed that I had failed to enter a minor of 'software' for Grace Hopper. The association might not be so automatic for another user of the same data base. Taking these subpictures allows adding further information to augment later utility.

Excessive search time lowers the utility of a data base. Looking, with no prior knowledge of what I would find, at a portion of a data base from a government agency, I discovered that it was all in capital letters, with extra spaces between. Surmising that the data was entered with punch cards, I used the text editor to recursively replace

occurrences of two spaces with one space. The altered file required just half as much storage! Even if pointers yielded no additional search time, operational costs would still be greater due to the excess store.

'Dirty' data also lowers the utility of a data base. I then did a concordance, and discovered the remarkable coincidence that a single oil company had three high executives named (respectively) Wohlgemuth, Wohlegemuth, and Wolgemuth! I do a concordance periodically on my consultant data base, to ensure uniqueness of the descriptors used. Thus I do not find all those for 'mgmt' whilst omitting those for 'management'.

These are but a few of the data base tools necessary to ensure clean and retrievable data, to expand the utility and value of the data, and to achieve additional types and forms of usage without major cost and delay. These tools are really necessary to maximize return on the investment of recording and storing the data base.

CONSTRUCTION OF CONTENT-ADDRESSABLE DATA BASES

I have designed and constructed several data bases for the internal business usage of our company. (See the four examples summarized at the end of this paper.) They are constructed and manipulated solely by programs written in a new text processing language. For this reason they are content-addressable, loosely structured, with relational capability as desired. All data entry is performed interactively, at a terminal, usually in response to prompting questions. I have noted a UK product, 'Olaf', that may be somewhat similar in principle and operation.

A text processing language

The text processing language used was specified by Eric Clamons and Richard Keys. Keys programmed the processor, which includes the text editor in the responsibility of Edward Parker. This specific credit is given here in anticipation that this language, or variants, will impact the business world as much or more than APL did the scientific and computation world. My own role was to commission it and advise.

It is a true programming language. Facilities exist for called programs, variables, loops, arithmetic, comparisons and matching, scanning for patterns, conditional statements, interactive input/output, etc. Every application thus far programmed with this language has shown cost and time reductions of 5:1 or better, in comparison with any other programming language used (thus APL is excluded from this comparison, as we have no data).

The diagnostic facilities are simple and elegant. One might call it a 'bug-a-minute' language. Programs are usually readable enough to be self-documenting. As illustration of this point, I once had a program that was not yet ready. A man in Los Angeles accessed the program in Phoenix, having never seen specifications for the language. From just the reading of my program he deduced the specs, and fixed a mistake that I had made! This understandability will be a very important factor in acceptance of the language by office workers.

It is a very good language for conditions, often found in business work, where requirements come up fast and go fast. Many applications never get on computers because they cannot be programmed and checked out fast enough to do any good while the problem is

still there. At present the processor is an interpreter, and may not be optimum for some long problems. But if an application persists, one can either use a compiler or re-do the slow parts. The original program may be used until the new program runs properly, and again to check out changes in methods before applying them to the new program.

The next section describes a feature of the text editor portion, my own contribution, that is vital to the processing of content-addressable data bases because it enables both 'content' and 'content type' to be stored together as an entity.

Prefixes (semantic line labels)

Our text editor, which is a subset of the text processing language, used to work in two search modes:

- 'find string' locates the first occurrence of the search argument string. In effect, the lines are formed up into a continuous string punctuated by Carriage Return characters (all files are created at terminals in timesharing). This is obviously the slowest possible search method.
- 'find' locates the first line *beginning* with the search argument string. Internal markings of line length, invisible to the user, permit jumping immediately to the beginning of the next line if the string is not yet found. This method is much faster.

The significance of Carriage Return serving a dual purpose, marking the logical end of one line and the logical beginning of the next, led me to wonder if the functions could not be split. They can, if one assigns a mode whereby a certain symbol is defined as the prefix delimiter. We now have two more search modes:

- 'find prefix' locates the first line in which the search argument string occurs *before* the prefix symbol is encountered. In effect, the search window opens with the CR, and closes with the prefix symbol.
- 'find text' operates correspondingly, except that the search window does not open until the prefix symbol is encountered, and closes with the CR.

The result is not only a search method of intermediate speed. We are now provided with a semantic label facility for each line. That label can contain the several properties of that line. Some possible usages of labels follow:

- This line (program statement) is a part of module 124.
- This line (program statement) participates in versions 3 and 4 of this software module. It will not disappear for audit purposes, but simply will not be selected for any versions other than 3 and 4.
- This line (program statement) is written in COBOL 68, level 3, and has to do with input/output control for a terminal.
- This line (text) participates in the third revision of a legislative bill.
- This line is a type 2 heading. When it goes to the line printer, underscore it.

When it is displayed on a video terminal, highlight it. When it goes to photocomposition, set it in 10-point Helvetica bold italic. When building the index for this document, it will be included.

- This line contains subtotal amounts for an entity which has a certain other prefix property.

A REPORT DATA BASE

A good example of a content-addressable, relational data base was provided by an internal business systems study. It combines elements of prefix notation, display requirements, and interactive entry. Without being specific as to its function, the building of this useful data base can be described.

Each of many groups is asked to describe its work on a printed form, in a structure of several majors, each having intermediates with minors. Each group has an assigned number, the key to all operation. Data services operates with a base program which, when called, asks for the number, whether help is needed (instructions are embedded in the program), and what task is to be performed. A service program is called for each specific task. The first task asks for each major in turn, then the intermediates, and then the minors. The resultant report, via line printer, is properly indented for structure and decimally numbered, e g:

```
2.3      Intermediate - (title)
  2.3.1   Minor - (title)
```

The prefix for each line contains, in this case, 't1' for major, 't1a' for continuation lines of the major, 't2' for intermediate, etc. These reports are returned for approval or markup to correct and make a new report. When approved, data services calls two tasks that build forms for the line printer, with entry headings and instructions for filling in. The data lines from the first report are embedded. Filled-in forms are returned to data services.

This new data is entered by calling another task, which interactively prompts for an item of data for each element of the minors. Any minor may be chosen for entry; it is displayed for verification before proceeding.

```
Minor number?      2.3.4
                  2.3.4 Minor - (title)
OK? (CR), or new minor?
Supervisory dollars? 15
Supervisory manyears? .6
Exempt dollars?    28
```

If the minor already has data entered, the program asks if this is a replacement. If so, the previous entry lines are deleted for entry of the updated data. Again, each class of data is prefixed appropriately for later recognition. The new data is inserted in place following each minor.

Now a report of standard paper width is generated, with proper headings, rounded-off and corrected and aligned figures (they were all closely monitored on entry), and dollar total lines generated as required for minors, intermediates, majors, and grand total.

If the grand total does not reconcile, the new report is marked up and returned to data services for correction and resuming. When the grand total does reconcile, another task produces a summary report with totals after the lines of the original report. An additional report form is generated for the line printer, embedding this data with new column headings for distribution and allocation in a different dimension.

There are other activities after this point. Selected content for all numbers is adjoined for automatic comparison and cross-referencing. But the essentials of the process are:

- Users work with paper forms just as they always have (the notion of management sitting at a terminal hasn't worked out as some had thought).
- Data entry is interactive and monitored for quality. Classifications are appended automatically. Converted keypunch operators think that this is great fun; the keypunch never talked to them.
- The data base is physically structured as a continuous report. A listing of the raw data will correspond to the actual report produced.
- The data base is actually a collection of named files, automatically created with different names, fetched, and resaved as needed. Subsetting of the data base is achieved by subsetting one file, appending the next, and subsetting it.
- The subsetting is achieved by specifying a logical relationship for the prefix contents of the lines to be extracted.

Our findings of getting applications operational 5 to 10 times faster with the text processing language, than with conventional programming languages, were well substantiated with this application. Six man-weeks of programming (myself) were required, three weeks of which were concurrent with the first part of the production process. The resultant data base was about 100 000 lines, or 3 megacharacters. The capability to adapt to requirements changes was much appreciated by management, who had never seen this sort of response time before.

SOME PRACTICAL RELATIONAL DATA BASES

Example 1: Telephone directory

Structure

One line of entry for each person. Contains name, telephone number, department, mail station, room number, and building -- all separated by Horizontal Tab characters.

Entry and change

Use text editor to find the line beginning with the proper string. Delete, replace, insert as required.

Output (display)

'alphatel' yields line printer listing, paged and numbered, with window control to at least three entries in a new section. Runs as many copies as requested. Weekly or daily update for telephone service, guards, receptionists.

'phototel' yields photocomposed equivalent, usually quarterly. For all personnel.

'mailtel' yields line printer listing of mail station and name only, blocked by alphabetical character of mail station. For the pigeonholes, usually monthly.

'bytel' yields printer listing of number and name. On demand, for the telephone service.

'secttel' yields printer listing of all data, but segregated by section. Each part given to secretary for that section.

'depttel' yields the same for the departments.

'strintel' gives a subset listing for any desired string. For example, what was the last name of that girl named Anne in Software Engineering?

Remarks

Originally a punch card file, all upper case. Converted to dual case semi-automatically (i e, the program wasn't perfect yet). Ordered by mail station, sent in groups to each mail station for verification. The average error rate of the original directory, which relied upon each person to send in his own changes, was about 20%. Among the entries removed were five deceased, a woman who had left six years ago to have a baby, and a spurious Fred Fortran in Manufacturing.

Room number and building were additions to the original file. The directory is now much appreciated, a matter of pride to all. The quarterly re-issuance period is shortened for major changes, so it is usually quite current. This is a major timesaver, eliminating queries to the old number, calling to find where someone is located, etc.

The telephone data base serves as a basis for some other programming pleasures.

call now

Your last name is? Bemer

Good morning, Robert

Today is Saturday, 1976 September 11

- the 6th day of Fiscal Week 37,

- the 255th day of the year, and

it is now 11:34

Here is the first part of 'today', which is called by 'now', after 'familiar', which gets your name from the telephone directory:

```

nove x=*date[3 mo=x]2 da=x[3
tord=9 week=7 month=/January /
if mo:eq:2 tord=40 month=/February /
if mo:eq:3 tord=69 month=/March /
if mo:eq:4 tord=100 month=/April /
if mo:eq:5 tord=130 month=/May /
...
if mo:eq:12 tord=344 month=/December /
tord=tord+da fw=tord/week sub=fw*7
tfd=tord-sub fd=tfd+1 ord=tord-9
suffix=/th/ day=/Sunday/
if fd:eq:1 suffix=/st/ day=/Monday/
if fd:eq:2 suffix=/nd/ day=/Tuesday/
if fd:eq:3 suffix=/rd/ day=/Wednesday/
if fd:eq:4 day=/Thursday/
if fd:eq:5 day=/Friday/
if fd:eq:6 day=/Saturday/
fd=fd,suffix tens=/O/,ord tens=tens[-2 units=tens[1
suffix=/th/
if units:eq:1 suffix=/st/
if units:eq:2 suffix=/nd/
if units:eq:3 suffix=/rd/

```

Almost every major business and governmental body has its own internal telephone directory and its own computer, but most of the examples I have seen have been poor. Many are done with just punch cards. The system described here is in fact easier and cheaper than with punch cards, and gives a variety of useful by-products.

Example 2: Vehicle directory

Structure

Integral with the telephone directory. A line for each vehicle registered follows the main entry. It begins with 'v', a unique identifier for vehicle, and contains name, telephone number, license number, year, make, model, existence of CB radio, and existence of alarm.

Entry

An interactive program at a video terminal asks for last name. If more than one exist, it asks for first name to pinpoint. Ditto for middle initials. The file line is located, and name and telephone number are entered automatically. All other data is entered in response to questions. Input checked for validity.

Change

Find the proper line. Delete, replace, or insert.

Output (display)

'byname' yields an alphabetical listing via line printer. Used for verification of entry.

'byplate' yields a listing ordered on last three digits of plate number. Used by patrol in parking lots. They will call the owner if lights are left on, if alarm goes off, or other emergency exists.

'byname' or 'byplate' also asks for subsets. For example, all owners of CB radios, when a special mailing was made to announce a visit by Phoenix police to lecture on security and the emergency radio network.

Remarks

Originated when the Vice President's secretary had her CB radio stolen. When it was suggested that vehicle registry could be incorporated in the telephone directory, it became known that Plant Security had wanted it for six years, but had been told that it would cost \$10 000 to program by conventional methods. A memo format was agreed at 14:00. It said that entry was voluntary, and had spaces to enter vehicle data. A text processing program was invoked to put two messages side-by-side, three to a printer page - six in all. Mail station and name were abstracted from the directory, and dealt into the boilerplate messages, then into the mailroom in station order without covers.

At 10:00 the next morning, as the Plant Security Director was telling his superior that we might do something, sometime, the morning mail arrived with one of 4000 messages, personally addressed to the superior, together with his copy of my memo to the Vice President telling how it all had been done. And at 0.4 cents per memo!

The telephone data base will serve for electronic mail inplant and to other HIS locations. Mail for each person can be accumulated and go out to COM for filing once a month. Saves on filing cabinets and space, and makes for easier retrieval.

Other extensions are under consideration, to expand the telephone data base into a more comprehensive personnel system. Skills inventory, job openings, health records, and residential data are some that come to mind.

Lest anyone become upset, remember that HIS computers have the best security features available, and we apply them. These data bases have very limited access, particularly because they are content-addressable!

Example 3: Company organization chart

Structure

A line of entry per individual. Name is enclosed in curly braces, and is prefixed by number signs (#) in the amount to indicate the level of reporting. It is followed by the job title, and the line ends with a vertical bar. Formatting controls and organizational component titles are embedded in the file.

Change

Straight text editing. Find the right name. Delete, insert, or modify.

Output (display)

'orgrun' yields the requested number of line printer listings. Done on demand, for those that must have a current structure. Printing is done 2-up, and each printer page

is folded.

'runindex' provides interactive entry of grid numbers (e g, 5F, as for maps). Name and grid number only are then extracted, and ordered by name for an index to the chart.

'setorg' yields the photocomposed output, with proper indentation to indicate reporting level, together with the index in multiple columns. Names are boldfaced, titles are standardweight. Produced quarterly.

Remarks

This chart used to be done by the box over boxes method, with name inside. It required an Avco plotter, a programmer for it, and a 5 cm ring notebook for each recipient of the chart. Net cost, with paper, was about \$40 000 per year. Now it requires 4 pages, printed two sides and stapled, and is issued quarterly instead of yearly! Photocomposition is via Videocomp, and is done in 7-point Helvetica Medium, a font that is excellent for both paper and microfiche, being void of serifs.

This organization chart is companywide, but it is obvious that lower levels of organization can be inserted. Various pictures of the organization can then be taken, controlled by range of levels and by subgroup.

Example 4: Consulting file

Structure

Five lines for each individual entry, as follows:

- 1 Last name, first name, and middle (initial).
- 2 Telephone number (home number and telex, if any).
- 3 ZipCode.
- 4 The full address, the start of each new line indicated by a vertical bar.
- 5 Descriptors.

Each line contains a left curly brace, the number shown above, the content, the number, and a right curly brace. All entries are stored alphabetically in 24 different files, e g, 'addr/a', 'addr/opq'.

Change

Straight text editing. Call the proper file, find the name, verify it. Modify, enter, or delete as required.

Output (display)

'runaddr' formats lines 2, 4, and 5 to make hard copy for filing and modification purposes. (It cannot be on-line all the time for this purpose.) Usually done one initial letter at a time. Printing is 2-up; that is, when the printer page is folded in half and punched for a ring binder, each side has a page printed on it.

'finddesc' prints a 4-up ordered listing of all descriptors used in the entire data base; duplicates are eliminated. Used to ensure uniqueness of descriptors, which usually just come from the mind as one remembers or meets a person.

Examples: hdwe, stwe, univ, govt, history, code, COLOSSUS, security, ISO, ECMA, BSI, and - yes - Infotech.

'who' (or 'newwho' when much updating has been done) asks for a combination of descriptors, and gives the names of all the qualifiers. The program first asks if telephone number and/or mailing label are wanted.

'profile' does rather more. For any major descriptor, it will print the names of all qualifiers, together with the 'hits' for any of several minors that may be identified. The minors are numbered sequentially, and the hits indicated by printing that number in its proper column across the page. This gives the subpictures, or cross-sections, of the data base to scan to ensure completeness.

Remarks

There are presently about 1200 names in this data base - people that I know in and about the computer business. I regret deeply the need to remove the names Gill and Strachey this last year.

I use it in the sense (for privacy) of the exempt Christmas List, except that instead of sending cards I act as one gatekeeper for people who have open positions, need speakers or conference organizers, or wish to contact the experts on certain subjects. Size is presently at 0.5+ megacharacters, for fellow HIS employees are not listed. Carl Hammer of Univac (when last I knew) kept his list on file cards, and it was 3 to 4 times the size of mine.

It is a most helpful memory extender, one that I would now hate to be without. Because it contains what Hedley Voysey calls 'heavily-typed data', it is very browsable. I cannot spend the effort to keep it fully up to date, but even so it is vastly more useful than, for example, the list of ex-IBM employees that is now sold.

HONEYWELL

SOFTWARE PRODUCTIVITY

SYMPOSIUM

APRIL 26-28, 1977

MINNEAPOLIS, MINNESOTA

TOWARD THE COMPLETE SOFTWARE FACTORY

R. W. Bemer, HIS PHX

Abstract

One does not build a successful factory with just tools and environment. The workers must be trained, and the assembly methodology (e.g., drawings, parts lists, inventory) must be in place. A review of these elements as they have been developed in Phoenix in the past 18 months includes:

- Methodology -- use of the prefix notation in TEX has opened the way to classical assembly methods -- assembly drawings, bills of material, and selective assembly via dash number parts. The great flexibility achieved has opened the question of whether 1) all programmers should be trained in structured programming methods, or whether 2) piece parts and exemplary coding should be created by the very ablest in the craft, for menu selection and use by the less able.
- Control -- all software units are complete in a single file, prefixed to differentiate specs, schedules, size, working code, test code, etc. This aids both the maker and management in inspecting status, alteration, audit trail, etc.
- Training -- a new structuring of training methods is proposed. First the printed documentation is graded for display methods ranging from top graphic quality manuals for terminal usage, through COM-generated hard copy, through COM for viewing. Next, certain procedures are removed from documentation by replacement with working programs that ask interactively for input parameters (e.g., for a system library edit). Finally, a substantial portion of documentation is eliminated in favor of CAL (Computer-Assisted Learning). This has two substantial advantages -- (1) it is interactive, private, and available at any time, and (2) it has controlled currency and correctness that is difficult to achieve with printed and distributed material.

Brief Biography

A.B., Mathematics, Albion College, 1940
 Certif., Aero. Engg., Curtiss-Wright
 Inst. of Tech., 1941

Consultant, HIS, Phoenix, AZ

Created first load-and-go compiler, PRINT I for IBM 705. Wrote first timesharing paper, 1957 March. Profiled New Yorker Magazine, 1957 January. Developed FORTRANSIT, first programming language to run on both decimal and binary computers. Developed COMTRAN, one of three inputs to COBOL. Developed XTRAN, predecessor to ALGOL. Only non-Britisher to address Annual Meeting, British Computer Soc. (Fellow, 1968). Originated what became ASCII and ISO Code. Invented ESCape mechanism. ACM Council - 6 years. Wrote original scope, work program for ANSI X3, ISO TC97. Chairman ISO TC97/SC5, Common Programming Languages. U.S. rep., IFIP Vocabulary. Funded SIMULA. Chiefly responsible for decision to build Univac 1108. Program Chairman, ACM70; editor, "Computers and Crisis". Program Chmn., First NCC. Chmn., ANSI SPARC, Text Processing. 73 published papers.

TOWARD THE COMPLETE SOFTWARE FACTORY

R. W. Bemer, HIS PHX

METHODOLOGY

Who is the more productive?

Joe makes his automobile from scratch. He hogs out the axles on a lathe from a square bar; he makes the frame by cutting, heating, bending, and welding steel beams; for the engine he builds a wooden model, makes a plaster cast, and pours hot metal. And so on, and so on...

Sam goes to a junkyard. He finds a pair of axles that will do if he cuts 8 mm off either end, a Chevy frame that he can weld his special clips to, and an engine that can have the cylinder diameter bored to a larger size.

If you can agree that Sam is the more productive, because he can complete his car in 3 months with 1/100th the costs that Joe incurs, then I will ask why is software built by the Joes? Structured programming is certainly a valuable technique for increasing productivity. But should it be stressed for all programmers as the best method, or just for those building standard software parts?

Q: How can I know what software piece parts are useful for my assigned project?

A: There are parts catalogs for autos. And inventories of available parts. There are parts manufacturers that subcontract to auto manufacturers.

Q: Yes -- but you can see what they look like. Software is pretty invisible. Even reading a listing of a program often doesn't tell you too much.

A: You're talking about the average software of today. Auto parts manufacturers aren't average, run-of-the-mill -- they're experts with special factories and tools. Imagine 10% of today's programmers in the special parts business (and they'd be the best programmers). To sell the parts they'd have to explain each module thoroughly, wouldn't they?

Q: But that's when I'm buying a new part. How do I know what's in my existing software that I can reuse?

A: The way librarian's of all kinds do it. With keywords and searches. Get on a system equipped with TEX and TEXTLIB, and "call textlib/inventory". You'll be asked for your keywords, and then told what tool programs in TEXTLIB got hits. For example, ask for the keywords "line", "length", and "equal". You'll get the most hits for the program "fixed!". So "call textlib/fixed!explain" to get narrative about what "fixed!" does. If you think you can use all or a part of it, list it to find out.

Q: Suppose I modify a piece part for a certain use of mine. Will everyone having the same need as mine have to do the same, or is there some way my version can be recorded?

A: The very biggest 10-4! That's the secret of the whole software piece part game. You make a drawing of a numbered part. It is itself perhaps made up of other numbered parts, put together in specified ways, with specified processes (e.g., welding) applied. The drawing contains a parts list for making *all* different versions of that part. The "dash numbers" indicate which version.

1236-06 is part number 1236, version 06

1236-06 might differ from 1236-04 in that there are two holes in the angle brace instead of one, and the top right corner is scarfed off at 45 degrees, 32 mm from the corner.

Prefixes

The key to applying standard manufacturing practices to software manufacture is the prefix notation, which provides semantic labels. (See Appendix A).

The semantics in a label are constrained as to content, because it must reflect a property of the text line. Typeless attributes such as originator and date of origination can apply to all types of text, but it would be improper to say that a Fortran statement should be underscored, or that a footnote was executable. (See Appendix B for prefix design considerations).

Note that frequently occurring combinations can be replaced by other keys with fewer characters, recursively even unto binary keys (but excluding the binary representation of the prefix character).

With the .include or .mark we are ready to start the specification process. A "drawing" file is called, and the command .include (or .mark) given. The editor finds all occurrences, e.g.:

```
.include turrets
```

and replaces each with the actual content of the referenced file -- in this case, "turrets". At end-of-file, the pointer is backed up to repeat the process in case any of the included files also contain .include instructions. What a one-instruction way to build an operating system!

But suppose there are several possible forms of "turrets", as defined by prefixes. We must specify which, so we say:

.include turrets [V6]

This gets us all lines from "turrets" that contain "V6" in the prefix (the prefixes are deleted before inclusion).

CONTROL

A major flaw and expense in most existing software production is the lack of reusable modules (piece parts, standard subassemblies, or what you will). Most new software is built independently from other software because the function is not perceived to be related or similar. The 400 software had 20 different "get/put" routines -- one SYSOUT for FORTRAN, and one for everything else.

It is of paramount importance that the existence of previously-designed and manufactured parts be known to the designer. This is a matter of management control (aided by inventory programs -- see, next section on training). In software production this has been difficult heretofore due to invisibility and lack of such control (like reading of code by supervision) and not having a standard parts book available to everyone.

The pieces to make the various dash-numbered parts are kept in a single file, which is named with the part number (see Appendix C). The sublists of parts are indicated in the prefixes; each program statement either does or does not participate in the subassembly so indicated.

Suppose a manager wishes to know the status of the modules under his supervision. He would do a "call status". The program "status" would ask which major module, and get the list of submodules it comprises. Using this to call each file in turn, it would add up the number of lines of working code included, check the projected number, and use the start/finish dates and build curve to estimate where the module stood with respect to schedule. "status" would then provide a summary listing. It's not precise in any way, but does give a good indication of whether trouble spots exist.

This program is not built yet, because it waits upon the methodology being in actual use. It might take a day to write.

TRAINING

We realize that much of our documentation methods are obsolete. And expensive, particularly in paper consumed. Photocomposition reduces paper usage greatly, and the methods have been available for a decade. Yet the first official photocomposed manual from Phoenix is just about to appear. It will be very difficult to read, because a substantial knowledge of graphics arts must be acquired to complement the photocomposition capability.

An experimental photocomposed manual (L66 TimeSharing) was produced nearly two years ago. It was readable, and copies went fast (partly because they were free). The 600 copies came to about 100 per computer system, compared to 2 to 4 per system for our customers. It was in A6 size, with integral microfiche edition. But we were unsuccessful in getting the format adopted -- the only people that were enthused were those that actually used computers. Management, particularly Marketing, couldn't see it. A compromise is now proposed in an A5 (or 5.5 x 8.5) size manual. This will still fit in a jacket pocket, or in front of a terminal.

Documentation Display

We have now realized that all computer system documentation doesn't have to be nice hardcopy printed manuals. Some we use everyday, and for these we prefer hard copy; some perhaps once a year, and it's foolish to endure the cost of hard copy when COM or printer listings will suffice. Some changes slowly, and is suitable for hard copy; some changes very often, and demands either COM or getting it from a single authoritative source file in the mother computer. Most source listings are now run on COM, as is the Library Catalog furnished to customers.

Creation and Entry

Level 66 methods differ substantially from Level 68. For L68, the working programmers build the documentation without intermediaries. For L66 we have three levels -- programmer, tech writer, and keyboard/formatter. The latter two do not use the computer they are writing about.

It is surely more desirable to have the programmer create the documentation. He must, to some extent, to specify what his program is going to do. And going further to the usage aspects cannot but improve his design. Formatting seems to be a major deterrent. An AEP student has written a program in TEX that does specifications semiautomatically and interactively. The sections are standard; each must have some verbiage supplied. The terminal issues entry requests -- fill in for this, then that. The formatting (heading, numbering, indentation, etc.) is supplied gratis and without pain.

Replacing Documentation by Programs

There are many thick manuals for specific procedures. A outstanding example is the System Library Editor. The manual must be written and updated, taught to our internal personnel, Field Engineering, and customers. Then two out of three tries will abort, and customer handholding is required. Manuals that are procedural can all be replaced by interactive query-answering programs.

Learning Tools

The method described here applies only to tools written in the TEX language, but it is generally applicable to other tools (command library) and to production software.

TEXLIB is a user-id for a depository containing:

- TEX - to move to system level
- ETEX - latest experimental version, to be used only by specific call
- TEXDOC - source for manual
- TEX.INFO - runoff manual
- Sundry tools, written in TEX
- Other tools, written in perhaps Fortran, such as front-door aprint and 2print
- INVENTORY
- LEARN

The prospective user is asked to

```
call texlib/explain
```

A message says that the programs in TEXLIB (on such a date) are the following, listed alphabetically. To know the details about any of these, type its name. Details appear. This cycles as long as interest continues. Explanations are in the actual files at "\$* \$l explain", except for some non-TEX-program files. The user can reach these independently by doing

```
call somefile!explain
```

at any time. When the user responds with only a Carriage Return, a message says that most TEX files have read permissions, so one can see both good practice and how they work.

No programs are to be entered into TEXLIB without a section headed "\$* \$l explain" following the working code. This should give the operational characteristics, possible inputs, choices, and outputs. Sometimes it is useful to give examples of situations where the program is particularly useful.

We intend to eventually follow this with "\$* \$l maintain", wherein one gives the essence of the program design that may not be visible in the code -- interactions, design choices, time dependencies, etc.

■ Inventory

If the user should ask for an explanation of "inventory", he may be curious to try it, and does a

```
call texlib/inventory
```

This program asks for as many keywords as the user wishes to supply. It then lists the TEXLIB programs that had hits, what hits, and how many hits. For files of interest he again does

```
call somefile!explain
```

If not just what he wanted, the source can be copied and modified, with local renaming. If he thinks the program can be more generalized to include his features, he contacts the author, whose name and phone number usually appear close to the beginning of the program. If the program has been called, a simple way is to type

```
out:author
```

This signing of programs is both good practice and a matter of social responsibility. If this doesn't work, he contacts R. W. Bemer, whose phone number appears liberally throughout the programs and explanations.

■ Learn

When the user enters

```
call texlib/learn
```

he is given a list of the features of the TEX language, and asked to enter the name of the one he is interested in. He gets a description of how it works, and then a line headed

```
ACTION!.....ACTION!...
```

He replies to the questions, follows instructions, sees the results, and reads a message "To do this you would have written .."

(the actual code to cause what he had done)

He may cycle several times on one feature, then go to another feature, or leave the "learn" program altogether. Regretfully we have not had enough support for all features of "learn" to be exercised at this time. Those that are written, however, demonstrate the power and human interface.

For someone that hasn't some understanding of a certain feature it's actually faster to learn this way than to read documentation in a manual. And by having to construct a working program to operate interactively in this way, the aspects of the feature are more likely to be covered completely than by writing verbiage.

This has two substantial advantages -- (1) it is interactive, private, and available at any time, and (2) it has controlled currency and correctness that is difficult to achieve with printed and distributed material.

Appendix A -- The Prefix Mode (Semantic Line Labels)

The 6000 text editor, a subset of TEX, used to work in only two search modes:

- "find string" locates the first occurrence of the search argument string. In effect, the lines are formed into a continuous string punctuated by Carriage Return characters (all files are created at terminals in timesharing). This is obviously the slowest possible search method.
- "find" locates the first line *beginning* with the search argument string. Internal markings of line length, invisible to the user, permit jumping immediately to the beginning of the next line if the string is not yet found. This method is much faster.

The dual-purpose Carriage Return marked the logical end of one line and logical beginning of the next. The functions can be split by assigning a mode whereby a certain symbol is defined as the prefix delimiter. We now have two more search modes:

- "find prefix" locates the first line in which the search argument string occurs *before* the prefix symbol is encountered. In effect, the search window opens with the CR, and closes with the prefix symbol.
- "find text" operates correspondingly, except that the search window does not open until the prefix symbol is encountered, and closes with the CR.

The result is not only a search method of intermediate speed. We are now provided with a semantic label facility for each line. That label can contain the several properties of that line. Some possible usages of labels are:

- This line (program statement) is a part of module 124.
- This line (program statement) participates in versions 3 and 4 of this software module. It will not disappear for audit purposes, but simply will not be selected for any versions other than 3 and 4.
- This line (text) participates in the third revision of a legislative bill.
- This line (program statement) is written in COBOL 68, level 3; it concerns input/output control for a terminal.
- This line is a type 2 heading. When it goes to the line printer, underscore it. When it is displayed on a video

terminal, highlight it. When it goes to photocomposition, set it in 10-point Helvetica bold italic. When building the index for this document, it will be included.

- This line contains subtotal amounts for an entity which has a certain other prefix property.

Some character, at user option, may be predefined as a "prefix delimiter". The current version of TEX must be put in this mode by the command:

```
-pref (any acceptable character)
```

This character may now be used to separate each line into two components -- prefix and text. The prefix consists of all characters to the left of the delimiter; the text consists of all characters to the right. Using a "p" to replace the "s" in string verbs limits their operation to the prefix only; using a "t" instead of "s" limits their operation to the text only. String verbs in their normal "s" form apply to the full line, even though the prefix mode is in force, and the lines contain the delimiter.

When using verbs with "p" or "t", the delimiter itself is unaccessible. It may be changed dynamically with another "pref" command, but do not use more than one delimiter character for any file. The choice of a delimiter character should be made with care that it does not need to, or will not, appear in either prefixes or text.

The prefix mode is designed for selective text processing. Lines may be printed, replaced, deleted, found, cut, copied, etc., selectively on the basis of a string existing in the prefix.

Prefixes may contain one or several identifiable strings. Their length is not restricted, and may be variable. A simple way to delete all prefixes from a file is:

```
-b befline;*/ */ b ds:*/ */(the delimiter);* b
```

After calling in a prefixed file to operate upon, the mode must be set for the delimiter it utilizes. Thus it is advisable to begin the file with a statement like:

```
This is a prefixed file. The delimiter is "%".
```

Appendix B -- Designing Prefix Sets

Principles

- 1 - The symbols must be left to right hierarchical, to permit selection of like sets, subsets, etc.
- 2 - Major symbol classes should have very little mnemonic meaning. The inevitable contention for assignment to the shortest symbols may be overcome by using more characters, but this taxes resources and operating time.
- 3 - The various classes of symbols must be unique and impossible to mistake by overlap, else the individual components of prefixes must be separated by a specific character (also used in the search or filtering process).
- 4 - Major classes are best distinguished by an alphabetic character; subclasses best by digits. Dewey decimal notation may be used.

- 5 - Other prefixes may be designed at the option of individual users. They may be mnemonic if required. If such prefixes fall into general use, so that program tools take them into consideration, then they should be replaced by standard prefixes.

What Flavors are Needed?

The following presents a first-cut, strawman classification. Not all classifications are applicable in all cases. However, some standard set must be agreed at the outset, and augmented in a controlled manner, so that software tooling will not fail due to variety.

A Proposed Classification

A	Preamble for this file module	M nnn	Submodule number, for group identification
0	Name	N 1	Processor Level 1
1	Structure	2	Processor Level 2
2	Other requisites to read file	3	Processor Level 3
3	Ownership	4	Processor Level 4
4	Subownerships	O	Operating Mode
5	Content	1	GRTS
6	Missing content	2	NPS
B ?	Specification, external	3	RNP
C	Specification, internal	4	HDSA
6	Existing modules to be used, with or without modification	P	Processor types
D ?	Scheduling information	1	EIS
E ?	Estimates (speed, size, storage)	2	non-EIS
F ?	Simulator program for the module	3	NSA
G ?	?	4	non-NSA
H ?	?	Q 0	Mainline loops within loops,
I ?	?	1	First sublevel subroutines, nesting,
J ?	?	2	Second sublevel subparagraphs, etc.
K ?	?	etc.	
L	Language class	R nnn	Revision level (named software release, edit levels, alters, etc.)
01	COBOL (all)	S nnn	System configuration
014	COBOL 74 only	T	Test statements/instructions
018	COBOL 68 only	01	Insertion position 1
02	Fortran	U	Statement function
03	BASIC	1	I/O
04	TEX	11	- terminals
05	APL	12	- disc
06	PL/I	13	- tape
07	Card-in Job Control	14	- printer
08	GMAP	15	- card reader
09	?	16	- card punch
10	Multics	17	- sorter
101	?	18	- OCR scanner
102	QED	2	Table handling
103	ted4	3	String handling, parsing
11	TRAC	4	Communications
12	APT	5	Database
13	2000	V ?	?
14	Level 64	W	Working statements/instructions
15	Xerox	1	Public, furnished
		2	Public, separately priced
		3	Inhouse, experimental
		4	Inhouse, libraries
		X ?	?
		Y ?	?
		Z ?	?

Thus the following line:

W1L01N3U11 (The statement itself)

is declared to be part of our basic software furnished to the customer without charge, the statement itself being written in COBOL, compiled by a module that is part of the Level 3 compiler, and having to do with I/O at a terminal.

Now we consider that control should be extended past the working program into specification, levels of documentation, simulation, and testing. Only thus can we achieve what manufacturing engineers call "reduction of diversity".

Consider a program that is a file of this structure:

prefix text

M16 This file is Module 16
 S These are the lines of specification
 S for what this program is to do.
 S When we use it for another purpose
 S (another dash number) we identify the
 S variable areas by saying that this
 S1 is the spec for dash number 1, and this
 S2 is the spec for dash number 2.
 S Then we go back to the common spec.
 sim This is coding that simulates the
 sim action of this module, usually
 sim before it is coded. It would
 sim simulate the expected elapsed time
 sim of execution, and assign proper
 sim values to variables and conditions
 sim that must be passed on to
 sim other modules.
 Sch This is the production schedule.
 Sch Start, finish, and production pattern.
 Sz This is the predicted module size.
 D This is documentation for manuals
 D of all kinds. Perhaps some of the
 D specification lines can also be
 D used, in which case we would have
 DS multiple-purpose lines like this.
 D1 This is the first level of nitty-
 D1 gritty detail, perhaps not appearing
 D1 in the manual, but useful for the
 D1 originating and validating programmers.
 D2 Perhaps this is a different level or
 D2 type of documentation, for training.
 W This is working program, the only part
 W of the file that will be compiled or
 W interpreted into operating software
 P for distribution.
 C Here are comments interspersed among
 P lines of working program.
 T These lines are also working program,
 T but only included for compilation when
 T desired to run in a diagnostic mode.
 T Even here we can be selective.
 T23 Lines so prefixed are assigned to only
 T23 a particular portion of the module.
 T At option, we can include all "T"
 T23 or specific "T23" only.
 T Don't forget that lines so prefixed
 T can serve to introduce correct values
 T as an override during a diagnostic run,
 T so that many successive parts of the
 T module may be tested without interruption
 T due to failure of previous code.

Appendix C -- Actual Prefix Usage

Some Practical Considerations

Q: Will the combined length of prefix and text be awkward or exceed line capacity?

A: Not if the file system is well-designed and constructed. It is also possible to find, in prefixes containing many elements, certain frequent groupings that may be replaced by a single symbol.

Q: Can the prefix elements be confused?

A: Yes, because they are found solely by string matching. Therefore use caution to ensure uniqueness.

Q: Is it difficult to read a file so intermingled, with prefix fields of variable length?

A: Possibly, for some persons. There are solutions in displaying alternate forms of the file after processing:

- Replace all prefix symbols by a Horizontal Tab and the prefix symbol. Then do a runoff.
- Use TEX to analyze both the length and qualities of the prefix field. Then spaces can be inserted appropriately to effect both alignment and indentation by classes.
- Subset the file, selecting for display only those lines with the characteristics you wish to examine.

Q: Isn't it tedious to keep entering the appropriate prefix and the prefix delimiting symbol when creating a file?

A: Yes, if you don't use the program "texlib/prefix", which does it for you automatically.

Q: Do I keep the prefixes after subsetting?

A: It depends. If you wish to execute or compile a program, no. To compare different versions, one can perform selected deletion of prefix. Thus the instruction common to both versions 4 and 6 could be displayed without prefix, whereas those unique to either version 4 or version 6 would still be prefixed by those indications.

Q: Would I still want to keep comments on the same line as the instructions?

A: Preferably not. Insert them between working instructions or subgroups of instructions. You will get better flexibility and more readability.

INTERFACE TM ASCII

MICROCOMPUTING FOR CONSUMER AND BUSINESS APPLICATIONS

VOLUME 3, ISSUE 5 MAY 1978 \$2.00

CANADA/MEXICO \$2.50 INTERNATIONAL \$3.50

84 85 86

Inside ASCII – Part I

Floppy ROMTM #4
in the IAPSTM Format



Author Index – 1975-1977



COVER STORY

This month's cover, supplied by Space Byte, represents the emergence of the microcomputer from the realm of an idea to a useful business tool.

The cover captures the growth of the microcomputer industry and represents the freedom it provides to the business world. With the use of microcomputer systems, the traditional methods of bookkeeping and paper world can be tossed away to disappear into the oblivion of today's and tomorrow's technology.

INTERFACE AGE™

MICROCOMPUTING FOR CONSUMER AND BUSINESS APPLICATIONS

BUSINESS FEATURES

A FULL FUNCTION MAILING LIST 56
by John M. Billing

HOW TO LOAD FLOPPY ROM™ NUMBER 4
AND NOTES ON IAPS™ — International ASCII Publication Standard ... 70
*by Dr. Alan R. Miller, Contributing Editor
W.W. Turner, Southwest Regional Editor
with additions by Bill Thames*

AN EVALUATION OF THE AM-100 COMPUTER SYSTEM 81
by James W. Kitzmiller, Kitzmiller Systems

CONSIDERATION FOR COMPUTER IMPLEMENTATION IN A
SMALL BUSINESS — PART I 88
THE SMALL BUSINESSMAN'S GUIDE TO HELP
AVOID THE PITFALLS OF COMPUTER
IMPLEMENTATION. *by Roger Williams*

GENERAL FEATURES

INSIDE ASCII — PART I OF 3 PARTS 96
THE WHATS, WHYS, AND THE HOW TO'S —
A HANDBOOK ON ASCII *by R.W. Bemer, The Father of ASCII*

RCA CDP 1802 COSMAC MICROPROCESSOR 104
by Joseph Pallas

A DEVELOPMENT-PROJECT REPORTING & CONTROL TECHNIQUE .. 110
by William Rosenbluth, IBM Federal Systems Division

CUMULATIVE AUTHOR INDEX TO INTERFACE AGE MAGAZINE
DECEMBER 1975 THROUGH DECEMBER 1977 135

HARDWARE FEATURES

AN INTERRUPT DRIVEN FLOPPY DISK CONTROLLER
FOR THE S-100 BUS 144
by Robert Wright and Chester P. Quinn

COMPUTER TUTORIAL — PART IV 156
MORE MEMORIES *by Roger Edelson, Hardware Editor*

SOFTWARE FEATURES

BASIC'S TOKEN OF GOOD FORTUNE 160
by Tim Ryan, University Data Systems

LOOK 166
by Kenyon Swartout

A MEMORY CATALOG PROGRAM 170
by Jim Baumgardt

Advertiser Index 176
Book Reviews 133
Calendar 24
Editor's Notebook 4
European Interface 47
FIFO Flea Market 175
From the Fountainhead ... 42
Jurisprudent Computerist . 38
Letters to the Editor 8
Micro-Market 172
Mind Revolution 32
New Products 121
SenseLine 52
Update 12
White Collar
Microcomputer 29

INTERFACE AGE Magazine, published monthly by McPheters, Wolfe & Jones, 16704 Marquardt Ave., Cerritos, CA 90701. Subscription rates: U.S. \$14.00, Canada/Mexico \$16.00, all other countries \$24.00. Make checks payable in U.S. funds drawn on a U.S. bank. Opinions expressed in by-lined articles do not necessarily reflect the opinion of this magazine or the publisher. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsement of the product or products by this magazine or the publisher.
INTERFACE AGE Magazine COPYRIGHT © 1978 by McPheters, Wolfe & Jones. ALL RIGHTS RESERVED. Material in this publication may not be reproduced in any form without permission. Requests for permission should be directed to Kathy Robertson, Rights and Permission, McPheters, Wolfe & Jones, 16704 Marquardt Ave., Cerritos, CA 90701. INTERFACE AGE Magazine is catalogued in the Library of Congress, Classification No. QA75.5.155. Membership in Audit Bureau of Circulations applied for.
POSTMASTER: Please send change of address form 3579 and undelivered copies to INTERFACE AGE Magazine, 16704 Marquardt Ave., Cerritos, CA 90701. Second-class postage paid at Artesia, California 90701 and at additional mailing offices.



INSIDE ASCII

By R. W. Bemer

The data alphabet called ASCII (Figure 1, page 98, and Reference 1), also has two other names—International Standard 646 (the ISO Code [Reference 2]) and Alphabet No. 5 of CCITT (the International Consultative Committee for Telephone and Telegraph). It is used throughout the world, incorporated in billions of dollars of equipment.

But is it used correctly and wisely? Not always. There are misinterpretations, and gaps in definition that permit nonstandard usage. This article (in three parts) will give you the background, peculiarities, preferred practices, and new developments for ASCII. You will find a lot of information not too generally known or realized; it should help in the correct and safe usage of ASCII. For additional help, you can reference the various national and international standards given in Table 1. Some other detailed articles are listed in References 3, 4 and 5.

	100	ECMA	ANSI	FIPS PUB	CSA	BS	AS	CCITT	JIS	GOST
Binary-coded Character Set	646	6	X3.4-1977 \$4.50	1	2243.4	4730	1776	V.3	66220	13052-07
Graphics for Control Characters	2047	17	X3.32-1973 \$3.50	36		4730				
Character Set for Nonprinting	9773 W119		X3.43-1974 \$5.75	35	2243.34.1					
Additional Controls Character Imaging	48		BSI X3.64							
4-bit Sets	963	14		15	2243.6	4731/1	1070			
Code Extension Techniques	2022	35	X3.41-1976 \$6.00	35	2243.35	4953				
Registration Procedures for Escape Sequences	2575									
8-bit Coded Character Set	811 4871	43	X3.2777/08							
Character Set for P x 9 Matrix Printers	42									
Keyboard	2550	23	X4.16-1971 \$3.75			4822/1	1922			
Character Sets for Programming Languages	9775 W450	33								

Legend

- ISO - International Standards Organization
- ECMA - European Computer Manufacturers Association
- ANSI - American National Standards Institute
- FIPS - Federal Information Processing Standard
- CSA - Canadian Standards Association
- BS - British Standard
- AS - Australian Standard
- CCITT - Consultative Committee International, Telephone & Telegraph
- JIS - Japanese Industrial Standard
- GOST - USSR Standard

Table 1.

STICKS 4-7

ASCII, as a 7-bit code, is usually represented in 8 columns of 16 positions. The row positions are 0000 through 1111, the low-order 4 bits, 0 through 15 in decimal. The columns are 000 through 111, the next higher 3 bits, 0 through 7 in decimal. For some reason, the developers of ASCII found it convenient to refer to these eight columns as "sticks." So shall we. Each position will be represented in this article by its usual decimal representation. For example, capital A is position 4/1. Figure 2 is a representation of ASCII that is more convenient to those working in octal, rather than hexadecimal, notation.

HIGH ORDER OCTAL DIGITS	00	02	04	06	10	12	14	16	LOW ORDER OCTAL DIGIT
	NUL	DLE	SP	0	@	P	^	p	0
	SOH	DC1	!	1	A	Q	a	q	1
	STX	DC2	"	2	B	R	b	r	2
	ETX	DC3	#	3	C	S	c	s	3
	EOT	DC4	\$	4	D	T	d	t	4
	ENQ	NAK	%	5	E	U	e	u	5
	ACK	SYN	&	6	F	V	f	v	6
	BEL	ETB	'	7	G	W	g	w	7

HIGH ORDER OCTAL DIGITS	01	03	05	07	11	13	15	17	LOW ORDER OCTAL DIGIT
	BS	CAN	(8	H	X	h	x	0
	HT	EM)	9	I	Y	i	y	1
	LF	SUB	*	:	J	Z	j	z	2
	VT	ESC	+	;	K	[k	{	3
	FF	FS	,	<	L	\	l		4
	CR	GS	-	=	M]	m	}	5
	SO	RS	.	>	N	^	n	~	6
	SI	US	/	?	O	_	o	DEL	7

Figure 2.

PART 1 OF 3 PARTS



The first positions of sticks 4 and 6 are respectively the "commercial at" and "accent grave." Then the upper and lower case Roman alphabets follow. This offset of one position is historical (from the United Kingdom), and of no importance as long as you remember that it is so.

Following the alphabet in both sticks 5 and 7 are three positions each that one must be very cautious about. In ASCII they are assigned as [, /, and] in stick 5 — {, |, and } in stick 7. But in the ISO Code and CCITT versions they are reserved for national usage. Table II gives the national use assignment for these positions. Surely you remember that the Scandinavian alphabet has 29 letters, not 26? My friend Orjar Heen in Oslo is very protective of these positions. He says "If you Americans want to sell computers and software abroad, don't use the ASCII characters for these positions in your software."

To be more precise, positions 5/11, 5/12, 5/13, 7/11, 7/12, and 7/13 (noted above) are called *primary* national usage positions. So is 4/0, where ASCII has the "commercial at." Honeywell, for example, uses the "at" in its timesharing systems for deleting the previous character upon entry. But this isn't too serious, because many nations also have the "at" in their primary sets.

Also in sticks 4-7 are three diacritical marks. They are accent grave (`) in 6/0, circumflex (^) in 5/14, and tilde (~) in 7/14. These are called *secondary* national usage positions. In some countries the tilde is a straight overline.

But it is the circumflex where we have a lot of confusion. Teletype first made it an "up arrow" in an earlier version of ASCII, to serve as an exponentiation symbol, primarily for BASIC. But that doesn't do very well, because the exponentiation for FORTRAN is a double asterisk! The FORTRAN version is preferable in France, certainly, because they use such words as crane, cote, cout, and so on.

A companion problem exists in position 5/15, with the underscore. The underscore is neither national nor diacritical; all countries use it just as underscore (and for typesetting it is a U.S. convention to indicate italics, but in Italy it means boldface, except when it is the last character in a line!). But Teletype's early version of ASCII used it as a "left arrow" — probably for an assignment symbol equivalent to := in ALGOL. The up and left arrow have been carried over from Teletype into many video terminals. Ask your terminal manufacturer to cease and desist and retrofit. It's not ASCII and will only cause trouble forever.

The last character in sticks 4-7 is the Delete, symbol DEL, in position 7/17. It was put here because the binary code is 1111111, which would be all punched holes in perforated (not always paper!) tape, and that is the only way to make sure that it cannot be misread as some other character. ASCII is a complete set; all positions are assigned to have meaning.

STICKS 2-3

These are usually called the sticks for digits and specials. Remember that they are the "digits" 0 to 9; not numbers, not numerals, not anything but digits! They are in 3/0 through 3/9 so that the low-order 4 bits are the representations for packed decimal. Originally we considered the possibility of a special 4-bit set for numerical applications (see the fifth entry in Table Ia), but it turned out that computer hardware became inexpensive enough to not deprive ourselves of the extra capabilities of the 7-bit and 8-bit sets.

	currency		1st 7 national			dia	dia	1st 7 national			dia	
	2/3	2/4	4/0	5/11	5/12	5/13	5/14	6/0	7/11	7/12	7/13	7/14
Netherlands—A			¶		^	~	^					
Australia												
Belgium—A	¶											
W. Germany—A		§										
US												
Japan					¥							
UK	£						^					
Italy—A	¶	¶					^					
Switzerland—A		¶										
France—A		¶										
USSR		¶			¥		^					
Netherlands—B					U				¶	¶	¶	¶
Belgium—B			¶		c				¶	¶	¶	¶
France—B	£	§	¶		c		^		¶	¶	¶	¶
Switzerland—B			¶		c				¶	¶	¶	¶
Italy—B	¶	§	¶		c		^		¶	¶	¶	¶
Switzerland—C					c				¶	¶	¶	¶
Hungary	¶	¶	¶	¶	O		^		¶	¶	¶	¶
Germany—B	£	§	¶	¶	O		^		¶	¶	¶	¶
Switzerland—D					O				¶	¶	¶	¶
Sweden	¶	¶	¶		O		^		¶	¶	¶	¶
Finland					A				¶	¶	¶	¶
Denmark					A				¶	¶	¶	¶
Norway					A				¶	¶	¶	¶
Spain					¶				¶	¶	¶	¶

Table 2.

		0000		0001		0010		0011		0100		0101		0110		0111		
		0		1		2		3		4		5		6		7		
COL	ROW	NUL		DLE		SP		0		NOTE 1		P		NOTE 1		p		
	0000	0	□	☐	☐	△	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
	0001	1	SOH	DC1	!	1	☐	☐	A	Q	a	q	☐	☐	☐	☐	☐	
	0010	2	STX	DC2	"	2	☐	☐	B	R	b	r	☐	☐	☐	☐	☐	
	0011	3	ETX	DC3	NOTE 1	3	☐	☐	C	S	c	s	☐	☐	☐	☐	☐	
	0100	4	EOT	DC4	NOTE 1	4	☐	☐	D	T	d	t	☐	☐	☐	☐	☐	
	0101	5	ENQ	NAK	%	5	☐	☐	E	U	e	u	☐	☐	☐	☐	☐	
	0110	6	ACK	SYN	&	6	☐	☐	F	V	f	v	☐	☐	☐	☐	☐	
	0111	7	BEL	ETB	'	7	☐	☐	G	W	g	w	☐	☐	☐	☐	☐	
	1000	8	BS	CAN	(8	☐	☐	H	X	h	x	☐	☐	☐	☐	☐	
	1001	9	HT	EM)	9	☐	☐	I	Y	i	y	☐	☐	☐	☐	☐	
	1010	10	LF	SUB	*	:	☐	☐	J	Z	j	z	☐	☐	☐	☐	☐	
	1011	11	VT	ESC	+	;	☐	☐	K	NOTE 1	k	NOTE 1	☐	☐	☐	☐	☐	
	1100	12	FF	FS	,	<	☐	☐	L	NOTE 1	l	NOTE 1	☐	☐	☐	☐	☐	
	1101	13	CR	GS	-	=	☐	☐	M	NOTE 1	m	NOTE 1	☐	☐	☐	☐	☐	
	1110	14	SO	RS	.	>	☐	☐	N	NOTE 1	n	NOTE 1	☐	☐	☐	☐	☐	
	1111	15	SI	US	/	?	☐	☐	O	—	o	DEL	☐	☐	☐	☐	☐	

Note 1
 These 12 positions are variable for national usage — 2 for currency, 7 primary national usage, and 3 secondary usage which are diacritical marks when preceded by BSP. The presently-known assignments are given in the table below.

Figure 1.

Position 2/0 is officially called "space." I don't and didn't like it, and would have preferred "blank." Which is why the IBM community often uses a lower case "bee" with a slash through the vertical as its symbol. From the Univac side, the space has the official symbol "delta."

Having mentioned packed decimal, where two digits go into each 8-bit group ("byte" to the American, "octet" to the French), a word of caution on the plus and minus signs — they are in stick 2, rather than stick 3 with the digits. But the low order 4 bits are distinct, and + should be used only as 1011, — only as 1101. I mention this because the nonstandard code EBCDIC permits multiple representations of + and — in packed decimal. And the ASCII representations are not even coincident with any of these, with obvious dangers!

Watch out for the "currency" positions, 2/3 and 2/4. They also have national variations. In ASCII they are customarily # and \$, but there are some things to be remembered:

- # is not "number sign" for many countries, most of which use "No." or "Nr." for that purpose. And when it is "number," it must precede the digits, not follow.
- # closely resembled the "sharp sign" in music.
- # is "pound sign" only for the U.S., the only major country still not using the metric system. To the rest, it's kilograms. For now, it's best to use the abbreviation "lb." in the U.S., not the #. In any case, both must follow the numeral.
- To the British, a "pound" has the symbol "£", which is why that is the symbol in position 2/3 for the UK. They get very irked when # is called a "pound" sign, especially in software manuals.
- The "dollar" is peculiar to the U.S., Canada, and some others. There are also francs, marks, escudos, pesos, lire, etc., etc. Which is why the ISO code uses the universal currency symbol in position 2/4. It's a circle with outside spikes at 45, 135, 225, and 315 degrees (⊘), called "scarab." Table II also shows these assignments for several countries.
- ECMA has provided a separate guideline for specifying international currencies. See the "Where to Get More Information" at the end of this article.

It's a tough problem, and will get worse when we get into expanded character sets for photocomposition and such. For now, all we can do is follow the ASCII standard, which says that # is a "number sign."

Only a few more peculiarities remain for sticks 2-3. An important one is in the double quote, position 2/2, and the single quote, position 2/7. That is, you may think it is a single quote, and even use it so, but it is really an "accent acute" for vowels. It slants from top right to bottom left, to complement "accent grave" in 6/0, which slants from top left to bottom right. Some terminal makers do not realize this pairing, and will have accent grave slanting correctly, but put accent acute as a single quote in the unstylized up and down method. My Terminet is one of those that is OK.

Don't forget that to the typesetter, in contrast to typewriters, both single and double quotes have two forms — opening and closing. In fact, the typesetter gets his double quotes by using two single quotes, of either form, because the quote uses very little space in variable space typesetting. Most terminals, either video or hardcopy, use constant spacing. So double and single quotes must be distinct for that reason.

The last variation is in position 2/6, the ampersand. There are many legitimate different symbols for the ampersand. Neither ASCII nor the ISO Code prescribe any particular one. But this leads us to the next topic — how to represent the ASCII characters in handprinted form, so that they may be input to computer systems.

HANDPRINTING FOR STICKS 2-7

The classical confusion for many years was between the digit zero and the letter "oh," but there are other possibilities for confusion. American Standard X3.45 specifies the handwritten character shapes shown in Figure 3.

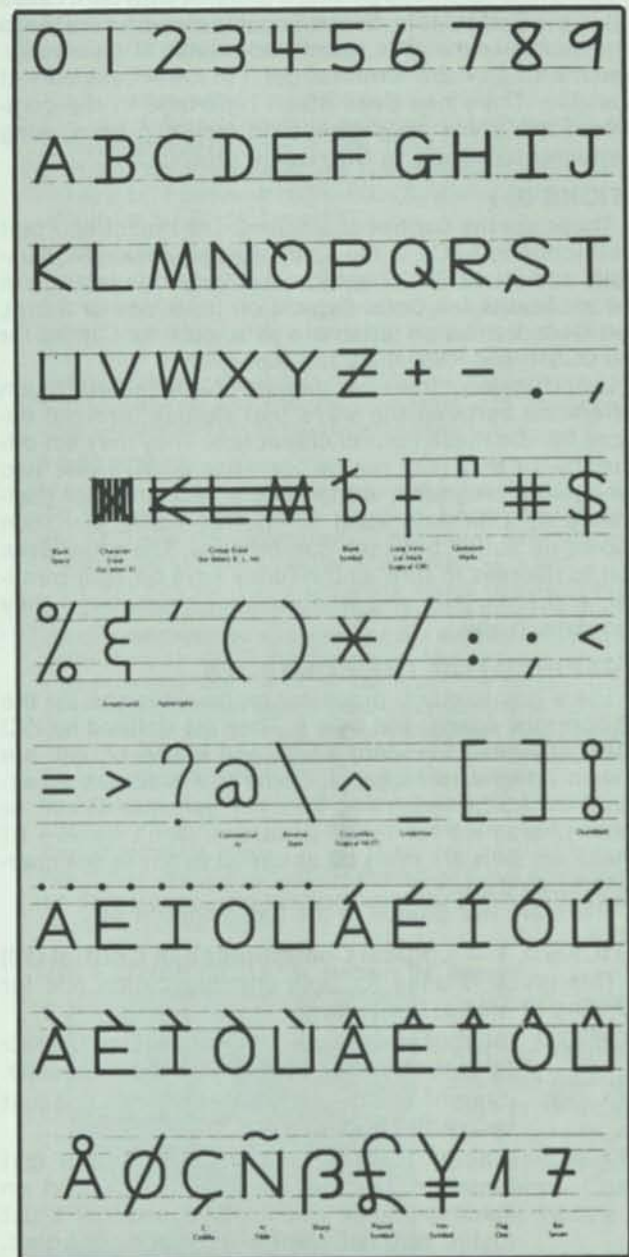


Figure 3.

This clears up a longstanding problem. The communications types, and the armed services, used to put a slash through the zero; somehow the IBM users got to putting the slash through the letter "oh" instead, confusing the Scandinavians greatly. Now it's neither (which helps), just a 180-degree rotation of the letter Q. The earlier German Standard DIN 66 002 prescribed the cursive loop in the upper right, as some may have learned in penmanship courses. It now permits the ANSI form as well.

UPPER AND LOWER CASE LETTERS

Many people are accustomed to using upper case only. This is a hangover from early line printers and limited sets (until the Stretch computer of IBM, characters were usually 6 bits in size). It would have been far better if they had all been lower case in those smaller sets. Putting it simply, would you buy a book to read if it were all

in upper case? Because lower case is much easier and faster to read, lower case should be the default case when one has only the one case. There is no reason why FORTRAN or BASIC processors cannot understand lower case variable names and verbs just as easily as they can understand upper case.

I always recommend getting a terminal with both cases if it is at all affordable. Second best is making sure that a single-case terminal is retrofittable later, if necessary. And if a single-case terminal, get it in lower case only, if possible. There has been much reportage in the computer trade press about eyestrain resulting from using computer terminals. Is the reason obvious?

STICKS 0, 1

These are the control characters. The most important distinction in ASCII is the split between sticks 0-1, Controls, and sticks 2-7, Graphics. We'll see this later on in the standards for Code Expansion (to 8 bits or more), and Code Extension (alternate sets, such as Cyrillic for the USSR, and Kata Kana for Japan).

Unfortunately, there is, despite the standard, much difference between the ways that various terminal devices handle these control characters. They may act differently, or they may not be operative at all. I have two very useful programs, written in the TEX language (Reference 6). One lists each symbol by name and then shows its action between parentheses. The other asks you to depress in turn all the funny keys on your terminal, and then tells you what control character(s) they generate, if any.

GRAPHICS FOR THE CONTROLS

There are standard graphical representations for the 32 controls, space, and delete. They are defined by ISO 2047, American Standard X3.32, and ECMA-17, and are shown integral to Figure 1. Some terminals are advertised as ASCII terminals, and yet generate Greek or other characters for these positions. Don't believe it! These symbols are every bit as useful as any Greek characters could be.

There are five groups in the basic control set.

STICKS 0, 1 — Logical Communication Control (10)

This group is used for both communication and for labeling of media. It includes:

- SOH (0/1) (Start of Heading) — used as the first character in the heading of an information message.
- STX (0/2) (Start of Text) — terminates the heading just before the text.
- ETX (0/3) (End of Text) — Last character in the text message. Unfortunately, it is generated on many terminals via Control-C, and that's just to the right of Control-X on the keyboard, which is commonly used to cancel a bad input line. And if you mis-key — ouch!
- EOT (0/4) (End of Transmission) — the last character in any transmission, and usually it turns your device off!
- ENQ (0/5) (Enquiry) — requests a response from a remote station, either an identification of that stations (Who are you?) or its status.
- ACK (0/6) (Acknowledge) — used by a receiver to reply "yes" to a sender.
- DLE (1/0) (Data Link Escape) — an Escape character, especially for communications, analogous to ESC (1/11). It signals the start of a character sequence that causes a shifting into another set of communication controls, whenever they are needed.
- NAK (1/5) (Negative Acknowledge) — used by a receiver to reply "no" to a sender.

SYN (1/6) (Synchronous Idle) — needed by synchronous transmission systems to get into, or stay in, synchronization when no other such signal is available to them.

ETB (1/7) (End of Transmission Block) — indicates the end of some division of data that the transmission system must make, unrelated to any division in the format of the logical data itself.

One lists each symbol by name and then shows its action between parentheses. The other asks you to depress in turn all the funny keys on your terminal . . .

STICKS 0, 1 — Physical Communication (4)

This group is used for communications. It includes:

- NUL (0/1) (Null) — the standard says that it is "used" to accomplish media fill or time fill" . . . "may be inserted into or removed from a stream of data without affecting the information content of that stream." And that's exactly what the standard also says about DELeTe (7/15), which it lists as a control character even though it is not in the control sticks! The only difference I can see between them is that on perforated tape you can make any character into a DELeTe, but none into a Null.
- CAN (1/8) (Cancel) — the receiver is to disregard the data received up to that point, starting from restart point that receiver and sender have agreed upon. It is common in timesharing for Cancel (often generated by a Control-X) to work on a line-at-a-time basis, to delete an unwanted string of entry characters, and effectively put one back to the position of re-entering the entire line. In this case, the agreement between sender and receiver is "back to the last CR." But there are many other ways that Cancel could be used, and for parallel as well as serial transmission.
- SUB (1/10) (Substitute) — a character that says probably we would have had another character in this position if we could have figured out what it was supposed to be! There are many reasons for such confusion — perhaps parity didn't check out. But it is better to put in a SUB to keep the field lengths and such correct. Moreover, note its symbol, a mirror image (not the Spanish inverted) question mark. If this is displayable, it will tell you definitively that the system doesn't know what it is, and you can make a good guess in many cases, particularly in word text.
- EM (1/9) (End of Medium) — defines the previous character as the last usable character on that medium, whether or not there is more recordable space on the medium.

STICKS 0, 1 — Device Control (11)

This group is used for control of devices such as terminals.

- HT (0/9) (Horizontal Tabulation) — the standard says that is "advances" the active position to the next predetermined character position on the same line." There are two ways this can work:

FS(File Separator — 1/12)
GS(Group Separator — 1/13)
RS (Record Separator — 1/14)
US(Unit Separator — 1/15)

FS is most inclusive, US the least inclusive. And we can consider the blank/space as the next lower order separator from these. Suppose we had a line of text like this:

```
(text1)US(text2)US(text3)RS(text4)US(text5)GS(text6)
```

On many terminals these delimiting control characters would not print, so we would see only a continuous stream. On others they might show as spaces. A TEX command to break the line at the record separator would be:

```
scan:line:*rs
```

The variable *left would contain "(text1) . . . (text3)". The variable *right would contain "(text4) . . . (text6)".

STICKS 0,1 — Changing Sets (3)

This group is used for moving to and from alternate graphic and control sets. This includes ESCape (1/11), Shift Out (0/14), and Shift In (0/15).

These basic control characters have permitted design of a quite marvelous structure for extension and expansion. It allows us to code and classify most of the world's graphic symbols for computer storage, interchange, and display. This big area will form most of Part III of this article.

IN THE NEXT INSTALLMENT

The ASCII Collating Sequence

ASCII and Programming Languages

ASCII and Media

Keyboards

ASCII and Display/Printing

Code Extension — Alternate Controls

Code Extension — Alternate Graphics

ASCII and Non-Latin Alphabets

Code Expansion — 8-bit ASCII

WHERE TO GET MORE INFORMATION

There are four sets of Information Processing Standards that may be of concern to you:

- ISO. Sold only through ANSI (American National Standards Institute), which has the franchise. That makes the prices high — much higher than in other countries.
- ANSI. These are American National Standards developed via the X3 and X4 committees, mostly. Prices still pretty high.
- ECMA (European Computer Manufacturers Association), 114 Rue du Rhone, 1204 Geneva, Switzerland). Free, and they have a lot more advanced standards than ISO and ANSI. But a modest donation would not be unwelcome.
- Your friendly U.S. Government, in the person of the Department of Commerce, National Bureau of Standards, Institute for Computer Sciences and Technology, in Gaithersburg, MD 20760. If by any chance you are employed by the U.S. Government, you get FIPS PUBS (Federal Information Processing Standards Publications) for cheap. Otherwise, see ANSI. (Refer to Tables 1a, 1b, and 1c). In many cases they are essentially reprints of the ANSI standards, for a fraction of the cost.

If you can't wait for the standards to be approved and published, catch them in progress. Ask CBEMA, the sponsor of ANSI X3, to put you on an observer list for the committee in your area of interest. The address is:

Robert Brown, Director of Standards
Computer & Business Equipment Manufacturers
Association
1828 L Street NW

Washington, D.C. 20036
(202) 466-2288 Telex 89 29042

REFERENCES

1. ANS X3.4-1977, available from the American National Standards Institute, 1430 Broadway, New York, NY 10018.
2. ISO 646, available from ANSI (Reference 1).
3. R.W. Bemer, "ASCII — the data alphabet that will endure," in *Management of data elements in information processing*, National Bureau of Standards, 1975 October, 17-22.
4. R.W. Bemer, "A view of the history of the ISO character code," *Honeywell Computer J.* 6, No. 4, 1972, 274-282.
5. E.H. Clamons, "Character codes: who needs them?," *Honeywell Computer J.* 5, No. 3, 1971, 143-146.
6. The TEX Subsystem of the Timesharing System, Series 60 Level 66, Honeywell Information Systems, 200 Smith Street, Waltham, MA 02154, Order DF72.

ACKNOWLEDGEMENTS

Thanks go to co-workers at Honeywell Information Systems: Eric Clamons for much background, insight, and experience gained from working for a long time as chairman of X3J2 — the committee charged with the development of ASCII. And to Pat Skelly, ACM representative on ANSI X3, for collecting all the various national and international standards documentation upon which many of the figures were based.

FOOTNOTES

¹For those curious about the reverse slash, it came from ALGOL 58. The reference language specified A and V as the symbols for AND and OR respectively. I put the reverse slash in so these could be made as 2-character groups — and

²You will still see many terminals where this vertical bar is broken in the middle. This resulted from a hassle with the PL/I people, who wanted to stylize the exclamation point (2/1) as a vertical bar for OR in that language. And of course that would make the graphics the same. The compromise (at horrendous cost in people time) was to break the real vertical bar in ASCII. But it turned out that the PL/I people didn't really need it, or else it gained no momentum, so the real vertical bar is back to normal in ASCII-1977. Let's fix those terminals.

³The Italians also have a different solution to hyphenation and right justification. It ignores the syllable structure and simply demands that if, when you get to the last position in the line, the current word is not yet completed, that last character shall be underscored, and the word continued without fuss on the next line. I rather like it.

THE FATHER OF ASCII, Robert W. Bemer



Robert "Bob" Bemer received his A.B. in Mathematics from Albion College in 1940, and a Certificate in Aeronautical Engineering from Curtiss-Wright Tech a year later.

His vast work experience includes employment with the major leaders of the aircraft and electronics industries—

most recently, as Senior Consulting Engineer with Honeywell Information Systems.

Highlights of his many accomplishments include: The discovery of polynomial telescoping (1954); creation of the PRINT 1 programming system for IBM (1956); development of FORTRANSIT; development of COMTRAN, one of the three major inputs to COBOL; development of XTRAN, predecessor to ALGOL (1958); was a major influence in the choice of the 8-bit character in IBM System 360 (1960); an influence in building the 1108 and 6000 systems; and editor of *Honeywell Computer Journal*.

He has an impressive list of over 71 publications to his credit. □

1. Right at the terminal, if it has the horizontal tab capability built in. Sometimes you can set the tab positions by using the terminal only; almost always the computer can be made to set the tabs on the terminal. Then when you hit HT during entry, or HT is read from the computer output, the printing or displaying (active) position will skip to the next tab setting.
2. By a formatting program in the computer, which must be given some indication of the tab setting positions in force at any particular point in the file. The program then simulates horizontal tab movement by filling the lines with spaces as needed to achieve the alignment.

VT (0/11) (Vertical Tabulation) — the standard says that it "advances the active position to the same character position on the next predetermined line." And if you agree with somebody else, it can be to the first position in that line instead. This is a very dangerous character to use. It cannot be used directly on any terminal that I know of. Even if it could, the implementation rules are not supplied unambiguously in the ASCII standard. And for use by a formatting program, one would have to predefine the number of lines to be skipped. That's pretty tough when you are inserting and deleting lines, as every programmer knows.

LF (0/10) (Line Feed) — like vertical tab, but just to the next line, which is clean enough. If receiver and sender agree (again as in vertical tab), it can be to the first position of the next line, in which case it is called New Line (NL). Some manufacturers implement this. I personally prefer having a separate Carriage Return and Line Feed. Both codes can be generated with a single keystroke, and they often are.

FF (0/12) (Form Feed) — again like vertical tab, to the same character position unless sender and receiver agree that it is to the first position in the new line, except that the tab is to a new line position that is related to a form of some size (those that fold 11 inches apart, for example). This control could run wild if your terminal or other display device is not equipped to handle it, so use it with caution in files.

CR (0/13) (Carriage Return) — moves the active position to the first position on the *same* line! Not like typewriters. They have effectively incorporated the New Line feature. But the non-advancing CR is better for terminals, even if it is misnamed. Neither video terminals nor ball and daisy wheel typewriters have carriages, so live with it.

BS (0/8) (Backspace) — Backspace is a very tricky character. On some terminals, such as video terminals, there is no key to generate Backspace for entry into the text stream or buffer. On many it can be created via Control-H. Even then, it may or may not be operative.

Backspace is meant for physical movement of the active position (which may or may not coincide with a cursor position, when such exists). Historically, it was included for hardcopy terminals and other hardcopy devices for some of these uses:

- Underscoring (underlining).
- Other forms of highlighting, such as bold.

For example, the sequence A BS A BS A would strike the A three times on a hardcopy device, and make it look boldface (such a sequence can also be translated to call a boldface font in photocomposition).

- Editing indications. For example, in legislative bill drafting to indicate the deleted or changed portion:

This is obsolete.

- Forming composite characters, e.g.:

Š ± ≠ † ‡ † } € (Hungarian forint)

- Forming accented letters, primarily for European languages. Examples:

Ä Å Ö (Scandinavian letters following Z)
Ñ ã à ô ü

Warning: Backspace is entirely different from a cursor movement on a video terminal! When the cursor is moved to a position where a character is already entered, succeeding entry in that position usually destroys the original character and replaces it with the new entry.

I personally haven't seen any video terminals with a true backspace. A former president of Infoton told me it could be done as an engineering special for about \$5,000 one-time cost.

Warning: There are three ways to create underscored text for hardcopy terminals:

1. The characters, that many backspaces, and that many underscores (or vice versa).
2. A character, BS, underscore, the next character, etc. This is called the canonical form, and is used quite commonly.
3. Underscore, BS, character, underscore, etc.

I have noticed a lot of difficulty moving back and forth between hardcopy (at my home) and video (in my office) terminals. One tends to underscore on the hardcopy terminal and forget that half of the pairs are going to be wiped out by the cursor on the video terminal. In the first two methods above, it's the text that gets wiped out, and it's hard to read on the fly. So if you plan to display a file on a video terminal, find another highlighting method, or use the third underscoring convention. Even that may give problems if done by embedding an underscoring command in the file you pass to a formatting program; most such programs put the underscore last instead of first.

BEL (0/7) (Bell) — sounds an audible signal to get the user's attention. Some terminals are not so equipped, but they should be. It's good human engineering. But please give me an adjustable volume control!

And then there are the four device controls for unspecified purposes, DC1, DC2, DC3, and DC4 — in positions 1/1 through 1/4. Different manufacturers treat these like a wild card in poker — they make them anything that they want. Doesn't lead to much compatibility, so beware.

STICKS 0, 1 — Field Separators (4)

This group is used for formatting and string processing. These are the separators in positions 1/12 to 1/15. I got the idea originally from the Word Mark in the IBM 1401, which used an extra bit in the low-order character in a field as a delimiter. ASCII uses special and separate characters to indicate a hierarchical structure. Originally I put in eight such characters, but only these four remain:

FS(File Separator	— 1/12)
GS(Group Separator	— 1/13)
RS (Record Separator	— 1/14)
US(Unit Separator	— 1/15)

FS is most inclusive, US the least inclusive. And we can consider the blank/space as the next lower order separator from these. Suppose we had a line of text like this:

```
(text1)US(text2)US(text3)RS(text4)US(text5)GS(text6)
```

On many terminals these delimiting control characters would not print, so we would see only a continuous stream. On others they might show as spaces. A TEX command to break the line at the record separator would be:

```
scan:line:*rs
```

The variable *left would contain "(text1) . . . (text3)". The variable *right would contain "(text4) . . . (text6)".

STICKS 0,1 — Changing Sets (3)

This group is used for moving to and from alternate graphic and control sets. This includes ESCape (1/11), Shift Out (0/14), and Shift In (0/15).

These basic control characters have permitted design of a quite marvelous structure for extension and expansion. It allows us to code and classify most of the world's graphic symbols for computer storage, interchange, and display. This big area will form most of Part III of this article.

IN THE NEXT INSTALLMENT

The ASCII Collating Sequence

ASCII and Programming Languages

ASCII and Media

Keyboards

ASCII and Display/Printing

Code Extension — Alternate Controls

Code Extension — Alternate Graphics

ASCII and Non-Latin Alphabets

Code Expansion — 8-bit ASCII

WHERE TO GET MORE INFORMATION

There are four sets of Information Processing Standards that may be of concern to you:

- ISO. Sold only through ANSI (American National Standards Institute), which has the franchise. That makes the prices high — much higher than in other countries.
- ANSI. These are American National Standards developed via the X3 and X4 committees, mostly. Prices still pretty high.
- ECMA (European Computer Manufacturers Association), 114 Rue du Rhone, 1204 Geneva, Switzerland). Free, and they have a lot more advanced standards than ISO and ANSI. But a modest donation would not be unwelcome.
- Your friendly U.S. Government, in the person of the Department of Commerce, National Bureau of Standards, Institute for Computer Sciences and Technology, in Gaithersburg, MD 20760. If by any chance you are employed by the U.S. Government, you get FIPS PUBS (Federal Information Processing Standards Publications) for cheap. Otherwise, see ANSI. (Refer to Tables 1a, 1b, and 1c). In many cases they are essentially reprints of the ANSI standards, for a fraction of the cost.

If you can't wait for the standards to be approved and published, catch them in progress. Ask CBEMA, the sponsor of ANSI X3, to put you on an observer list for the committee in your area of interest. The address is:

Robert Brown, Director of Standards
Computer & Business Equipment Manufacturers
Association
1828 L Street NW

Washington, D.C. 20036
(202) 466-2288 Telex 89 29042

REFERENCES

1. ANS X3.4-1977, available from the American National Standards Institute, 1430 Broadway, New York, NY 10018.
2. ISO 646, available from ANSI (Reference 1).
3. R.W. Bemer, "ASCII — the data alphabet that will endure," in *Management of data elements in information processing*, National Bureau of Standards, 1975 October, 17-22.
4. R.W. Bemer, "A view of the history of the ISO character code," *Honeywell Computer J.* 6, No. 4, 1972, 274-282.
5. E.H. Clamons, "Character codes: who needs them?," *Honeywell Computer J.* 5, No. 3, 1971, 143-146.
6. The TEX Subsystem of the Timesharing System, Series 60 Level 66, Honeywell Information Systems, 200 Smith Street, Waltham, MA 02154, Order DF72.

ACKNOWLEDGEMENTS

Thanks go to co-workers at Honeywell Information Systems: Eric Clamons for much background, insight, and experience gained from working for a long time as chairman of X3J2 — the committee charged with the development of ASCII. And to Pat Skelly, ACM representative on ANSI X3, for collecting all the various national and international standards documentation upon which many of the figures were based.

FOOTNOTES

¹For those curious about the reverse slash, it came from ALGOL 58. The reference language specified \wedge and \vee as the symbols for AND and OR respectively. I put the reverse slash in so these could be made as 2-character groups — and

²You will still see many terminals where this vertical bar is broken in the middle. This resulted from a hassle with the PL/I people, who wanted to stylize the exclamation point (2/1) as a vertical bar for OR in that language. And of course that would make the graphics the same. The compromise (at horrendous cost in people time) was to break the real vertical bar in ASCII. But it turned out that the PL/I people didn't really need it, or else it gained no momentum, so the real vertical bar is back to normal in ASCII-1977. Let's fix those terminals.

³The Italians also have a different solution to hyphenation and right justification. It ignores the syllable structure and simply demands that if, when you get to the last position in the line, the current word is not yet completed, that last character shall be underscored, and the word continued without fuss on the next line. I rather like it.

THE FATHER OF ASCII, Robert W. Bemer



Robert "Bob" Bemer received his A.B. in Mathematics from Albion College in 1940, and a Certificate in Aeronautical Engineering from Curtiss-Wright Tech a year later.

His vast work experience includes employment with the major leaders of the aircraft and electronics industries —

most recently, as Senior Consulting Engineer with Honeywell Information Systems.

Highlights of his many accomplishments include: The discovery of polynomial telescoping (1954); creation of the PRINT 1 programming system for IBM (1956); development of FORTRANSIT; development of COMTRAN, one of the three major inputs to COBOL; development of XTRAN, predecessor to ALGOL (1958); was a major influence in the choice of the 8-bit character in IBM System 360 (1960); an influence in building the 1108 and 6000 systems; and editor of *Honeywell Computer Journal*.

He has an impressive list of over 71 publications to his credit. □

INSIDE

PART 2 OF



The abstract aspects of ASCII were treated in Part 1. Now we come to some aspects of usage and implementation. Certainly one major use area is the ordering of files.

THE ASCII COLLATING SEQUENCE

To put items in some ordering, the entire precedence relationship for that ordering must be defined. Higher or lower, precedes or follows, or whatever. For single characters, this ordering relationship is called the "collating sequence".

The ASCII standard used to say that the collating sequence for both graphics and control characters is defined simply by their binary representations. Later it added a warning that this collating sequence "cannot be used in many specific applications that define their own sequence". What an understatement!

The 1977 version hedges and speaks all around the problem without making it clear. It's not all that difficult. Suppose you have two files, and you want to know how they differ and/or how they are the same. For this purpose, the implied collating sequence (straight binary comparison) is just fine. The two files will be in the same order, and can be matched.

Whether that straight binary ordering can be used for any other purpose is doubtful. It won't work for signed numbers.

Ordering Numerals

Take these four values: 22, 13, minus 6, and minus 31. If the sign is placed before the digits, ordering by the ASCII collating sequence yields:

+ 13
+ 22
- 06
- 31

This is obviously worthless. It's because ordering is decided left to right, and the minus sign has a binary value 2 higher than the plus sign. Or if the sign were to follow the numerals we would get:

06 -
13 +
22 +
31 -

because the complete decision is made in the leading digit. Again, a worthless sequence.

The way to achieve a proper ascending sequence is to separate the values into two groups, ordering those with plus signs in ascending sequence, and those with minus signs in descending sequence. Then put the plus group following the minus group. And vice versa for a total descending sequence. Notice that this works regardless of whether the sign precedes or follows the digits.

Ordering Alphabetic Fields

Alphabetic ordering is even more complex, particularly in handling both upper and lower case. Again the implied ASCII collating sequence can go wrong. People who have not studied the collating problem for data containing both upper and lower case are inclined to jump to wrong conclusions. I did myself, for the IBM Stretch computer in 1958, assigning the ascending binary sequence as AaBbCc. Using this for a telephone directory would give us the lefthand column. The straight binary sequence of ASCII would yield the righthand column, just slightly different:

De Carlo	De Carlo
De La Rue	De La Rue
De Long	De Long
DeLair	DeLaRue
DeLancey	DeLair
DeLaRue	DeLancey
Delancey	Delancey
de Carlo	de Carlo
de la Rue	de la Rue
deLancey	deLancey

Either version will get a lot of anguished subscribers!

In the simplest case, two alphabetic items must be compared with the case ignored. Only if they are *then* equal is case called into consideration to break the tie, and it is also applied successively left-to-right!

In short, the upper and lower case versions of a letter do not both get full graphic significance. Typing either "Y" or "y" will indicate a "yes" reply, but "N" will not. Because the case distinction is minor, comparisons must first be made on major distinctions, with the minor distinctions used only as tie-breakers. Accenting of let-

ASCII

3 PARTS

By R. W. Bemer



ters must also be considered minor, if accomplished via backspace, but this leads us into rules controlled by foreign governments, and won't be considered here.

Real life is more complicated than this. The ordering and sequencing of characters and words cannot always be accomplished by simple binary comparison of codes. There are constructions such as O'Reilly, l'Informatique (as data processing is called in French), and Smith-Jones — to say nothing of the Juniors, IIIs, Esq., FBCS (which I am), and so on.

Making an ASCII comparison, with the case as a minor, gives us:

De Carlo
de Carlo
De La Rue
de la Rue
De Long
DeLair
DeLancey
Delancey
deLancey
DeLaRue

Because we at first ignored case here, De Carlo and de Carlo have identical bit patterns. Tie-breaking is done by appending the binary pattern representing case, "0" for upper, "1" for lower. Specifically, 01001111 for De Carlo, 11001111 for de Carlo.

	D	E	C	A	R	L	O	
De Carlo	44	45	20	43	41	52	4C	4F (4F)
de Carlo	44	45	20	43	41	52	4C	4F (CF)

But even this method will not put "DeLaRue" and "De La Rue" in the same cluster. And surely this is desirable and even mandatory. It will require some special handling for spaces. The New York Telephone Company's document on this problem runs to several pages! They'd probably give you a copy upon request. You might need to know those rules before trying one of the toughest acts in data processing — putting last name first, or vice versa.

Using Controls in Ordering

There is one more aspect of ASCII useful to the order-

b.	0	0	0	0	1	1	1	1
b.	0	0	1	1	0	0	1	1
b.	0	1	0	1	0	1	0	1
	0	1	2	3	4	5	6	7
00000			SP	0		P	←	→
00011				1	A	Q	a	v
00102			"	2	B	R	i	p
00113				3	C	S	n	r
01004				4	D	T	L	~
01015				5	E	U	e	†
01106			&	6	F	V	x	u
01117			'	7	G	W	w	w
10008			(8	H	X	Δ	▷
10019)	9	I	Y	ι	∧
101010			*	:	J	Z	o	◁
101111			+	;	K	[÷	≅
110012			/	<	L	\	□	∩
110113			-	=	M]	≠	≥
111014			.	>	N	↑	∩	∩
111115			/	?	0	_	o	o

Figure 1.

b.	0	0	0	0	1	1	1	1		
b.	0	0	1	1	0	0	1	1		
b.	0	1	0	1	0	1	0	1		
	0	1	2	3	4	5	6	7		
b.	b.	b.	b.							
0	0	0	0	0		SP	0		P	
0	0	0	1	1		!	1	A	Q	
0	0	1	0	2		"	2	B	R	
0	0	1	1	3		#	3	C	S	
0	1	0	0	4		\$	4	D	T	
0	1	0	1	5		%	5	E	U	
0	1	1	0	6		&	6	F	V	
0	1	1	1	7		'	7	G	W	
1	0	0	0	8		(8	H	X	
1	0	0	1	9)	9	I	Y	
1	0	1	0	10		*	:	J	Z	
1	0	1	1	11		+	;	K		
1	1	0	0	12		,	<	L		
1	1	0	1	13		-	=	M		
1	1	1	0	14		.	>	N	^	
1	1	1	1	15		/	?	0	_	

Figure 2.

ing problem. In the days of punch cards, before computers, one often used several card files related by a key. A sorter (with pockets for the cards to drop into) might be used to select the cards for all redheaded females between 18 and 24 years of age. But these cards would have only the employee number and such characteristics on them. To get the name, address, and telephone number one might have to go to a second (related) deck of cards. So the first deck (the subset of interest) would be placed in the first hopper of a collator, and the deck with all names and phone numbers in the second hopper. Then a card would be fed from the first hopper, followed by successive cards from the second hopper, until a match was found on employee number. Obviously both decks had to be in the same ordering for this to work, and thus the term "collating sequence".

In effect, we were sticking the cards of the first deck upright just in front of the corresponding cards of the second. To do this with ASCII requires that we have characters that collate lower than the lowest graphic, the space (2/1). We do have them. The best to use are NUL, FS, GS, RS, and US. Put one of these after each search key, then put the two files together and order them as adjoined. Now those records having a search key with one of our five control characters appended will precede the corresponding record having an ASCII graphic following the key.

Note that the four information separators (FS, GS, RS, US) are designed to collate just behind Space, in that order. This contiguity means that they can be used as a hierarchy of spaces of different class.

Other Collating Features

ASCII was designed when there was substantial in-

b.	0	0	0	0	1	1	1	1		
b.	0	0	1	1	0	0	1	1		
b.	0	1	0	1	0	1	0	1		
	0	1	2	3	4	5	6	7		
b.	b.	b.	b.							
0	0	0	0	0		SP	0		P	
0	0	0	1	1		!	1	A	Q	
0	0	1	0	2		"	2	B	R	
0	0	1	1	3		#	3	C	S	
0	1	0	0	4		\$	4	D	T	
0	1	0	1	5		%	5	E	U	
0	1	1	0	6		&	6	F	V	
0	1	1	1	7		'	7	G	W	
1	0	0	0	8		(8	H	X	
1	0	0	1	9)	9	I	Y	
1	0	1	0	10		*	:	J	Z	
1	0	1	1	11		+	;	K		
1	1	0	0	12		,	<	L		
1	1	0	1	13		-	=	M		
1	1	1	0	14		.	>	N		
1	1	1	1	15		/		0		

Figure 3.

vestment in files already ordered on a Topsy-class IBM sequence, where the basic punctuation was low to the alphabet, but the digits were high to it. How then to accommodate this and still provide a 4-bit subset? My morning shower provided a solution (it still does!).

The 4-bit subset is formed of the first 10 graphics of stick 3 (the digit graphics) and the last 6 of stick 2. This jog was shown shaded in the early forms of ASCII, but has all but disappeared from memory now. It enables stick 3 (with the digits and new special graphics) to be ordered high to all the others via passive logic, thus overcoming opposition to the adoption of ASCII.

ASCII AND PROGRAMMING LANGUAGES

Standard ECMA-53 (1978 Jan), "Representation of Source Programs for Program Interchange," gives the subsets and/or modifications of ASCII as they are used for these five programming languages (Footnote 1):

NO. OF CHARACTERS USABLE

Language	Subset of ASCII	Other
APL	57	32
Minimal BASIC	60	0
COBOL	51	0
FORTRAN	49	0
PL/I	55	2

Figures 1 through 5 are the character sets for these languages as given in ECMA-53. They show the only characters permissible for use in source programs, except for non-numeric literals in COBOL
comment-entries in COBOL
comment lines in COBOL
character constants in FORTRAN
comments in FORTRAN

b. b. b. b.				b. 0 0 0 0 1 1 1 1									
b. 0 0 1 1				b. 0 0 1 1 0 0 1 1									
b. 0 1 0 1				b. 0 1 0 1 0 1 0 1									
				0 1 2 3 4 5 6 7									
0 0 0 0	0					SP	0			P			
0 0 0 1	1						1	A	Q				
0 0 1 0	2						2	B	R				
0 0 1 1	3						3	C	S				
0 1 0 0	4					\$	4	D	T				
0 1 0 1	5						5	E	U				
0 1 1 0	6						6	F	V				
0 1 1 1	7					'	7	G	W				
1 0 0 0	8					(8	H	X				
1 0 0 1	9)	9	I	Y				
1 0 1 0	10					*	:	J	Z				
1 0 1 1	11					+		K					
1 1 0 0	12					,		L					
1 1 0 1	13					-	=	M					
1 1 1 0	14					.		N					
1 1 1 1	15					/		O					

Figure 4.

b. b. b. b.				b. 0 0 0 0 1 1 1 1									
b. 0 0 1 1				b. 0 0 1 1 0 0 1 1									
b. 0 1 0 1				b. 0 1 0 1 0 1 0 1									
				0 1 2 3 4 5 6 7									
0 0 0 0	0							̣	0		P		
0 0 0 1	1								1	A	Q		
0 0 1 0	2							2	B	R			
0 0 1 1	3							3	C	S			
0 1 0 0	4					\$	4	D	T				
0 1 0 1	5					%	5	E	U				
0 1 1 0	6					&	6	F	V				
0 1 1 1	7					'	7	G	W				
1 0 0 0	8					(8	H	X				
1 0 0 1	9)	9	I	Y				
1 0 1 0	10					*	:	J	Z				
1 0 1 1	11					+	;	K					
1 1 0 0	12					,	<	L					
1 1 0 1	13					-	=	M					
1 1 1 0	14					.	>	N	ˆ				
1 1 1 1	15					/		O	-				

Figure 5.

character-string-constants in PL/I
 comments " "
 For these purposes only, other ASCII characters may be used, providing there is agreement between the sender and receiver for any interchange of source programs.

The TEX language has gone farther than this general caution. There the specific characters have permanent names. For example, one could say:

```
linefeed = "
" (actual line feed inside the quotes)
if !eqs:linefeed ....
```

and it would be true, because "LF" is the permanent name of Line Feed. The control characters have names that are the letters from the ASCII chart, preceded by the asterisk to show that they are read-only variables with permanent content. TEX can in fact operate upon all 256 characters of ASCII in an 8-bit byte, all 512 in a 9-bit byte.

Specific Notes on the Figures

APL -Sticks 6 and 7 (ordinarily lower case alphabet) are replaced entirely except for the DELETE position.

-Space is nonprinting, although the symbol shown is SP.

-Ampersand (2/6) is not used for writing source programs, except as the last character of a line if that line is to be continued on the next line.

PL/I -In position 2/1, the exclamation point is replaced by a vertical bar for OR.

-In position 5/14, the circumflex is replaced by the symbol shown, for NOT.

-If you have to use your terminal for both PL/I

and some other programming language, forget that foolishness. You can get by with the exclamation point as OR, and the circumflex as NOT. The important point in source program interchange is to have the encoded representations of the characters exchanged correctly.

(all) -Although the character BLANK (space) is shown as the flagged lower case "b" in the FORTRAN and PL/I sets, there is no printing graphic to indicate it. For all practical purposes, it is really the Space of ASCII (2/0).

-Four of these five languages (not APL) have the "\$" shown in 2/4. When the International Reference Version of the code is used, this becomes the universal currency symbol, which is also acceptable.

-Minimal BASIC uses "#", which is the International Reference Version symbol. The national symbols, such as the English pound sign, are also acceptable.

ASCII AND MEDIA

ASCII and Punch Cards

Reading and punching equipment for punch cards, being very mechanical, is so expensive that microcomputer people are unlikely to use them. So you might ask why we bother here with the representation of ASCII on this medium? I can think of at least three reasons:

- A scientist at the U.S. National Bureau of Standards said once that if punch cards were on the way out, it was the only product he ever saw dying on an upward usage curve. Thus they are likely to be around for a long

	ISO	ECMA	ANSI	FIPS PUB	CSA	BS	AS	CCITT	JIS	GOST
Hollerith Punched Card Code	1679 2021	44	X3.26-1970 \$4.25	14	Z243.14 .36	4636/3 /4	1063			
Track Assignment - 25.4 mm Perf. Tape	1113	10	X3.6-1965 \$3.00	2	Z243.8	3880/3	1062		C6221	
Track Assignment - 12.7 mm Mag Tape 200 cpi NRZI 9-track	1862	5	X3.14-1973 \$3.25			3968	1008			
Track Assignment - 12.7 mm Mag Tape 800 cpi NRZI 9-track	962 1863	12	X3.22-1973 \$3.75	3-1		4503/1	1009		C6222	
Track Assignment - 12.7 mm Mag Tape 1600 cpi PE 9-track	3788	36	X3.39-1973 \$3.75	25		4503/2				
Track Assignment - 12.7 mm Mag Tape 6250 cpi GCR 9-track	DP 5652		X3.54-1976 \$5.25	50						
Labeling & File Structure - 12.7 mm MT	1001	13	X3.27-1977 (unpriced)		Z243.7	4732	1068			
Track Assignment - Magtape Cassette 3.81 mm, 32 bpmm	3275 3407	34	X3.48-1977 \$5.75	51		5079/1				
Labeling & File Struct. - 3.81 Magtape Cassette	DIS 4341	41								
Track Assignment - 6.35 mm Cartridge Tape 64 bpmm PE	DIS 4057	46	X3.56-1977 \$4.24							

Table 1a. Standards for ASCII on Physical Media.

	ISO	ECMA	ANSI	FIPS PUB	CSA	BS	AS	CCITT	JIS	GOST
Bit Sequencing in Serial Transmission			X3.15-1976 \$3.00	16-1				V.4 X.4		
Char. Structure & Parity Sense - Serial-by-Bit			X3.16-1976 \$3.50	17-1				V.4 X.4		
Char. Structure & Parity Sense - Parallel-by-Bit			X3.25-1976 \$3.50	18-1				V.4 X.4		
Procedures for Using Commun. Control Chars.	1745	16	X3.28-1976 \$10.50		Z243.13	4505/1	1484/1			
Message Heading Formats	1745		X3.57-1977 \$5.25							
Advanced Data Commun. Control Procedures			BSR X3.66							

Table 1b. Standards for ASCII in Communications.

				b ₈																
				b ₇																
				b ₆																
				b ₅																
					00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
b ₄	b ₃	b ₂	b ₁	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
0	0	0	0	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
0	0	0	1	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15		
0	0	1	0	02	03	04	05	06	07	08	09	10	11	12	13	14	15			
0	0	1	1	03	04	05	06	07	08	09	10	11	12	13	14	15				
0	1	0	0	04	05	06	07	08	09	10	11	12	13	14	15					
0	1	0	1	05	06	07	08	09	10	11	12	13	14	15						
0	1	1	0	06	07	08	09	10	11	12	13	14	15							
0	1	1	1	07	08	09	10	11	12	13	14	15								
1	0	0	0	08	09	10	11	12	13	14	15									
1	0	0	1	09	10	11	12	13	14	15										
1	0	1	0	10	11	12	13	14	15											
1	0	1	1	11	12	13	14	15												
1	1	0	0	12	13	14	15													
1	1	0	1	13	14	15														
1	1	1	0	14	15															
1	1	1	1	15																

Figure 6.

	12	11	0	12	11	10	12	11	10	12	11	10	12	11	10	
	02/06	02/13	03/00	02/00	07/11	07/12	07/13	11/10	10/08	11/01	11/09	06/00	12/03	12/10	13/01	13/08
1	04/01	04/10	02/15	03/01	06/01	06/10	07/14	13/09	00/01	01/01	08/01	09/01	10/00	10/09	09/15	11/11
2	04/02	04/11	05/03	03/02	06/02	06/11	07/03	13/10	00/02	01/02	08/02	01/06	10/01	10/10	11/02	11/12
3	04/03	04/12	05/04	03/03	06/03	06/12	07/04	13/11	00/03	01/03	08/03	09/03	10/02	10/11	11/03	11/13
4	04/04	04/13	05/05	03/04	06/04	06/13	07/05	13/12	09/12	09/13	08/04	09/04	10/03	10/12	11/04	11/14
5	04/05	04/14	05/06	03/05	06/05	06/14	07/06	13/13	00/09	08/05	00/10	09/05	10/04	10/13	11/05	11/15
6	04/06	04/15	05/07	03/06	06/06	06/15	07/07	13/14	08/06	00/08	01/07	09/06	10/05	10/14	11/06	12/00
7	04/07	05/00	05/08	03/07	06/07	07/00	07/08	13/15	07/15	08/07	01/11	00/04	10/06	10/15	11/07	12/01
8	04/08	05/01	05/09	03/08	06/08	07/01	07/09	14/00	09/07	01/08	08/08	09/08	10/07	11/00	11/08	12/02
9	04/09	05/02	05/10	03/09	06/09	07/02	07/10	14/01	08/13	01/09	08/09	09/09	00/00	01/00	08/00	09/00
10	05/11	05/13	05/12	03/10	12/04	12/11	13/02	14/02	08/14	09/02	08/10	09/10	14/08	14/14	15/04	15/10
11	02/14	02/04	02/12	02/03	12/05	12/12	13/03	14/03	00/11	08/15	08/11	09/11	14/09	14/15	15/05	15/11
12	03/12	02/10	02/05	04/00	12/06	12/13	13/04	14/04	00/12	01/12	08/12	01/04	14/10	15/00	15/06	15/12
13	02/08	02/09	05/15	02/07	12/07	12/14	13/05	14/05	00/13	01/13	00/05	01/05	14/11	15/01	15/07	15/13
14	02/11	03/11	03/14	03/13	12/08	12/15	13/06	14/06	00/14	01/14	00/06	09/14	14/12	15/02	15/08	15/14
15	02/01	05/14	03/15	02/02	12/09	13/00	13/07	14/07	00/15	01/15	00/07	01/10	14/13	15/03	15/09	15/15

Figure 7.

time, and you may need to transfer some of those files to other media that you do use.

- There is some likelihood that microcomputers could be used in the reading and punching equipment itself, to make it less expensive.
- ASCII users are going to be confronted for a while yet with one of the several versions of IBM's EBCDIC, and the punch card assignments provide the only legitimate link for conversion of EBCDIC files to ASCII.

So Figure 6 defines the hole patterns for the binary encodings. And Figure 7 defines the encodings for the hole patterns. Don't worry about the inconsistency in the relationships. Nothing can be done about it now, because it started with Herman Hollerith's first U.S. Census machines in 1890. At first only digits and + and - signs were used. Then the code was expanded to the upper case alphabet. And other special characters for commercial use. When FORTRAN came along in 1964, it turned out that the limited capability of the subset of a 6-bit set would not permit the graphics needed for scientific work. For a long while there were dual graphic representations for several of the punch card code combinations, and this carried over into printer chains, and so on.

The only logic that the patterns follow is that they do or do not have a punch from among these six possibilities:

- 12-punch (top row)
- 11-punch (next to the top row)
- 0-punch
- 8-punch
- 9-punch (bottom row)
- a punch from among the digits 1 through 7

Including the no-punch-at-all combination (NUL), this gives 256 combinations, just right for the 8-bit code. Although ASCII was technically only a 7-bit code at the time this rule was formulated, it was felt necessary to plan ahead a little.

ASCII and Magnetic Tape

Figure 8 gives a compact representation of several relationships, among which is the assignment of ASCII bit pattern to 9-track magnetic tape. The jumbled assignment may remind you of the "firing order" for the cylinders of an automobile engine. In fact, we used to call it just that. It was intentional for increased reliability. As in so many cases, better technology has removed the need for peculiar design, but the assignments are unchangeable because of data file investment.

There is no parallelism in recording and reading on cassettes and cartridges. The ASCII bits are recorded serially in the track. Thus Figure 8 does not consider these media.

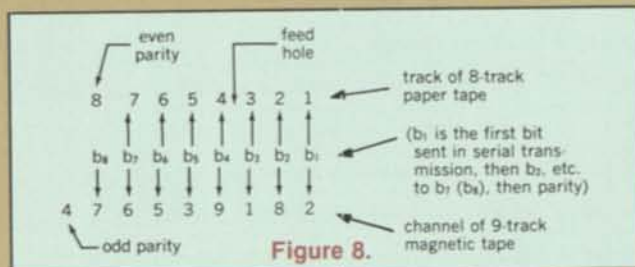


Figure 8.

ASCII and Communications

Not only is the topic of ASCII and communications a very complex and large dissertation for this article — it is also undergoing substantial rethinking, enlargement, and invention. You will have to follow on your own the workings of the CCITT, the various networking systems of the several large and many small manufacturers of computer systems, and the offerings of the common carriers — either on the local distribution system (via ATT) or direct distribution (via Satellite Business Systems).

Many of the existing standards are listed in Table 1b.

Many more are under development. Arguments are raging internationally on the merits of packet switching, byte protocols, value-added systems, open-working systems, tariffs, data movement across national borders, the X.25 protocol, etc., etc. ATT is offering a new service because they suddenly discovered data-undervoice (DUV). All I can tell you now is that it is all based upon ASCII, and the proposed protocols are all dependent upon the ASCII control characters in stick 0 and 1. It will take years for this to shake out, and for now all one can do is get on the CBEMA mailing list (see reference, Part I, INTERFACE AGE, May, 1978).

ASCII AND THE METRIC SYSTEM

The full ASCII graphic set (both cases) is sufficient to indicate all symbols and prefixes of the SI (International System of Units, the new metric system), with three exceptions. They are the Greek letters "omega" for "ohm", and "mu" for "micro", and the degree symbol for Celsius temperature. These three characters will be provided in 8-bit ASCII (see Part III, next month). Meanwhile, for these, and also for such equipment that has only a single case, there is a standard way of representing the SI units and prefixes. This is given in International Standard 2955, "Representations of SI Units and Other Units for Use in Systems with Limited Character Sets", and also in American Standard X3.50-1976.

To keep the record straight, let's first look at the characters used for the prefixes. They're shown in Table 2, which indicates multiples from 10 to the - 18 up to 10 to the + 18:

10 + i	i	10 - i
exa (E)	18	atto (a)
peta (P)	15	femto (f)
tera (T)	12	pico (p)
giga (G)	9	nano (n)
mega (M)	6	micro (μ)
kilo (k)	3	milli (m)
hecto (h)	2	centi (c)
deka (da)	1	deci (d)

Table 2. Metric Prefixes

Above 3 there are no powers except multiples of 3. This practice breeds better comprehension, like marking off three's in writing numbers of many digits. Also, as a memory convenience, all symbols are upper case for powers greater than + 3. And there are no conflicts with the symbols for the units of measurement.

Now, again for the record, here are the ASCII character(s) used as symbols for the units:

A	ampere	cd	candela
Bq	becquerel	d	day
C	coulomb	g	gram
°C	degree celsius	h	hour
F	farad	l	litre
Gy	gray	lm	lumen
H	henry	lx	lux
J	joule	μ	micro
K	kelvin	m	metre
N	newton	min	minute (time)
Ω	ohm	mol	mole
Pa	pascal	rad	radian
S	siemens	s	second (time)
T	tesla	sr	steradian
V	volt	t	tonne/metric ton/ megagram
W	watt		
Wb	weber		

Table 3. Metric Units

Table 3 shows the rules clearly. Units not named after people are all lower case, as shown in the righthand column (although I do know a Mr. Day). In the lefthand col-

umn are the units that are named after people. The names of the units are not capitalized at all, but the symbols begin with an upper case letter.

I said previously that there were no conflicts between unit and prefix symbols. But you've probably noticed "d" for both "day" and "deci", "h" for both "hour" and "hecto", "m" for both "metre" and "milli", and "T" for both "tesla" and "tera". OK. But there isn't any confusion in actual usage, because the prefix precedes the unit:

dd	is a deciday (2.4 hours)
hh	is a hectohour (100 hours)
hH	is a hectohenry (but don't ever use the term)
mm	is a millimetre
Mm	is a megametre (1/300 the speed of light)
TT	is a teratesla (Wow!)

I am not suggesting that the prefixes should be applied to other than the primary metric units (the second is the primary time unit; hour and day are not), even though the timesharing system I customarily use figures my time in millihours. But when you get accustomed, the prefixes are very valuable in other ways. For example, an American billion is a kilomillion, whereas the British billion is a megamillion! And my metric teaching program understands such things as kilofathoms.

The "space" character is also vital to correct SI usage. It must occur between values and units, like 123.6 mm, and 22 °C.

And don't forget another peculiarity of ASCII as an international alphabet: (1/14) is absolutely not defined as a "decimal point" (nor is it defined as "period", which in the United Kingdom is "full stop"). For most of the rest of the world, the comma (1/12) is the decimal marker, and the period is used to mark off threes. That's why the recommended practice for marking off threes is to use the space, not either comma or period. E.g., "1 234 567 mm".

To save you the bother of looking up the standards for use with limited character sets, here is the algorithm:

1. If you have ASCII with both cases of alphabet, the three missing symbols are handled as:

ohm	for Ω
Cel (initial cap)	for °C
u (lower case)	for μ (micro)

2. If you have only one case of alphabet (either upper or lower), use it, and these three replacements remain as:

OHM	or	ohm
CEL		cel
U		u

And in addition:

S (siemens)	become	SIE	or	sie
h (hour)		HR		hr
t (tonne)		TNE		tne

Examples:

16 UOHM is 16 μΩ
373.15 K = 100 Cel

Notice that no plurals are used in symbol combinations — MICROOHMS, but UOHM.

ASCII AND KEYBOARDS

Technically, a keyboard is an ASCII keyboard if it generates the proper codes for the full set of ASCII graphic and control characters. Moreover, none of the graphic characters should have any control properties.

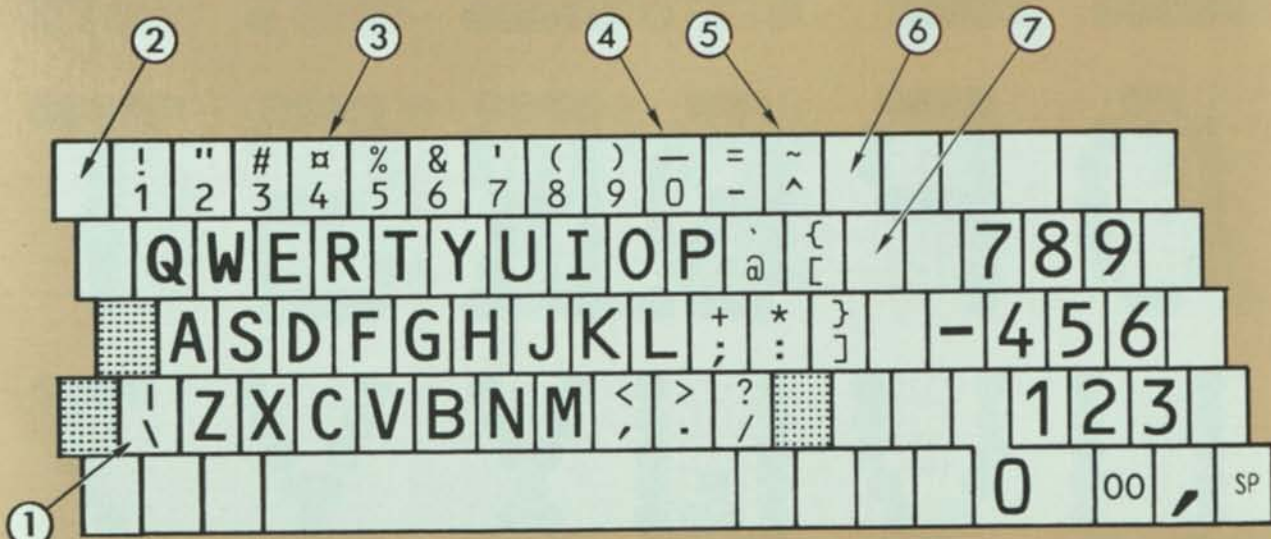
There are many types of special keyboards — Dvorak, a two-sided one used like an accordion with the hands in a vertical plane, Touch-Tone and its derivatives, etc. There are no formal standards to relate these keyboards to ASCII. For typewriter-style keyboards, however, there are two versions given in the American National Standard. One is derived from the usual electric typewriter keyboard, the other is called the "bit-paired" keyboard. Only the bit-paired keyboard will be shown and discussed here, because the other form is the subject of proposals for extensive change due to the growth of Word Processing. ANSI Committee X4A12 is studying this now.

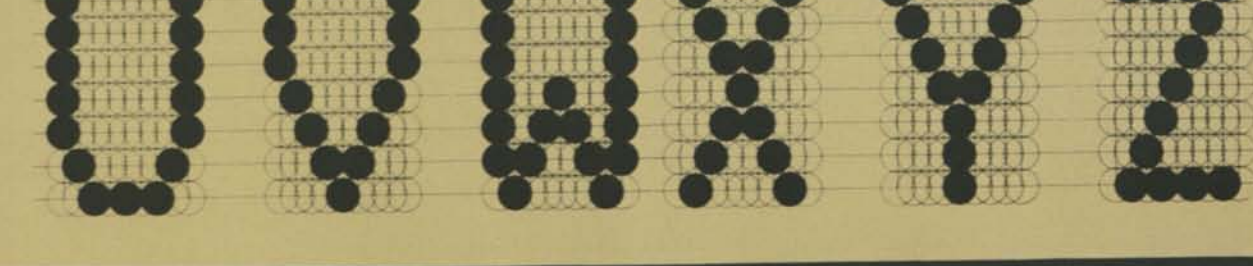
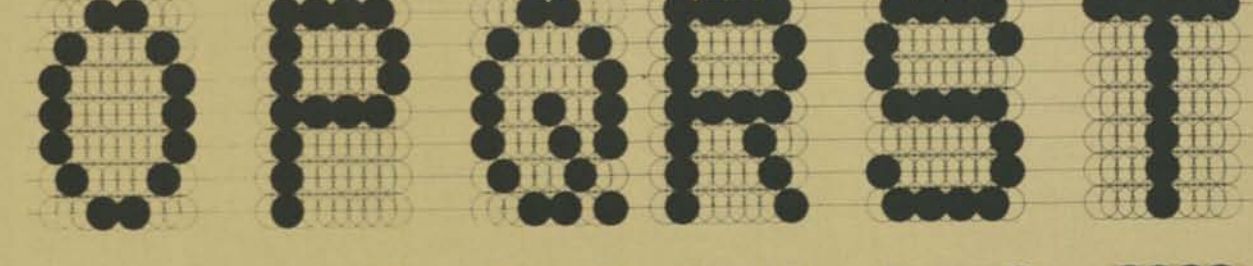
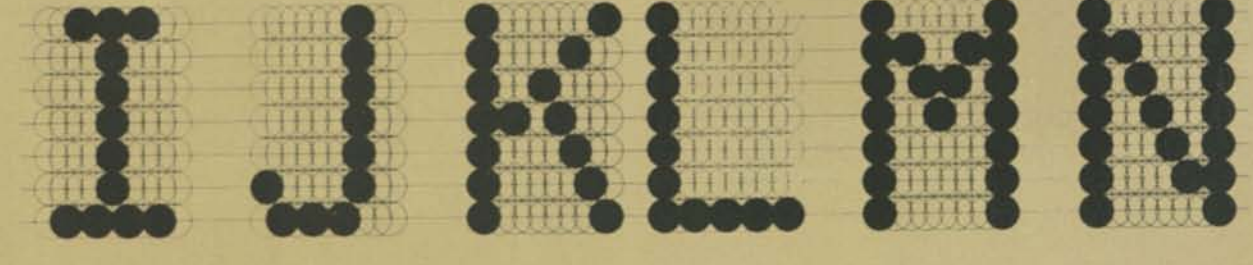
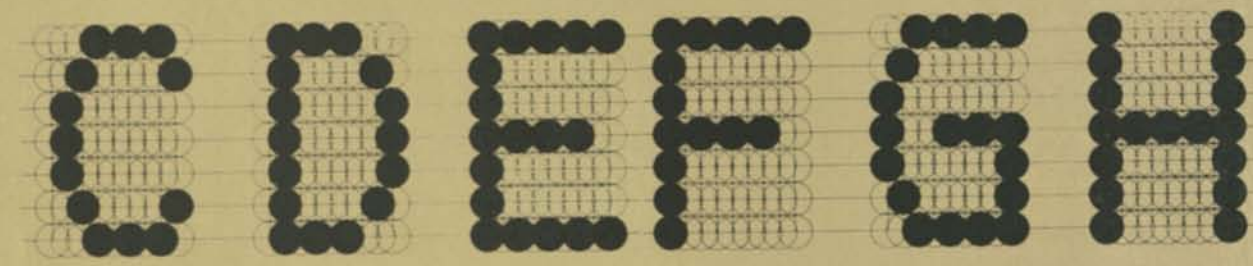
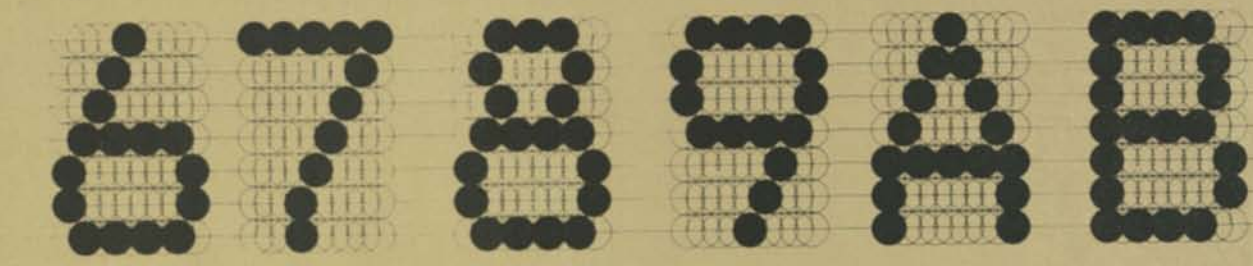
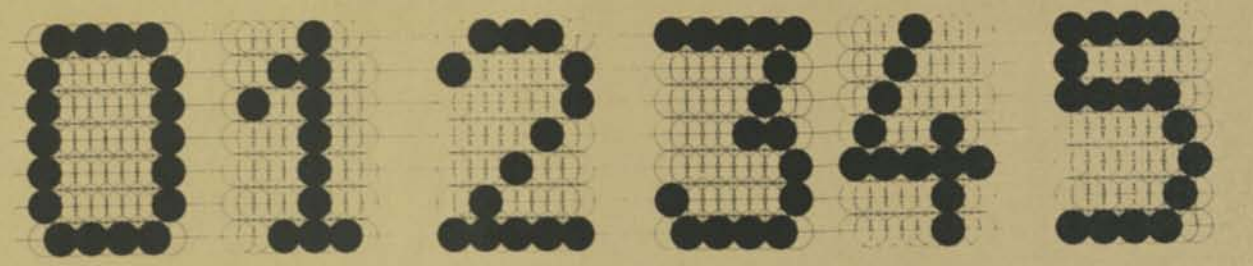
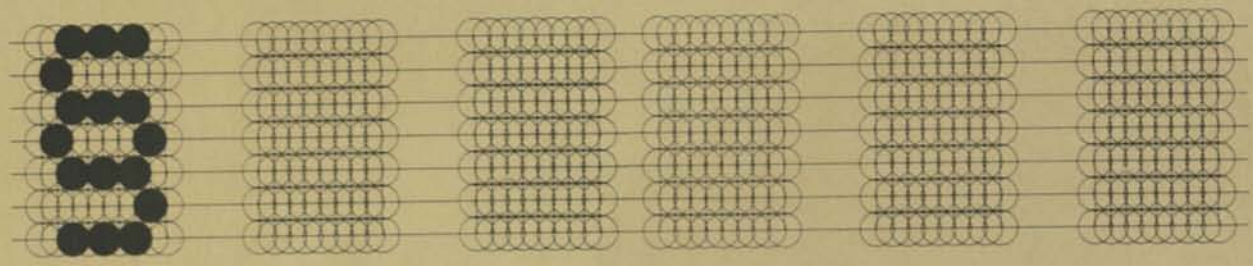
The bit-paired keyboard was designed for minimum circuitry cost. Thus the "at" symbol (4/0) is paired with the accent grave (6/0), "A" (4/1) with "a" (6/1), and "+" (2/11) with ";" (3/11). Thus the shift key affects each other key by only a 1-bit change.

This keyboard is shown in Figure 9. It is the interchange keyboard of ECMA-33. The numbered arrows key to the notes on changes that would make this ECMA keyboard into the ANSI keyboard for ASCII. It is also equivalent to the keyboard of ISO Standard 2530-1975 (Footnote 2).

Notes for Figure 9

1. For an ANSI keyboard, this key is put to the right of the circumflex key, on the top row (see Note 6). The Shift Key is put in its place.
2. If this key exists and is available, the ECMA and ISO standards put the underscore here, removing it from the "zero" key.
3. The ANSI keyboard of course puts a "\$" here in place of the international currency symbol.
4. This is where the underscore is removed for the 48-key keyboard (see Note 2).





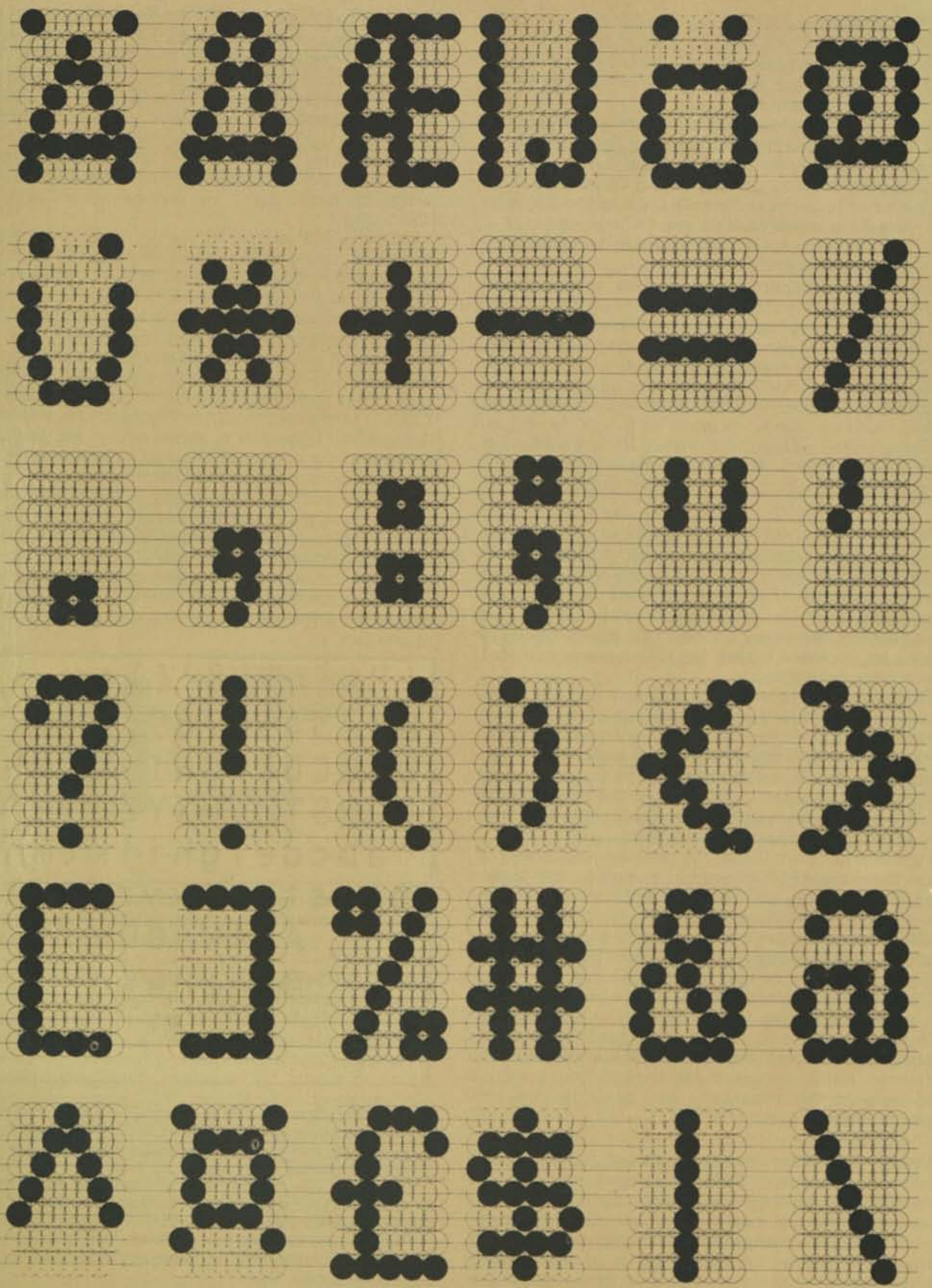


Figure 10.

5. Here ECMA and ISO show the "overline" instead of the "tilde." It's a question of styling.
6. The ANSI keyboard has the reverse slash and vertical bar here, rather than between the shift key and "Z" (see Note 1).
7. The ANSI keyboard specifies the underscore here, in both shifts, rather than the positions shown as options in Notes 2 and 4. Practically no keyboards follow this. In fact, as I am entering this text, this is the **only** key where my Infoton Vistar deviates from the ANSI standard. It has Line Feed there, with Return to its right — a very sensible arrangement.

Customarily, the Control Key is also tied to bit-pairing in such keyboards. The standards recommend that characters created in combination with the Control Key should use the graphic key in sticks 4 or 6 units higher. Thus "X" (5/8) or "x" (7/8) in combination with the Control Key produce CAN (1/8). Unfortunately this also means that Control-C generates ETX (0/3). And whereas Control-X as CAN is used frequently, to erase an input line of text, ETX is not often wanted. Yet it is a common miskeying to hit C rather than X. In many timesharing systems you will get a disconnect rather than a line delete.

Control and Function Keys

The so-called "QWERTY" arrangement is prevalent throughout the world. Even the French "AZERTY" set is being considered for change. But on top of these basics there are hundreds of keyboard varieties. Some of them have "dead keys" (i.e., the platen or printing element is not advanced when they are hit). This avoids having to use BS for accented letters, but it also creates difficulties in code generation.

There are some general good practices that ASCII keyboards should follow. To facilitate usage by those experienced with typewriters, all controls not used with typewriters should be located outside the customary touch-typing area. As a specific example, the Break/Interrupt key should be located where it is a definite effort to reach it (not mixed in with the keyboard). ISO 3244 may be consulted for these considerations.

Function Keys are those that generate sequences of more than one ASCII character. Examples are cursor keys, Erase-to-EOL, etc. They should be located in special clusters. Most importantly, they must all generate ASCII codes for transmission when in character-at-a-time mode. I know of video terminals where the cursors do not generate codes, as they should not while in full page buffered mode; but they still operate in line mode without generating codes. In this case the screen is alterable, but there is no way of detecting it in the computer.

Many keyboards will have some function keys that are unlabeled, for do-it-yourself assignment. These should also be clustered separately, and generate code sequences when in line mode.

ASCII AND DISPLAY PRINTING

When ASCII characters are displayed, it may be on a video screen, paper, or COM (microfiche).

On the video screen there are a number of methods to form the characters, mostly at the manufacturer's preference. They are usually at pica (constant-width) spacing for economy, so an approximation of graphic quality (such as typesetting) is not obtainable. When lower case is available, the risers and tails extend above and below the line for some screens. In others, they fall within the boundary lines of the upper case characters. They may be shown in inverse video (light background block), or highlighted by different brightness or blinking. Controls for this work will be taken up in Part III of this article.

For paper copy one usually finds either direct impact of a formed letter, or stylus printing. Either method is suitable to proportional spacing if desired. Recently

there has been a general trend toward using the 7x9 dot matrix shapes of ECMA Standard 42 for stylus printers. This set of graphics is shown in Figure 10.

For hard print elements, of course, one can get a nearly infinite variety of styles and fonts. There are only two, however, specifically associated with computers — OCR-A and OCR-B. "OCR" stands for "Optical Character Recognition", meaning that the shapes are so styled that a computer-controlled scanner can read the characters as printed on paper, and encode them directly from their shapes.

OCR-A is not suitable for human reading. It's the funny looking one with the diamond-shaped letter "Oh." I won't dignify it by showing the font here. It was thought formerly, with technology of that day, that making humans work harder to read letters would make it easier and thus cheaper for computers to read them. This argument turned out to be specious, and with today's technology there is no need to use anything other than OCR-B.

OCR-B is specified in ISO 1073/2, ECMA-30, and ANSI X3.49. It is the font shown in Figure 11. I have it on my IBM golfball typewriter at home, and on my daisywheel element at the office. So it should be available for most hard elements, including the carousel type.

The first six rows correspond to ASCII sticks 2-7. In the first row, the pound and universal currency symbol are for replacement as needed. In the fourth row, the underline is discontinuous; a continuous form is shown in the auxiliary set. This set also contains a matching accent acute instead of single quote, the real circumflex (not an up arrowhead), a cedilla, and an "m" of better proportion. □

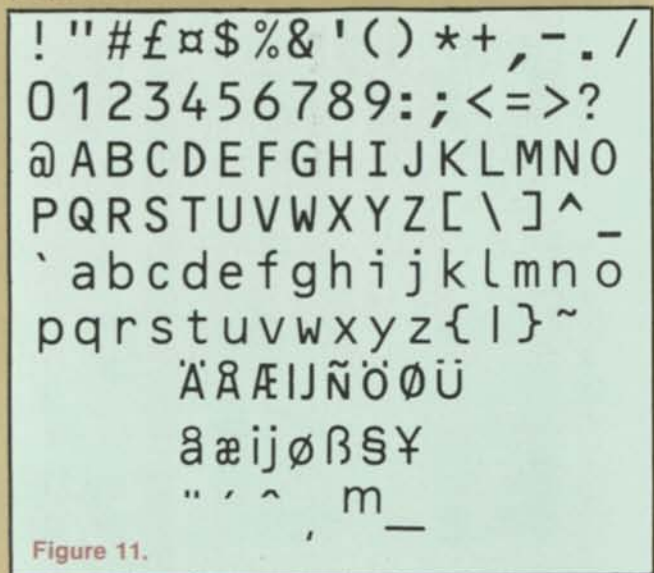


Figure 11.

Footnotes

1. With the distribution, ECMA said "ECMA-53 is an attempt to improve portability of programs. It links the language character sets defined by the language standards, their coded representatives by means of the 7-bit code and the implementations on data carriers (punched tape, punched cards, magnetic tape and magnetic tape cassettes and cartridges). It is a standard of a new type in which already standardized features are assembled in a new standardized combination aimed at supporting interchange and decreasing implementation dependency."
2. ISO 2530 is for the alphanumeric area of the keyboard only. It is augmented by ISO 3243-1975 — Keyboards for Countries whose Languages have Alphabetic Extenders, Guidelines for Harmonizations, and also by ISO 3244-1974 — Principles Governing the Positioning of Control Keys on Keyboards.

The fact that these are "guidelines" and "principles" indicate the complexity of the subject. Typewriter manufacturers now supply over a hundred different keyboard arrangements, as their catalog will indicate.

But what about the severely handicapped, such as quadriplegics? How can they communicate with the world? How can they be educated? Quite possibly this will be accomplished by the movement of their eyes. Bioelectric signals from the eye muscles are one of the last physiological systems to be degraded in many neurological impairments. Noting this, Mr. Ira Laefsky, a student in the Computer and Information Science Department of the University of Pennsylvania is designing a microcomputer-based educational and environmental control device for the severely handicapped using eye muscle movements or electro-oculograms (EOGs) as the means of communication with the student (Figure 2). Various words or symbols are displayed on the TV-type-writer display. As the student looks at one, his eye positions are digitized and sent to the microcomputer, and in this manner he can "tell" the machine answers to questions (for educational purposes). He can also tell it to turn on lights, dial phones, and, if connected to a speech synthesizer, to talk.

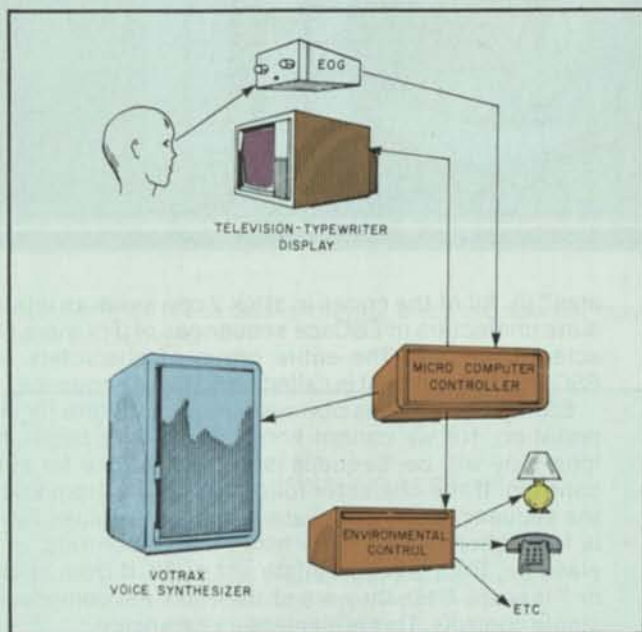


Figure 2. Educational and environment control device for the severely handicapped using electro-oculograms for person-computer communication.

There are many projects of varying degrees of difficulty that would be potentially beneficial to the handicapped, such as a talking terminal for the blind using a speech synthesizer, a voice input telephone dialer for the severely arthritic, a cheap and reliable device for embossing the Braille character set, and a computer communication network for shut-ins. Unfortunately, there are problems with the commercial development of these devices in that the market is limited for a particular device. Therefore, the price must be high, and many of the handicapped can barely afford even the most basic items presently available to them today.

To this end a small group of interested people, under the leadership of Dr. Robert Suding (The Digital Group), met at Personal Computing '77 in Atlantic City last August to discuss some problems of the handicapped and how the computer might help. Efforts of the group have largely been carried on by Warren Dunning of the Franklin Institute in Philadelphia and myself. The group's main purpose is to let people know what problems need solutions and also what items have potential applications towards helping the handicapped. We are doing this through articles in the computer publications, by having booths at the major computer shows, and by a

newsletter soon to be published. For more information on this group, write to Computers for the Handicapped, c/o Warren Dunning, 5939 Woodbine Avenue, Philadelphia, Pennsylvania 19131.

MEDICAL OFFICE MANAGEMENT

Small systems aimed at managing small medical practices are beginning to be seen. Several are available and are in the \$10-15,000 range. These usually include CPU, 32 to 64K memory, dual floppies, CRT, and printer. Software included covers patient information, treatment, accounts payable and receivable, insurance coverage, and payroll.

A new publication aimed at keeping the physician informed about office management systems as well as a host of other developments on microcomputers and medicine is now available. It is called Physicians Microcomputer Report and is edited and published by Dr. Gerald Orosz, Box 6483, Lawrenceville, NJ 08648.

THE FUTURE

Medical instrumentation will continue to shrink in size and double in capacity until the physician's black bag actually contains a tiny diagnostic computer system. Stethoscopes will connect to it for on-line recognition of abnormal heart sounds and ophthalmoscopes will be connected for visual studies. But there will be more. Pop a few electrodes on the patient and the black bag will do ECGs, nerve conduction studies, evoked potential analyses, EEG spectral analysis, and EMGs. It will then process the data, tell the doctor if there are any abnormalities, and store the data in its memory until the day's results can be dumped into a central hospital computer.

Hopefully, fewer people will be in the hospital due to the personal computer revolution. When the day arrives when home computers are as common as television sets and when all of these computers are tied into central information networks, primary medical care might just as well be done electronically. A fairly comprehensive physical examination could be carried out by an average person guided by a computer graphics terminal and using only a minimum of readily available, inexpensive instruments (penlight, tongue depressor, stethoscope, reflex hammer, etc.).

Before self-diagnosis programs come about, there will be programs dealing with preventive medicine, and rightly so. Health risk analyses, to help you prevent diseases, are just now appearing on computer screens. The computer asks whether you smoke, drink, exercise, what you eat, your weight, height, sex, and age, and many other questions. It then reports back with a health appraisal in the form of how many years you can expect to live continuing the same life style. Most important, it can analyze your life style, diet, and social habits and make recommendations for living a healthier life.

Prosthetics research, however, with its bionic flavor, will certainly be the field to lead us into the "far out" future of microcomputer applications in medicine. Artificial appendages will increase in complexity until they approach their human analogs. They will be complete, from the optimum control and functions down to artificial skin itself. The real advance that will grow out of this field will be the development of a method of direct brain-computer communication. Once this has been achieved we will truly step into the Bionic Age and hope for a peaceful symbiosis of man and computer. □

An annotated bibliography of microcomputer applications in medicine is being prepared by the author. If you are interested in receiving a copy or in corresponding about the impact of microcomputers in medicine, write to: Dick Moberg, Department of Neurosurgery, Thomas Jefferson University, Philadelphia, PA 19107.

INSIDE ASCII

CODE EXTENSION — GENERAL PRINCIPLES

Over ten years ago it was recognized that ASCII was the basis for codification of the various symbols used throughout the world. Through it, libraries could store encoded books as well as printed books. And while electronic mail may be quite simple with ASCII and its Roman alphabet, that's not the alphabet of all countries. The USSR uses Cyrillic, the Japanese use Katakana, and the Arab world uses its own semi-script alphabet. Moreover, to send a mathematics textbook by electronic mail one would have to be able to encode the formulas and special symbols peculiar to mathematics, which includes many Greek characters!

This is where the ESCape character and ESCape sequences come in. You can get the whole complicated story from ISO Standard 2022 (or ECMA-35) on Code Extension Procedures. But it will be easier to think of reproducing many ASCII Code Tables on the pages of a book, then replacing the ASCII symbols on all but the first page with the other alphabets we need.

Then we make sure that everyone in the world has the same (code) book. (The resemblance to military code books is intentional.) That's done by registering the page number assignment to characters (actually either a control set or a graphic set, but not both) with the French Standards Body AFNOR. That's the Association Francaise de Normalisation, Tour Europe Cedex 7, 92080 Paris La Defense, FRANCE. But you'll find it perhaps easier to get it from ANSI (see data in the first installment, INTERFACE AGE, May 1978).

The registration procedure is spelled out in ISO Standard 2375. It is carefully controlled to prevent frivolity and cluttering up the assignment books, for that all costs money. But the important control and graphic sets of the world may be registered and assigned their own unique ESCape sequence for calling or invoking them.

CODE EXTENSION — BASIC RULES

The control ESC, when encountered in a datastream, means that all characters following it, up to and including the first character from sticks 3 to 7, have special interpretation. The delimiting character is called a "final" (F). Those between ESC and the final are called "intermedi-

ates" (I). All of the codes in stick 2 can serve as intermediate characters in ESCape sequences of 3 or more characters in length. The entire group of characters from ESC through the final is called an ESCape sequence.

ESCape sequences obviously require buffers for interpretation, for we cannot know, when they begin, how long they will be. Sequences of length 2 are for single controls. If the character following ESC is from stick 3, the sequences are for private usage, of the class Fp. If it is from sticks 4 or 5, they mean single controls, of the class Fe, from an appropriate set of 32. If from sticks 6 or 7 (except 7/15), they are of the class Fs, composed of single controls. This is elementary extension.

A more complex type of extension is the simulation of one or more 8-bit character sets by alternating between two 7-bit sets. The home base set consists of the C0 (32 controls) set and the G0 (94 graphics plus space and DEL). The alternate sets consist of the C1 (32 controls) set and the G1 (94 graphics plus space and DEL). The 8-bit set (it doesn't have to be just theoretical if you have a full 8-bit capability) consists of the four parts C0-G0-C1-G1.

These four types of sets are all invoked (designated) by 3-character ESCape sequences in this manner, where F is the final (3rd) character:

Sequence	Invokes Set Type
ESC 2/1 F	C0
ESC 2/2 F	C1
ESC 2/8 (or 2/12) F	G0
ESC 2/9 (or 2/15) F	G1

The final character "F" selects the particular set to invoke. Once invoked, encountering or entering an SO shifts to the G1 set in force; an SI shifts to the G0 set in force. SO and SI do not affect the control set.

ISO Standard 2022 defines these matters in far more detail, but that is enough for here. That document is complicated and ingenious, and deserves substantial study.

THE CODE EXTENSION REGISTRY

Table 1 identifies the graphic sets registered to date. Table 2 identifies the control sets registered to date. Re-

PART 3 OF 3

PARTS

By R.W. Bemer



member that these assignments, once registered, may never be changed!

Regis. No.	Final Char.	Name
002	4/0	IRV (Intl. Reference Version) Graphics
004	4/1	UK Graphics
006	4/2	US Graphics (ASCII)
008-1	4/3	NATS Main Graphic Set (Finland, Sweden)
008-2	4/4	NATS Additional Set (Finland, Sweden)
009-1	4/5	NATS Main Graphic Set (Denmark, Norway)
009-2	4/6	NATS Additional Set (Denmark, Norway)
010	4/7	Swedish Basic Graphics
011	4/8	Swedish Graphics for Names
013	4/9	JIS Katakana Graphics
014	4/10	JIS Roman Graphics
015	5/9	Italian Graphics
017	5/10	Spanish Graphics
018	5/11	Greek Graphics
019	5/12	Latin-Greek Graphics
021	4/11	German Graphics
025	5/2	French Graphics
027	5/5	Latin-Greek Mixed Graphics (Greek Capitals only)
031	5/8	Greek Alphabet Set for Bibliographic Use

For a G0 set the ESCape sequence is ESC 2/8 plus the final shown.
For a G1 set the ESCape sequence is ESC 2/9 plus the final shown.

Table 1. Registered Graphic Character Sets

Regis. No.	Final Char.	Name
001	4/0	ISO 646 Controls
007	4/1	Scandinavian Newspaper Controls
026	4/3	IPTC Controls

The ESCape sequence for a C0 set is ESC 2/1 plus the final shown.

Table 2. Registered Control Character Sets

The registry set is available from AFNOR for approximately 172 French francs, say \$35. It would be vital for an equipment or software manufacturer to have it, and it comes in a beautiful 4-ring binder symbolizing worldwide interchange compatibility. But the summary provided here will fill most needs.

...the work I had to do to compact the standard, trying to make it understandable, turned up more than unreadability. So it's back to the drawing board, perhaps for a considerable period of time. . .it's sometimes useful to have symbols whose meaning you can reassign without harm to programming languages. . .

CONTENT OF THE EXTENDED SETS

Figure 1a shows, against the ISO Code, International Reference Version, how the other graphic sets differ in the column/row positions shown. The rows are keyed to Table 1, reminding you that ASCII is "006", or "ISO 646-006".

From this figure we can see that many countries need accented letters as individual characters, not compound via BS (BackSpace). This is particularly true for the double sets 008 and 009, for Scandinavian newspaper transmission, which have characters that cannot be made from ASCII in compound form. For example — Ring-A, a solid, and the angle open and closed quotes.

col	02				03		04	05					06	07			
row	01	02	03	04	10	15	00	11	12	13	14	15	00	11	12	13	14
002	!	"	#	¤	:	?	@	[\]	^	_	`	{		}	-
004			£	\$													
006				\$													~
008-1				\$			ua	Ä	ö	Å	■		ub	ä	ö	å	
009-1		«	»	\$			ua	Æ	Ø	Å	■		ub	æ	ø	å	
010								Ä	ö	Å				ä	ö	å	
011							É	Ä	ö	Å	Ü		é	ä	ö	å	ü
014				\$				Ÿ									
015			£	\$			Š	°	ç	é		ù	à	ò	è	ì	
017			£	\$			Š	i	ñ	ı				ñ	ç	~	
018			£	\$													
019			£	\$..
021				\$			Š	Ä	ö	Ü				ä	ö	ü	ß
025			£	\$			à	°	ç	Š				é	ù	è	..
027	Ξ		Γ		Ψ	Π	Δ	Ω	Θ	Φ	Λ	Σ					

Figure 1a. Registered Graphic Character Substitution

008-2 and 009-2 are shown in Figure 1b. Here these are not exceptions from the IRV, but rather the only graphics assigned in the set. The additions are necessary to set type for newspapers throughout Scandinavia. See the Crossbar-D, Crossbar-O, the A-E ligature, and the Icelandic Thorn.

col	04			05				06			07			
row	01	04	05	00	05	11	12	01	04	05	00	05	11	12
008-2	À	Ð	É	Ë	Ü	Æ	Ø	à	đ	é	þ	ü	æ	ø
009-2	À	Ð	É	Ë	Ü	Ä	Ö	à	đ	é	þ	ü	ä	ö

Figure 1b. Registered Additional Graphic Sets

Figure 1a doesn't show Set 031 because it deviates more and is not of that much general interest. It doesn't show the Japanese Katakana set because that is completely different from the IRV. In fact, Japanese Industrial Standard C6220-1969 is an 8-bit coded set with the IRV (see Set 014 for the dollar and yen signs) in the lower (bit 8 = 0) portion, and Set 013 in the higher portion, with space reserved for future additional controls. This Set 013 is shown in Figure 2. It is shown in its high-order position, to indicate the card codes at the same time.

Figure 2 also shows the Cyrillic set of the USSR state standard GOST 13052-67, but it is not half of an 8-bit set as the Japanese do it. Rather it is another page of extensions. After SO (Shift Out) is used, the Russian register is operative. Following SI (Shift In) it is the IRV. Although this set has no registry number now, it was submitted recently by ECMA, and we expect an assignment soon. By the way, both Katakana and Cyrillic are shown in their OCR font.

Figure 3 shows the contents of the registered control sets. Set 007 serves as control set for the graphic sets 008-1,2 and 009-1,2, for Scandinavian newspaper transmission. And set 026 is the control set for the worldwide newspaper transmission, defined by the IPTC (International Press Telecommunications Council). The 18 control positions not shown, and those where there is no entry, are the same as in the International Reference Version (646-001).

These newspapers are driving composition equipment, not line printers, so they don't need VT and FF. Their set is already defined, so they don't need SO and SI. They have (properly) assigned meaning to three device controls. And they're probably not doing payroll, so they don't need the four information separators. But they do transmit, and instead of choosing their own functions and placement they have chosen to be a registered variant of the ISO Code. And all variants within this controlled and registered cluster can at least recognize each other, even if they can't print it!

		1000	1001	1010	1011	1100	1101	1110	1111
		8	9	10	11	12	13	14	15
b ₄ b ₃ b ₂ b ₁	COL ROW								
0000	0				-	я	э	ю	п
0001	1			。	р	а	я	А	Я
0010	2			Г	й	т	р	Б	Р
0011	3			」	う	ち	с	Ц	С
0100	4			,	и	д	т	Д	Т
0101	5			.	о	е	у	Е	У
0110	6			ヲ	カ	フ	ж	Ф	Ж
0111	7			ア	キ	グ	в	Г	В
1000	8			イ	ク	ネ	リ	Х	Ъ
1001	9			ウ	ケ	ノ	ル	И	Ы
1010	10			エ	コ	ハ	レ	И	З
1011	11			オ	カ	ヒ	ロ	К	Ш
1100	12			パ	シ	フ	ウ	Л	Э
1101	13			ク	ル	ハ	ウ	М	Щ
1110	14			エ	ト	ホ	ニ	Н	Ч
1111	15			ツ	ユ	ヲ	。	О	

Figure 2. Katakana and Cyrillic Sets

Position	IRV			
	001	007	026	
0/09	HT	FO	FO	Format Control
0/11	VT	ECD	ECD	End (a typographical) Command
0/12	FF	SCD	SCD	Start (") Command
0/13	CR	QL	QL	Quad Left
0/14	SO	UR		Upper Rail
0/15	SI	LR		Lower Rail
1/01	DC1			Font 1 Change to normal
1/02	DC2			Font 2 Change to italic
1/03	DC3			Font 3 Change to bold
1/08	CAN	KW	KW	Kill Word (through previous space)
1/12	FS	SS	SS	SuperShift
1/13	GS	QC	QC	Quad Center
1/14	RS	QR	QR	Quad Right
1/15	US	JY	JY	Justify

Figure 3. Registered Control Character Substitution

CODE EXTENSION IN ACTION

To illustrate the operation of code extension, let's imagine some equipment that may not exist now:

- A microfiche reader with automatic location controls.
- A microfiche with ASCII (the 8-bit form) on the first two pages, the other pages containing other sets such as Katakana, Cyrillic, Arabic, Greek, Hebrew, mathematical symbols, astronomical symbols, etc. Also, symbol sets for selecting typestyles, weights, rotations, sizes, and elongations.
- A display screen for the microfiche; it is touch-sensitive and generates 7-bit codes according to location touched on the display.
- As an alternative, keyboard tops with fibre optic bundles molded in as a matrix, so that the keytops can be lighted with different symbols as selected.

Now imagine that we are writing an astrology book:

- Type

Those of you born under the sign of Aries (

- Depress the "astro" key on the special keyboard
- Notice the shift in display for the fiche screen and/or the keytop lighting
- Touch the Aries symbol on the screen (or the keytop)
- Depress SI (Shift In) on the special keyboard
- And return to typing the rest of the sentence

) will find this month ...

Now imagine what a computer would do to the input stream in driving photocomposition equipment. The "astro" key generated an ESCape sequence for an astronomical graphic symbol set that would have been registered by AFNOR. When the input parser recognizes ESC, it analyzes the following characters, and then calls this set of character formation methods from the backup store, generates the character shape for Aries according to the character code after the final character, notices SI, and returns to normal mode.

Now we can envision how all of the world's printed material can be stored in machine-readable form, and interchanged recognizably!

ALTERNATE CONTROLS

Work has been in progress for several years to develop a companion standard for controls for devices such as CRT terminals. In the US this is contained in the ANSI document BSR X3.64, Additional Controls for Character Imaging. In a similar form, this C1 set is before the Codes Committee of ISO Technical Committee 97 (Computers and Information Processing) as document 2 N 868, for consideration at its 1978 May 24-26 meeting.

I had hoped to give the essence of this work in this installment. There were only two negative votes in X3, which one could presume might be answered. Unfortu-

nately, the work I had to do to compact the standard, trying to make it understandable, turned up more than unreadability. It turned up many logical flaws and ambiguities. So it's back to the drawing board, perhaps for a considerable period of time.

Figures 4a through 4e will give, however, some flavor of the controls under consideration.

Figure 4 shows the controls of Format Type (FT) 1 and 2. Format 1 is either the single character of the 8-bit set, shown in the first column as "Ce", or the 2-character sequence of the type "ESC Fe", where Fe is a final character taken from 4/00 to 5/15, and whose column designation is 4 less than Ce. I.e., in an 8-bit code, INDEX would be 8/04. In a 7-bit code it would be ESC 4/04. Format 2 is of the type "ESC Fs", where Fs is a final taken from 6/00 to 7/14.

Figures 4b through 4e show controls with formats beginning with the control "CSI", defined in Figure 4a to be either 9/11 (in the 8-bit set) or "ESC [" (in the 7-bit sets). The six possible formats are:

3a = CSI Pn F	4a = CSI Pn I F
3b = CSI Pn ; Pn F	4b = CSI Pn ; Pn I F
3c = CSI Ps F	4c = CSI Ps I F

Pn stands for numeric parameter(s), Ps for a variable number of selective parameters separated by semicolons. The type 4 formats differ from type 3 only in inserting the intermediate character 2/00 just prior to the final.

In the figures, the parameter value enclosed in parentheses is the default value. That is, if the parameters are not actually inserted, i.e., being null, then the effect is the same as if the default value(s) were inserted.

To give an example of how these controls operate, look in Figure 4d for the second mnemonic, SGR (Select Graphic Rendition). It is represented first by CSI, the Control Sequence Introducer, the parameter, and the final 6/13. This means that when the 4-character string

ESC [6 m

is encountered, it should turn on rapid blink in the field(s) specified on your video screen.

AL = Active Line (containing AP)
 AP = Active Position (where the cursor is)
 EF = Editor Function
 FE = Format Effector
 HT = Horizontal Tabulation
 IN = Introductor
 PAD = Primary Auxiliary Device
 RD = Received Datastream
 SAD = Secondary Auxiliary Device
 SD = String Delimiter
 VT = Vertical Tabulation
 QA = Qualified Area (defined by DAQ, SPA, EPA)
 rfs = reserved for future standardization

Abbreviations for Figures 4a through 4e.

Ce	FT	Type	Param	Mnem	Name
8/00-03	1				(rfs)
8/04	1	FE		IND	INdEx
8/05	1	FE		NEL	NEXt Line
8/06	1			SSA	Start of Selected Area
8/07	1			ESA	End of Selected Area
8/08	1	FE		HTS	Horizontal Tabulation Set
8/09	1	FE		HTJ	Horiz. Tabul. with Justification
8/10	1	FE		VTS	Vertical Tabulation Set
8/11	1	FE		PLD	Partial Line Down
8/12	1	FE		PLU	Partial Line Up
8/13	1	FE		RI	Reverse Index
8/14	1	IN		SS2	Single Shift 2
8/15	1	IN		SS3	Single Shift 3
9/00	1	SD		DCS	Device Control String
9/01	1			PU1	Private Use 1
9/02	1			PU2	Private Use 2
9/03	1			STS	Set Transmit State
9/04	1			CCH	Cancel Character
9/05	1			MW	Message Waiting
9/06	1			SPA	Start of Protected Area
9/07	1			EPA	End of Protected Area
9/08-10	1				(rfs)
9/11	1	IN		CSI	Control Sequence Introducer
9/12	1	SD		ST	String Terminator
9/13	1	SD		OSC	Operating System Command
9/14	1	SD		PM	Privacy Message
9/15	1	SD		APC	Application Program Command
Fs FT Mnem Name					
6/00	2			DMI	Disable Manual Input
6/01	2			INT	INTerrupt
6/02	2			EMI	Enable Manual Interrupt
6/03	2			RIS	Reset to Initial State

Figure 4a. Controls for Character-Imaging Devices

Final	FT	Type	Param	Mnem	Name
4/00	3a	EF	(1)	ICH	Insert Character
4/01	3a	EF	(1)	CUU	Cursor Up
4/02	3a	EF	(1)	CUD	Cursor Down
4/03	3a	EF	(1)	CUF	Cursor Forward
4/04	3a	EF	(1)	CUB	Cursor Backward
4/05	3a	EF	(1)	CNL	Cursor Next Line
4/06	3a	EF	(1)	CPL	Cursor Preceding Line
4/07	3a	EF	(1)	CHA	Cursor Horizontal Absolute
4/08	3b	EF	(1;1)	CUP	Cursor Position
4/09	3a	EF	(1)	CHT	Cursor Horizontal Tabulation
4/10	3c	EF	(0)	ED	Erase in Display
			1		From AP to end (inclusive)
			2		From start to AP (inclusive)
					All of display
4/11	3c	EF	(0)	EL	Erase in Line
			1		From AP to end (inclusive)
			2		From start to AP (inclusive)
					All of line
4/12	3a	EF	(1)	IL	Insert Line
4/13	3a	EF	(1)	DL	Delete Line
4/14	3c	EF	(0)	EF	Erase in Field
			1		From AP to end (inclusive)
			2		From start to AP (inclusive)
					All of field
4/15	3c	EF	(0)	EA	Erase in Area
			1		From AP to end (inclusive)
			2		From start to AP (inclusive)
					All of QA
5/00	3a	EF	(1)	DCH	Delete Character
5/01	3c		(0)	SEM	Select editing Extent Mode
			1		Edit in display
			2		Edit in AL
			3		Edit in field
					Edit in QA
5/02	3b		(1;1)	CPR	Cursor Position Report
5/03	3a	EF	(1)	SU	Scroll Up
5/04	3a	EF	(1)	SD	Scroll Down
5/05	3a	EF	(1)	NP	Next Page
5/06	3a	EF	(1)	PP	Preceding Page
5/07	3c	EF	(0)	CTC	Cursor Tabulation Control
			1		Set HT stop at AP
			2		Set VT stop at AL
			3		Clear HT stop at AP
			4		Clear VT stop at AL
			5		Clear all HT stops in AL
			6		Clear all HT stops in device
					Clear all VT stops in device
5/08	3a	EF	(1)	ECH	Erase Character
5/09	3a	EF	(1)	CVT	Cursor Vertical Tabulation
5/10	3a	EF	(1)	CBT	Cursor Backward Tabulation

Figure 4b. Controls for Character-Imaging Devices

Final	FT	Type	Param	Mnem	Name
6/00	3a	FE	(1)	HPA	Horizontal Position Absolute
6/01	3a	FE	(1)	HPR	Horizontal Position Relative
6/02	3a		(1)	REP	REPeat
6/03	3a		(0)	DA	Device Attributes
6/04	3a	FE	(1)	VPA	Vertical Position Absolute
6/05	3a	FE	(1)	VPR	Vertical Position Relative
6/06	3b	FE	(1;1)	HVP	Horiz. and Vertical Position
6/07	3c	FE	(0)	TBC	Tabulation Clear
			1		Clear HT stop at AP
			2		Clear VT stop at AL
			3		Clear all HT stops in AL
			4		Clear all VT stops
6/08	3c			SM	Set Mode
			1	GATM	Guarded Area Transfer Mode
			2	KAM	Keyboard Action Mode
			3	CRM	Control Representation Mode
			4	IRM	Insertion-Replacement Mode
			5	SRTM	Status Reporting Transfer Mode
			6	ERM	Erasure Mode
			7	VEM	Vertical Editing Mode
			8		(rfs)
			9		(rfs)
			10	HEM	Horizontal Editing Mode
			11	PUM	Positioning Unit Mode
			12	SRM	Send-Receive Mode
			13	FEAM	Format Effector Action Mode
			14	FETM	Format Effector Transfer Mode
			15	MATM	Multiple Area Transfer Mode
			16	TTM	Transfer Termination Mode
			17	SATM	Selected Area Transfer Mode
			18	TSM	Tabulation Stop Mode
			19	EBM	Editing Boundary Mode
			20	LMN	Line feed New Line Mode
6/09	3c		(0)	MC	Media Copy
			1		To PAD
			2		From PAD
			3		To SAD
			4		From SAD
			5		Turn OFF copying RD to PAD
			6		Turn ON copying RD to PAD
			7		Turn OFF copying RD to SAD
					Turn ON copying RD to SAD

Figure 4c. Controls for Character-Imaging Devices

Final	FT	Type	Param	Mnem	Name
6/10-11					(rfs)
6/12	3c			RM	Reset Mode
					(same parameters as SM)
6/13	3c	FE	(0)	SGR	Select Graphic Rendition
			1		Primary rendition
			2		Bold, or increased intensity
			3		Faint, decreased intensity, or secondary color
			4		Italic
			5		Underscore
			6		Slow blink (< 2.5/second)
			7		Rapid blink (> 2.5/second)
			8		Negative (reverse) image
			9		(rfs)
			10		(rfs)
			11-19		Primary Font
			20		1st to 9th alt. font (via FNT)
6/14	3c		(0)	DSR	Device Status Report
			1		Ready, no malfunctions detected
			2		Busy - retry later
			3		Busy - DSR will notify ready
			4		Malfunction - retry
			5		Malfunction - DSR will notify ready
			6		Please report status (DSR or DSC)
					Please report AP via CPR
6/15	3c		(0)	DAQ	Define Area Qualification
			1		Accept all input
			2		Accept no input (protected); do not transmit (guarded)
			3		Accept graphics
			4		Accept numerics
			5		Accept alphabetic
			6		Right justify in area
			7		Zerofill in area
			8		HT stop at start of area (field)
			9		Accept no input (protected); permit transmit (unguarded)
					Spacefill in area

Figure 4d. Controls for Character-Imaging Devices

Final	FT	Type	Param	Mnem	Name
4/00	4a	EF	(1)	SL	Scroll Left
4/01	4a	EF	(1)	SR	Scroll Right
4/02	4b	FE	(100;100)	GSM	Graphic Size Modification
4/03	4a	FE		GSS	Graphic Size Selection
4/04	4b	FE	(0;0)	FNT	Font selection
			(0;0)		Primary font
			1;0		First alternative font
		
			9;0		Ninth alternative font
4/05	4a	FE		TSS	Thin Space Specification
4/06	4c	FE		JFY	Justify
			(0)		Terminate all justify actions
			1		Fill action ON
					(text to/from other lines)
			2		Interword spacing
			3		Letter spacing
			4		Hyphenation
			5		Flush left margin
			6		Center text between margins
			7		Flush right margin
			8		Italian form (underscore last)
4/07	4b	FE		SPI	SPacing Increment
4/08	4c	FE		QUAD	Quad
			(0)		Flush left
			1		Flush left, fill with leader
			2		Center
			3		Center, fill with leader
			4		Flush right
			5		Flush right, fill with leader

Figure 4e. Controls for Character-Imaging Devices

Code	Symbol	Code	Symbol
10/00	(same as 02/00)	11/00	Large circle
10/01	Opening double quote	11/01	Dagger
10/02	Closing double quote	11/02	Superior (superscript) 2
10/03	Club suit	11/03	Superior (superscript) 3
10/04	Diamond suit	11/04	Rectangle
10/05	Heart suit	11/05	Parallel
10/06	Spade suit	11/06	Partial derivative
10/07	Closing single quote	11/07	Lower left corner, floor
10/08	Is implied by	11/08	Upper left corner, ceiling
10/09	Implies	11/09	Upper right corner
10/10	Multiply	11/10	Lower right corner
10/11	Plus or minus	11/11	Perpendicular
10/12	Nabla, or del	11/12	Less than or equal
10/13	Em dash	11/13	Not equal, other than
10/14	Radix point	11/14	Greater than or equal
10/15	Divide	11/15	Paragraph mark, pilcrow
12/00	Section mark	13/00	Capital pi
12/01	Double dagger	13/01	Capital psi
12/02	Dot bullet	13/02	Square bullet
12/03	Capital theta	13/03	Capital sigma
12/04	Capital delta	13/04	Integral
12/05	At least one exists	13/05	Capital upsilon
12/06	Capital phi	13/06	Therefore
12/07	Capital gamma	13/07	Capital omega
12/08	Upward arrow	13/08	Downward arrow
12/09	Right arrow	13/09	Left arrow
12/10	Dot product	13/10	Approximately equal
12/11	Degree	13/11	Opening angular bracket
12/12	Capital lambda	13/12	Logical AND
12/13	Register	13/13	Closing angular bracket
12/14	Copyright mark	13/14	Logical NOT
12/15	Capital xi	13/15	Infinity
14/00	Opening single quote	15/00	Small pi
14/01	Small alpha	15/01	Small psi
14/02	Small beta	15/02	Small rho
14/03	Small theta	15/03	Small sigma
14/04	Small delta	15/04	Small tau
14/05	Small epsilon	15/05	Small upsilon
14/06	Small phi	15/06	Check mark, radical mark
14/07	Small gamma	15/07	Small omega
14/08	Small eta	15/08	Small chi
14/09	Small iota	15/09	Logical universal quantifier
14/10	Identically equivalent	15/10	Small zeta
14/11	Small kappa	15/11	Cap intersection
14/12	Small lambda	15/12	Logical OR
14/13	Small mu	15/13	Cup, union
14/14	Small nu	15/14	Overbar
14/15	Small xi	15/15	(same as 7/15)

Table 3. Names of the Additional Graphics, 8-bit Set

	08	09	10	11	12	13	14	15
0				○	§	π	·	π
1			“	†	‡	ψ	α	↓
2			”	‡	•	■	β	ρ
3			♣	‡	⊖	Σ	θ	σ
4			♦	□	Δ	f	δ	τ
5	RESERVED FOR CONTROLS		▼	l	E	T	ε	υ
6			♣	⊖	φ	∴	φ	✓
7			·	L	Γ	Ω	γ	ω
8			∩	∟	†	‡	η	χ
9			∩	∟	→	←	ι	ψ
10			x	∟	·	≈	≡	ξ
11			±	∟	°	<	κ	∩
12			v	<	Λ	Λ	λ	v
13			-	*	⊗	>	μ	∩
14			♣	>	⊗	∟	ν	—
15			+	‡	Σ	⊗	ε	

Figure 5. 8-bit ASCII Proposal

CODE EXPANSION

We have seen how ASCII was *extended* by making many related pages of the 7-bit code. It is also possible to *expand* ASCII into an 8-bit code, or even 9-bit and 10-bit if we wished, for that matter. But an 8-bit code is obviously the most logical one to concentrate on, and this has been under development for several years.

The proposed 8-bit Expanded ASCII Code is shown in Figure 5. The identification of the graphic symbols is given in Table 3.

One can observe many interesting things about this set. For example, it has the entire Greek set of small letters except for "omicron", with eleven capitals to go with others from the Roman capitals to complete the Greek set. But apparently the committee didn't follow 646-031, the Greek alphabet mentioned in Table 1. They didn't use the customary ordering "alpha-beta-gamma", the way we learn our "a-b-c's". I suppose it is argued that this set will never be used for language, only math symbols. And 646-027, shown in Figure 1a, does not demand the special capital "upsilon" shown in position 13/5. If the Greeks can agree to using a Roman capital "Y" for upsilon, could the Americans?

You'll notice some math symbols, but not enough for APL. In fact, the whole set seems highly slanted to mathematics, rather than business. Of course there are the four corner symbols for forms. Presumably the card suits will strike your eye, and you will wonder why so many other useful symbols were ignored in favor of these. Don't worry, they will always come in handy; it's sometimes useful to have symbols whose meaning you can reassign without harm to programming languages, etc. The committee were obviously bridge players, for spades collate high.

This proposal has not had real public scrutiny yet, and it must be considered no more than a proposal. Presumably X3 will agree about July that it should be sent out for formal public review and letter ballot. My guess is that it will not be adopted in just the form you see here.

FUTURE FOR ASCII

The methods are in place for codifying all symbols that people use. They may be language alphabets, signs, drawing symbols, or controls for equipments. Robots, for example. Satellites are augmenting conventional telecommunications systems, so that one can borrow cheaply and permanently from electronic libraries.

To prepare for this, other sets are being developed for registry, many through ISO Technical Committee 46/1, Automated Documentation. A 2-page mathematical symbol set is near submission, as are African sets. Work is started for Arabic, which will take about 5 sets to handle fully, although there is a commercial subset of 94 graphics. Another C1 set is being proposed for bibliographic controls. It contains four types — annotation controls, filing controls, reference controls, and subject designators. Other C1 sets can come from process control, animation and other graphics applications, etc.

West Germany has proposed a new ISO project on text communication, to harmonize teleconnection of the more than one hundred varieties of typewriters (and keyboards) throughout the world. The extension method of multiple 7-bit codes is ideal for this (8-bit codes imply too many keys or shift combinations for people to use easily).

I am convinced that microcomputer users are going to develop some fantastic applications that will become widespread enough for their special graphic and control sets to be registered. How about a control set or two for sewing machines?

In fact, it is very difficult to think of any general application where one could not find a usage for these registered variants and extensions. □

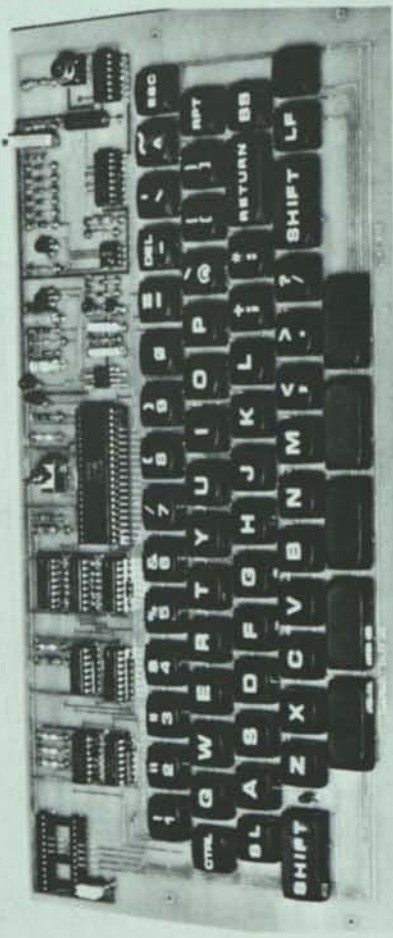
Stand Alone ASCII Keyboard Specification

☆ 4 SIMULTANEOUS OUTPUTS AVAILABLE: THE ONLY ONE ON THE MARKET

1. SERIAL TTL LEVEL
 2. BUFFERED 8 BIT (TRI-STATE LATCH) PARALLEL OUTPUT WITH VALID DATA SYNC PULSE AND LEVEL
 3. 20 MA OPTO-ISOLATED CURRENT LOOP, POLARITY INDEPENDENT
 4. EIA RS232C
- ☆ SINGLE +5 VOLT 300 MA (NOMINAL) POWER SUPPLY (REQUIRED)
 - ☆ INDUSTRY STANDARD 2 KEY ROLLOVER ENCODER
 - ☆ ANSI - COMPATIBLE KEY SET: FOR SLIM-LINE "HIDEAWAY" PACKAGING
 - ☆ SEGMENTED SPACE BAR ALLOWS FAST MULTIPLE-SPACING WITHOUT REPEAT KEY
 - ☆ REPEAT KEY REPEATS AT CHARACTER RATE
 - ☆ USER SELECTABLE UPPER CASE ONLY (KSR/ASR/33 REPLACEMENT) OR UPPER/LOWER CASE
 - ☆ FACTORY SET AT 110 BAUD BUT EASILY ADJUSTED BY USER TO ANY BAUD RATE FROM 110 TO 9600 BAUD
 - ☆ FLEXIBLE PARITY
 - ☆ LED INDICATOR FOR SHIFT-LOCK KEY ELIMINATES CASE UNCERTAINTY
 - ☆ 24 PIN DUAL - INLINE CONNECTOR
 - ☆ LOW PROFILE CASE (OPTIONAL) \$40.00

\$138.00 ASSEMBLED AND TESTED

Plus \$3.00 handling charge. California residents add 6 1/2% sales tax.



Orders accepted by phone or mail.

MASTERCHARGE ☆ VISA ☆ COD ☆ CHECK ☆ MONEY ORDER

COMPONENT SALES INC.

778-A BRANNAN, SAN FRANCISCO, CA 94103

[415] 861-1345

INTRODUCTION

7 8 8 9 to the

American National Standards Committee X3, for computers and Information Processing, has these active subcommittees for programming language standards.

X3J1	PL/I	X3J4	COBOL
X3J2	BASIC		
X3J3	FORTRAN	X3J6	Text Processing

Only X3J6 has a general name, not that of some specific language. Several languages exist in this area, but only one has been formally released to the subcommittee as not proprietary. It's the TEX language, developed at Honeywell Information Systems by Eric Clamons and Richard Keys, with me as advisor. My guess is that the final standard text processing language will look very much like TEX. Two reasons are: 1) it is a powerful general language that is extremely easy to use, and 2) it has the same antecedents as IBM's language SCRIPT.

I'll venture even further, predicting that TEX may supplant (or at least subsume) the BASIC language. The reason for this article is, of course, to give you an advance view of such future usage.

TEX IN GENERAL

TEX differs from many other programming languages in having the possibility of more than one active element. The program you write can act alone, operating on self-contained data, but more often it acts in conjunction with a file called the "current file", which became the current file by bringing it from the permanent store by saying:

```
old filename
```

From this point on TEX may use, in addition to its normal programming language features, the elements of a text editor — to operate on that current file just as you would manually. That is, there is always a pointer to some line of that file. It may be moved backward (as far as the beginning of file) and forward (as far as end of file).

TEX WITHOUT A CURRENT FILE

I like examples to explain programming languages. Figure 1 shows results from the program "power" of Figure 2, whose letter clues on the right identify lines for explanation. TEX doesn't need line numbers. You may use them with some difficulty if you can't break the habit. Labels, identified by "!" in the first position of the line, are better. They don't have to change if one inserts or deletes lines.

```
call power (puts the program into execution)
What number? 97
Up to what power? 30

Exponent Value
1 97
2 9409
3 912673
4 88529281
5 8587340257
6 832972004929
7 80798284478113
8 7837433594376961
9 760231058654565217
10 73742412689492826049
11 7153014030880804126753
12 693842360995438000295041
13 67302709016557486028618977
14 6528362774606076144776040769
15 633251189136789386043275954593
16 61425365346268570446197767595521
17 5958260438588051333281183456765537
18 577951262543040979328274795306257089
19 56061272466674974994842655144706937633
20 5437943429267472574499737549036572950401
21 527480512638944839726474542256547576188897
22 5116509725977649453468030598885114890323009
23 4963064143419831996986398968091856144361331873
24 481417221911723703707680699904910046003049191681
25 46697470525437199259645027890776274462295771593057
26 4529654640967408328185567705405298622842689844526529
27 439376500173838607834000067424313966415740914919073313
28 42619520516862344959898006540158454742326868747150111361
29 4134093490135647461110106634395370110005706268473560802017
30 401007068543157803727680343536350900670553508041935397795649
```

Task completed.

Figure 1. Results from "power"

```
!power out: " " in: " What number? " y:=*in x:=*in a
n=1 in: " Up to what power? " limit:=*in b
out: " " out: " Exponent Value" out: " " c
d
!loop out:(" ",n)l'6," ",y n=n+1 e
if n:gt:limit out:*lf,"Task completed.,"*lf return f
y=y*x goto !loop g
```

Figure 2. A TEX program called "power"

Explanation of Figure 2.

a "!"power" is a label. It's not used in this program. But with many such programs in a single file called "service", one would "call service!power", to execute the file "service" from this label entry point and return without executing any other subprograms in "service".

Several commands may be on a text line, separated by spaces. "out" means print at the terminal the line following the colon. Here it's a string consisting of one space.

"in" also prints the line at the terminal, but it waits for the user to reply and press the Return key. The read-

Part One

TEX Language

By Robert W. Bemer

only variable "in" then holds the reply. Variables "y" and "x" are created to have that value.

- b The power is set to 1, and the user is asked how far he wishes to take the calculation. A new variable "limit" is created for the reply.
- c Three lines are printed at the terminal — a blank line, the heading, and another blank line.
- d I always precede labeled lines with a line of two spaces. Program structure is easier to use. Other than this, and material at a section labeled "explain", few documentation aids are needed.
- e A labeled loop. The interior of parenthesis pairs must be evaluated first. Five blanks are put in front of the power count "n". "j" and "l" are truncation operators (and indicate direction like arrows). The part saved/used is shown by the accent acute ('). If the accent is to the left of the truncation operator, the left part is saved, and vice versa. Here the righthand 6 characters are saved.

A string of five spaces, and the value of "y", are appended to this right-justified (aligned) string. It is printed at the terminal. The value of "n", plus one, is put into "n".

- f If "n" is greater than (gt) the set variable "limit", a line is printed, consisting of a Line Feed (*lf), the closing message, and another Line Feed. "return" means go back to whatever called the program to execute — another program, or the person in manual control at a terminal.

When an *if* statement occurs, the rest of the line is executed if the condition is true. Else control goes to the next line (falls through).

- g "y" times (*) "x" is put into "y". The program then goes back to the label "!loop" and continues until the limit criterion is met.

SIMPLICITY OF TEX

TEX is not complex. There are no arrays to declare, or space to reserve. When a variable is created, by assignment, its space and type are set automatically. The only data type is an ASCII string, which can serve three purposes:

- The name of a variable.
- The content of a variable.
- A procedure, if the string is executable when bracketed by substitution characters (like ALGOL 68, a value will be returned if possible).

The first two properties, name and content, are easy. To define the string "man" as the name of a variable with

content "arm", we enter:

```
man="arm"
```

Many characters can serve as delimiter. Double quotes are best. "arm", as a string, can be the name of another variable, as well as the content of "man". So we can enter:

```
arm="hand" hand="finger" finger="nail"
```

Now if we type "out:man", the terminal responds by printing "arm". The underscore character is used to indicate levels of indirection. If we type in "out:___man", the terminal will print "nail".

If we also type in:

```
ape="arm"
```

and then type "out:___ape", the reply will also be "nail", because a tree was established by defining apes to have arms with the same properties as men.

The HIS processor limits indirection levels to 63, to warn against getting into a loop. It will take a very complicated database to approach this number.

Now the third property — that of being an executable procedure. We may declare a variable "cp" to have string content such as:

```
cp=\splitr:("00",var):2 cents=*r splitr:*l:1\
cp=cp,\ out: "$",*l>"0",*r, ".",cents\
```

The variable "cp" was built in two steps; the comma is the concatenation operator. You will recognize this as a dollar-edit function. The way the procedure string is read is:

- Two zeros preface the value of the variable "var". It's then split by coming in from the right two places.
- This creates *l (lefthand part), and *r (righthand part), now assigned to the variable "cents".
- Another split from the right ensures a forced zero dollar if the value is less than a dollar.
- Then the command is given ("out:") to print a dollar sign, followed by the lefthand part with the leftmost zeros removed. That is achieved by scanning right (>) until a not (*) zero character is found, and keeping the lefthand part at that point, we would have said "<" instead. (Scanning from the right is the same, except with "<").
- Still in the "out" command, there follows a decimal point and the cents value.

Now when "substitute" mode is turned on, the TEX sees the name "cp" delimited by substitute characters, the content of "cp" will get substituted. Being executable, a

value is returned. For some examples, if the vertical bar is made the "subs" character via the statement "subs|", here are some typical results:

```
subs |
var="0000123456" |cp|
$1234.56
var=0 |cp|
$0.00
var=69 |cp|
$0.69
```

Now we'll examine "today", another TEX program that doesn't use a current file (Figure 3).

```
!today _ 1978-01-27 author:RWBemer, 602-942-1360 a
clear * scan:*date:"- yr=*l Leap=(yr/4)*4 b
scan:*r:"- mo=*l da=*r suf="stndrdththth" c
del="000031059090120151181212243273304334" d
dy=" Mon TuesWednes Thurs Fri Satur Sun" e
m1=" January February March April" f
m2=" May June" g
m2=" July August September October" h
m2=" November December" i
a=(yr+11)/4 a=(a+yr)/7 incr=*rmdr+3 j
if mo:lt:7 M=(m1')(mo*10)<" " k
if mo:gt:6 M=(m2')((mo-6)*10)<" " l
tempord=(del')(mo*3)l'3+incr+da m
if mo:gt:2 if leap:eq:yr tempord=tempord+1 n
fw=tempord/7 fd=tempord-fw*7+1 ord=tempord-incr o
day=(dy')(fd*6)l'6>' "' , "day" p
fd=fd,(suf')(fd*2)l'2 q
a=("0",ord)l'2 I=a[l'1 X=a'l'1 suffix="th" r
if I:eq:1 if X:ne:1 suffix="st" s
if I:eq:2 if X:ne:1 suffix="nd" t
if I:eq:3 if X:ne:1 suffix="rd" u
ord=ord,suffix subs | v
out:*lf,"Today is |day|, 19|yr| |M| |da|" w
out:" - the |fd| day of Fiscal Week |fw|," x
out:" - the |ord| day of the year, and" y
time=0 sec=*time'l'2 hm=*time'l'5 z
if sec:lt:25 time=hm goto !end aa
min=hm'l'2 hr=hm'l'2 min=("0",min+1)l'2 bb
if min:eq:60 min="00" hr=hr+1 cc
if hr:eq:24 hr="00" dd
time=hr,"",min ee
ff
!end out:"it is now |time|",*lf nosubs return gg
hh
!explain out:" " ii
out:"'today' gives the characteristics of" jj
out:"the moment, including date, day of the" kk
out:"week, ordinal day, fiscal week and day," ll
out:"and time." return mm
```

Figure 3. A TEX Program for Today

```
Today is Tuesday, 1978 August 01
- the 2nd day of Fiscal Week 31,
- the 213th day of the year, and
it is now 08:26
```

Figure 4. Typical Output from "Today"

Explanation of Figure 3.

a The "_" following the separator space means that the rest of the line is a comment or remark. I always put a revision date here no matter what the file system does. If need be, time-of-day can be added to be

more precise as to latest version. TEX is a "find-a-bug-a-minute" language (but you make fewer mistakes). It's usual to develop programs very quickly, and so require at least hourly precision!

It's also my custom to "sign" my name as program author, as suggested by Dr. Grosch, President of the Association for Computing Machinery. I go further and give my home phone number in case my programs don't work. If so (I've only been called twice), I dial up the computer (one that I use, or else the caller's) from the HIS terminal and phone just next to mine.

b "clear *" clears (destroys and negates existence of) all variables the caller of this program may have at the time. It's not good practice to use it in general service routines.

"date" is a ready-only variable from the computer system, with the value YY-MM-DD for the current date, in International and American Standard form.

The "scan" verb is like "split", except that it breaks a string on a given substring. Here it is the "-", which becomes the content of "m" (*middle).

"yr=" assigns the value of "1" (YY) to the variable "yr", if it exists. If not, it creates such a variable, and remembers that it is a numeric-valued variable.

The variable "leap" is set to the value of "yr divided by 4", without remainder, multiplied by 4. For 1978, "leap" equals "76".

c The righthand part (MM-DD) is scanned for the "-", which puts "MM" in the lefthand part (l) and "DD" in "r. These are assigned to "mo" and "da" respectively.

A variable "suf" is created with the string content shown. It can't be used in arithmetic operations or comparisons; if tried, a diagnostic will say:

```
executing file goof line #3
which=da+suf
'stndrdthththth' is not a legal number
```

The content may look strange at first, but see how it transforms to "1st 2nd 3rd 4th 5th 6th 7th".

d A numeric string is defined. Its property is that the sequential groups of 3 are the accumulated days of the year at start of each month (ignoring leap year).

e "dy" is defined to be the sequential set of names of the days (less the string "day"), in equally spaced subsets of 6 characters each.

f Another string is defined — essentially a vector when taken in groups of 10 characters.

g "m1" grows by adding a further string to itself.

h "m2" is created for the last six months of the year.

i And completed by concatenating another string.

j This line embeds the algorithm for determining the first day of the year for the 19th century. For 1978 it is $78 + 11 = 89$, divided by 4 = 22, plus $78 = 100$, divided by 7 = 14, with a remainder (from another implicit variable "*"rmdr") of 2. Then "incr" = 5. The first Monday of the first fiscal week of a year has the ordinal value of (7-incr).

k This line is not executed, because it starts with "if mo is Less Than 7", and "mo" is 8 for August.

l This line is executed. It shows a scan operator in action. ">" means to scan to the right, and "<" means scan to the left. But first the function within the nested parentheses must be evaluated, getting a value of 20 $((8-6)*10)$. The variable m2 is then truncated right 20 positions, yielding " July August". Now all parenthetical evaluation is complete, and "<" says "scan from the right, to the left, saving the right part when a space is found". This leaves the string "August".

Now we see that declaring the variable "m2"

- created a vector of elements, each of variable length!
- m "tempord" is declared to have a value derived from using only the rightmost three (212) of the first 24 (8*3) characters of "del", adding the numeric day of month, and adding "incr", which reflects year start.
 - n From March on (mo Greater Than 2), and if it's a leap year (Leap EQuals yr), "tempord" gets bumped by 1.
 - o "fw" (fiscal week) is the integer quotient of "tempord" divided by 7. "fd" (fiscal day) is the remainder from the division (*rmdr wasn't used) plus 1.

Finally, the ordinal day of the year is obtained by subtracting "incr". These contortions were caused by fiscal weeks starting on Monday, not Sunday. Because we do it by division, Sunday must occur as the 7th day of the fiscal week, not the 0th.

- p The word prefix for the day of the week is picked up. E.g., for Sunday the fiscal day is 7. So we pick up all 42 characters of "dy", save the righthand 6, and scan right until we come to a character that is *not* (^) a space, and save the righthand part, which is "Sun". Then the string "day" is concatenated.
- q Now we pick up the correct ordinal suffix from "suf", as defined in line "c". Appending it to the numeric value for the fiscal day, we get the ordinal value, which is reassigned to "fd". Now, however, the value of "fd" goes from numeric to string, and the processor redefines the variable type.
- r TEX arithmetic produces answers in normal form only. Results have no leading zeros. Here "ord" is forced to 2 digits so a test can be made on both the tens (X) and the units (I) positions. The string character "0" is concatenated in front, and the two positions extracted. Also, the suffix for the ordinal date is set to "th".
- s If the units value is 1, and the tens value is not 1, the suffix is "st" (1st, 21st, ... 91st). For 11 only, it is 11th.
- t Similarly, it's 12th, but 2nd, 22nd, etc.
- u Similarly, it's 13th, but 3rd, 23rd, etc.
- v The suffix is now appended.

The "substitution mode" is put in force, with the vertical bar as the substitution delimiter for this time. From now on, until the mode is turned off with "no-subs", the TEX processor checks every line before execution, to see if it has pairs of this character. If so, the variable name(s) thus delimited has its value substituted before the line is executed. If the variable content thus substituted is executable, it is executed in its turn.

- w The delimited (by double quotes in this case) string is printed. If that string has been the content of a named variable, "out:variablename" would do. Four substitutions are made before display.
- x After two substitutions, the second display line.
- y After one substitution, the third display line.
- z "time" is another read-only variable from the computer system (it has its own time-of-day clock) with the format "hh:mm:ss". "sec" gets its value from the "ss" part, "hm" getting "hh:mm".
- aa If the seconds value has not exceeded 25, the "hm" value is good because it requires no rounding (the value is 25, not 30, to approximate system response and printing the first three lines of Figure 4). Here a jump to the label "lend" displays the last line, turns the subs mode off so as to not make succeeding programs act erroneously, and returns.
- bb Else "min" is extracted as the "mm" value of "hm", and "hr" extracted as the "hh" part. Rounding up is done by "min + 1", but again an arithmetic operation yields normal form. So possible single digit conditions must be preceded by a "0", and truncated left 2 positions from the right.

- cc This could yield a rounded value of 60, so we must go to the next hour, set "min" to "00", and add 1 to the hour.
- dd Hour roundup might result in going into the next day. But it wouldn't warrant going back to correct what was already displayed, because that might have been perfectly true when it came out.
- ee Now the hour-minute time is reconstructed.
- ff The standard lines of spaces preceding labels.
- gg After one substitution the last display line is put out. A Line Feed is added for easier display reading.
- hh Again a line of spaces before a labeled line.
- ii Every TEX program (without exception) has a label "explain". It's especially useful when keywords that describe programs are collected in a file. Then an inventory program can ask for the keywords describing the process you want to do. Getting one or more program names with a lot of hits, you'll want to know more explicitly what they do. So "call program!explain", which starts to execute "program" at the label "!explain". As you can see from the rest of the lines of this program, it is just a series of "out" to display. That's not the only use. Intermediate labels in the "explain" section can be used to give selected reminders when the called program is used incorrectly.

UTILITY OF TEX

TEX is used for many different types of applications. It's great for "programmerless" computing in large systems where one would otherwise have to learn a Job Control Language. Instead, the needed input is asked for interactively, and the replies are used to tailor a JCL pattern and run the job automatically.

It's been used for relational databases, computer-assisted learning, a software factory, and design prototyping of software. It's easy to teach, easy to use, and one can usually write an application in 1/2 to 1/3 of the time required to use most other languages.

In this brief introduction we haven't yet shown how the editor portion is used. You get two quickies for a start.

The one in Figure 5 will list line number and length for a source program in BASIC, etc. Just a few clues are needed to understand it — "*cl" is the current line, "*lcl" is its length, *eof is the end-of-file condition, and "f;1" means move the pointer forward 1 (go to the next line).

```
count=0

!loop if *eof out:"Done" return
count=count+1 out:count," ",("00",*lcl)E'2," ",*cl
f;1 goto !loop
```

Figure 5

```
!elimdup a=*cl f;1

!all if *eof out:"Task complete." return
b=*cl if a=eqs:b d goto !all
a=b f;1 goto !all
```

Figure 6

The one in Figure 6 will eliminate duplicate lines in an ordered list. The new clues here — "eqs" means "equals the string", and "d" means delete the current line. □

REFERENCES

1. The TEX Subsystem of the Timesharing System, Series 60 Level 66, Honeywell Information Systems, 200 Smith Street, Waltham, MA 02154, Order DF72.

INTRODUCTION

to the

TEX AND INDEXING VECTORS

A great strength of TEX is the ability to handle data not of fixed length. Yet it must be used often as elements of vectors. Some examples appeared in the program "today", shown in Part I. Here are others, starting with the program "morse" of Figure 1.

```

!morse call !init                                a
!start line=*null in:"CHAR(S)? " split:*in:0    b
if *lin:eq:0 nosubs nocase return                c

!loop split:*r:1 s=set>*l lg=line>*null        d
if lg:le:232 line=line," ",m$s$                e
if *lr:ne:0 goto !loop                           f
if lg:gt:232 out:"Output truncated."            g

!display split:line:72                           h
if *lr:eq:0 out:line out:" " goto !start         i
lineone=*l line=*r split:line:1                j
if *l:eqs:" " out:lineone line=*r>!* " goto !display k
scanr:lineone:" " out:*l line=*r,line goto !display l

!init clear * case subs $                        m
set=\etaoinshrdlucbfjgkmpqvxyz,-?:0123456789"()_/\ n
m0="." m1="-" m2="." m3="---" m4="." m5="." m6="." o
m6="." m7="." m8="." m9="." m10="."
m11="." m12="." m13="." m14="." m15="."
m16="." m17="." m18="." m19="." m20="."
m21="." m22="." m23="." m24="." m25="."
m26="." m27="." m28="." m29="."
m30="." m31="." m32="." m33="."
m34="." m35="." m36="." m37="."
m38="." m39="." m40="." m41="."
m42="." m43="." m44="." m45="."
m46="." m47="." m48=" " return                    p

```

Figure 1. A TEX program called "morse"

Comments on the program "morse" are keyed by the letters on the right:

- a The section "init" is called to set up the variables.
- m With "case" mode in force, lower case and capital letters will be interpreted as the same. The substitution mode is set for the character "\$".
- n The set of characters to be translated is put into the variable "set", in pseudo frequency order, to minimize search time.
- o Variables "m" (sub "s") are assigned the Morse code equivalents. The letter "e" has position zero in the variable "set", so "m0" is assigned its code — a single dot. Position 48 is one beyond the last in "set", so spaces and unassigned characters will be represented by no code, being "m48".

- b The output line is created as a null string. An input string is requested. Splitting it 0 places from the left puts the whole input string in *r, preparing it for the loop in line "d".
- c On a null response (a return only), the modes are turned off, and control returns to the caller.
- d The leftmost character is now in *l. The value of "s" is determined by scanning "set" for that character. The length of the current output line is determined by scanning for the null character.
- e TEX variables are limited to 240 characters. If the line is not already greater than 232, it is OK to add one space and the maximum of six dots and/or dashes.
- f If characters of the input remain, return to "loop".
- g Otherwise stop there if the limit is reached, and fall through to "display".
- h The first 72 characters of the output line are taken.
- i If no others remain, the line is displayed and the process is repeated by going to "start".
- j "lineone" takes the first 72 characters, "line" the rest. *l gets the initial character of "line".
- k If it is a space, the last Morse character in lineone is complete. We display it, kill the leading blanks in "line", and repeat the display process.
- l Otherwise "lineone" is scanned from the right for the first space. *l is complete for display, and the residual is shifted to the beginning of "line".

We usually record DWECO time (Develop, Write, Enter, Check Out) for TEX programs. This Morse program took an even three hours. When it ran, I guessed that it might take an hour to modify it to a Braille program. It actually took 38 minutes, including looking up Braille symbols in the dictionary!

Figure 2 is a later version of that original Braille program. Another 3 hours are invested, because there is now an option to read directly (for the sighted) or emboss (in reverse — the unsighted read the indentations on the back of the paper). Plus a self-test option not shown here.

It has also been modified to use single symbols for the digits and other punctuation. This permits direct comparison with a Fortran program for Braille that was

Part Two

TEX Language

By Robert W. Bemer

shown on page 25 of the 1978 April issue of Datamation magazine.

```

!braille call !init a
!start out:" " in:"CHAR(S)? " split:*in:0 b
if *lin:eq:0 nocase nosubs clear * return c

!loop if count:eq:24 call !reset d
split:*r:1 s=set>*l count=count+1 e
|see it| t=t," " t|s| m=m," " m|s| b=b," " b|s| f
|emboss| t=t|s|," " t m=m|s|," " m b=b|s|," " b g
if *lr:ne:0 goto !loop h
call !reset*goto !start i

!reset out:t=*null out:m=*null out:b=*null j
out:" " out:" " count=0 return k

!init in:"Want the embossing mode? " case q=*in'J| l
see it=\if q:nes:"y"\ emboss=\if q:eqs:"y"\ subs | m
count=0 a=" " b="." c="." d=".." n
|emboss| b="." c="." o
set=\etao\inshrdlucbfgjkmqpvxyz,.-+*0123456789%")&/\ p
t0=c t1=b t2=c t3=c t4=b t5=d t6=b t7=c t8=c t9=d q
t10=c t11=c t12=c t13=d t14=d t15=d t16=b t17=c t18=d
t19=d t20=d t21=c t22=b t23=d t24=d t25=c t26=a t27=a
t28=a t29=b t30=b t31=c t32=a t33=a t34=a t35=a t36=a
t37=a t38=a t39=a t40=a t41=a t42=d t43=c t44=c t45=b
t46=d t47=b t48=b t49=a
m0=b m1=d m2=a m3=b m4=c m5=b m6=c m7=d m8=d m9=b
m10=c m11=a m12=c m13=a m14=c m15=d m16=d m17=a m18=a
m19=c m20=d m21=c m22=d m23=a m24=b m25=b m26=a m27=b
m28=a m29=a m30=c m31=a m32=b m33=c m34=c m35=d m36=d
m37=c m38=d m39=d m40=c m41=b m42=d m43=c m44=d m45=d
m46=c m47=a m48=b m49=a
b0=a b1=c b2=a b3=c b4=a b5=c b6=c b7=a b8=c b9=a
b10=c b11=d b12=a b13=a b14=a b15=a b16=a b17=c b18=c
b19=c b20=c b21=d b22=b b23=d b24=d b25=d b26=b b27=b
b28=d b29=d b30=b b31=b b32=d b33=a b34=c b35=a b36=b
b37=b b38=c b39=d b40=d b41=c b42=b b43=b b44=d b45=d
b46=d b47=c b48=d b49=a t=*null m=*null b=*null return r

```

Figure 2. A TEX program called "braille"

Comments on the program "braille" are:

- l Initialization is started by selecting the embossing mode, or not. "q" is the first letter of the reply, and case-independent.
- m "see_it" is assigned to mean "if reply wasn't yes". "emboss" is assigned to mean "if reply was yes".
- n "count" is set to zero for the output line length. Variables "a" through "d" are assigned as the four possible conditions in the top, middle, and bottom rows of the Braille symbols.
- o But if the embossing mode is selected the symbols must be inverted, which affects "b" and "c" only.
- p Again the set of allowable input characters.

- q Through line r, the three components of each symbol are assigned. The output lines "t", "m", and "b" are created null, and the program returns to "!start".
- d When count reaches 24, 72 columns are used up, and it is time to output the line so far by calling "!reset" (j and k), which outputs the three lines and resets them to null, adding two spacing lines.
- e The input character position is found in "set".
- f For the condition "see-it", the new symbol "t|s|, m|s|, b|s|" is added on the right.
- g For the condition "emboss", it is added on the left.
- h If there is more to the input string, continue at "!loop".
- i Otherwise output the last short line and return to "!start" for a possible new input string.

```

!roman clear * subs | a
split="splitr:*l:1 if *lr:ne:0 d=*r*5 output=" b
em="MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM" em=en,em,em,em c
u_str=" I II III IV V VI VII VIII IX" d
t_str=" X XX XXX XL L LX LXX LXXX XC" e
h_str=" C CC CCC CD D DC DCC DCCC CM" f

!again in:"Your number, in digit form? " g
if *lin:eq:0 nosubs return h
split:*in:0 i
|split|(u_str'Jd)<" " j
|split|(t_str'Jd)<" " ,output k
|split|(h_str'Jd)<" " ,output l
if *ll:ne:0 output=em[*l,output m
out:output goto !again n

```

Figure 3. A TEX program for Roman Numerals

Figure 3 presents a few new usages:

- b Having set the substitution mode, "split" is defined to contain an incomplete piece of procedure (under the 240-character limit, being the content of a variable). This is done for both storage economy and understanding in lines j, k, and l.
- c The Romans didn't provide well for large numbers. A long string of "M"s must be created for truncation.
- d The units position equivalents are a string vector.
- e An equivalent pattern for the tens position.
- f And for the hundreds position.
- g After creation of constants, a number is requested.
- h On a null reply, control is returned to the caller.
- i The old trick, to get the input into *left.
- j The content of "split" is substituted prior to execution. It picks up the rightmost position, and if that is

not null (meaning the process is complete) it is multiplied by 5, and "output" is set to the value obtained by truncating the units string that many positions, scanning left until a space is found, and keeping the righthand part. Thus the Roman equivalent of the units digit is obtained.

- k The process is repeated for the tens position, and the units position data appended.
- l The same for the hundreds position.
- m If the length of the lefthand remainder is not zero, the number is 1000 or more, and a suitable number of "M"s is extracted.
- n The output is displayed, and one may try again.

Here are two final vector examples, from a program that converts spelled-out numerals to digit form. By this time you should be able to figure out their operation by yourself, and I might as well give you lots of generally good tricks to have in your string bag.

In this example, "xx" is a numeral, in words, from 1 to 99. It is also the output, in digits:

```
trans="##ontwthofisiseiniteel" case
scan:xx:"ty" if *lm:ne:0 call !over19 return
scan:xx:"teen" arg=*l'J2
if *lm:ne:0 call !translate xx="1",digit return
if xx:eqs:"twelve" xx=12 return
call !translate xx=digit return

!over19 arg=*l'J2 call !translate xx=digit
if *r:eqs:*null xx=xx,"0" return
arg>(*r<'*a)'J2 call !translate xx=xx,digit return

!translate digit=(trans>arg)/2 if s:gt:22 digit="???"
return
```

In this example, "xx" is a power of 10 (and digit output):

```
z="000000000000" zero=z,z,z
big="#####milbiltriquaquiseksepoctnondec"
if xx:eqs:"hundred" xx="00" return
if xx:eqs:"thousand" xx="000" return
if xx['6:eqs:"illion" s=big>(xx'J3) xx=zero'Js return
```

STAR FUNCTIONS/VARIABLES

Star functions are implicit (not explicit) variables. They cannot be assigned arbitrary values, but are "read-only". (Effectively, the "active functions" of C. Mooers).

The ones most frequently used are those derived from using the verbs "scan" and "split", which generally break a string into 3 parts, of which the middle part may be null.

Part		Length of Part	
Full Name	Abbrev.	Full Name	Abbrev.
*left	*l	*lleft	*ll
*middle	*m	*lmiddle	*lm
*right	*r	*lright	*lr

The forms are:

```
scan:A-operand:B-operand (becomes *m)
scanr:A-operand:B-operand (becomes *m)
split:A-operand:B-operand (becomes *ll)
splitr:A-operand:B-operand (becomes *lr)
```

For the scan verb, B-operand is the string to find within A-operand. "scan" seeks the first occurrence searching from the left; "scanr" seeks the first occurrence searching from the right. It is identical to the resulting "*middle".

For the split verb, B-operand is the number of characters to the point where A-operand should be split into two parts. For "split" the count is from the left; for "splitr" it is from the right. It is identical to the resulting "*ll" or "*lr" respectively. *middle is null, with zero length.

Both A- and B-operands may be the content of variables (including star functions) or explicit strings. Examples:

```
scan:variable:"Bob"
scanr:"ab..yz":substring
split:variable:6
splitr:variable:(3*apples+oranges)
```

There are at least two ways to get the length of a variable:

```
scan:variablename:*null (length = *ll)
split:variablename:0 (length = *lr)
```

"scan" and "scanr" also have variants "scann" and "scannr" respectively, which seek non-occurrence of the B-operand, rather than occurrence. It isn't practical to scan for non-occurrence except for a single character, so in these two cases (scann and scannr) the B-operand is so limited.

In all cases the B-operand may also be specified as a character of a class. TEX now has five star functions that are specifiers for generic classes of characters:

```
*lc lower case the 26 small ASCII letters
*uc upper case the 26 capital ASCII letters
*n numeric the 10 digits 0-9
*a alphabetic *lc (plus) *uc
*an alphanumeric *a (plus) *n
```

Scan always forms *left and *right, but not always *middle, due to conditions of not-character, or character-of-a-class. Figure 4 gives the rules. If *m is null, *r begins with the first character found to meet the scan condition.

On the A-operand	String-length		Class Specifier	Corresponding Operator
	Any	One		
scan	string	string	null	>
scanr	string	string	null	<
scann		null	null	>^
scannr		null	null	<^

Figure 4. *middle and the Scan Verb

Scan and split verbs have counterparts in operators (<>[]). But the operators do not affect functions *l, *m, *r, etc., and only one result is obtained. Figure 5 shows the scan operator rules. For explanation:

- Right portion begins with string/class found/not-found
- Left portion is A-operand less the right portion
- Length is the length of the left portion

Scan from	String-length		Class Specifier	Result saved	
	Left	Right		Number	A-operand
>	x	x	x	Length	
>^		x	x	"	
<	x	x	x	"	
<^		x	x	"	
'>	x	x	x	L portion	
>'	x	x	x	R portion	
'>^		x	x	L portion	
>'>^		x	x	R portion	
'<	x	x	x	L portion	
<'	x	x	x	R portion	
'<^		x	x	L portion	
<'>^		x	x	R portion	

Figure 5. Permissible Scan Operators and Results

Figure 5. Permissible Scan Operators and Results

Figure 6 shows the rules for the split operators:

Split from		Result Saved
Left	Right	
']		L portion
]'		R portion
	' [L portion
	['	R portion

Figure 6. Results of Split Operators

All star functions are operable in either upper or lower case. Some others that are available for use are:

- *account contains the "userid" subaccount number, if any.
- *cl contains the current line of the current file. If there is no current file, or if TEX is at end of file, *cl is null.
- *clvl contains a number equal to the current depth (level) of "calls".
- *date contains today's date in the form "yy-mm-dd". If necessary, fields of *date are zero-filled so that it always has 8 characters.
- *eof contains "f" if there is a current file and TEX is not at end of file. *eof contains "t" if there is no current file, or if TEX is at end of file.
- *in contains the last response to an "in" or "int" command. The carriage return at the end of a line entered at a terminal is not included. If a null response was obtained from a file, or if a single carriage return was obtained from a terminal, *in will be null. *in is initially null.
- *lcl is the length of *cl.
- *lin is the length of *in.
- *null contains the null string, of zero length.
- *random contains a randomly-selected digit whenever *random is referenced.
- *rmdr has the remainder of the last division operation.
- *snumb contains the identifying number of the last batch job spawned in the current timesharing session. If no job has been spawned, *snumb will be null.
- *time contains local 24-hour time in the form hh:mm:ss. Like *date, *time always has 8 characters.
- *userid contains the userid under which TEX is being used.

This trivial program, to find the number of characters in a file, shows the use of *lcl.

```
count=0
!loop if ~*eof count=count+*lcl f;1 goto !loop
out:"File contains ",count," characters." return
```

Star functions also exist for the 32 control characters in the first two columns of ASCII and the "delete" character (shown here in the capital option):

*NUL Null	*DLE Data Link Escape
*SOH Start of Heading	*DC1 Device Control 1
*STX Start of Text	*DC2 Device Control 2
*ETX End of Text	*DC3 Device Control 3
*EOT End of Transmission	*DC4 Device Control 4 (Stop)
*ENQ Enquiry	*NAK Negative Acknowledge
*ACK Acknowledge	*SYN Synchronous Idle
*BEL Bell	*ETB End of Transmission Block
*BS Backspace	*CAN Cancel
*HT Horizontal Tab	*EM End of Medium
*LF Line Feed	*SUB Substitute
*VT Vertical Tab	*ESC Escape
*FF Form Feed	*FS Field Separator
*CR Carriage Return	*GS Group Separator
*SO Shift Out	*RS Record Separator
*SI Shift In	*US Unit Separator
*DEL Delete	

These will be found useful for terminals or other devices that are not full ASCII. They can also be used directly in "out" commands. The following draws a small box, sounding the bell at start and finish:

```
out:*bel,"____",*cr,*lf,"|","*cr,*lf,"|_|",*bel
```

TEX EXECUTIVE FILES

The previous example illustrates an important aspect of TEX. If commands are entered directly at the terminal, they will be executed after the Return Key is pressed. If the same command is made a part of a file, it will not be executed until that file is called. It may be called as the current file ("call **") or as a saved file ("call filename").

Such files are called "executive files". An implicit response from the terminal, such as an extra "return", is

The columns mode localizes the operational mode of editing actions to between two defined column positions.

simulated by a single entry line of "*null".

One valuable byproduct of this separation is that if one is hesitant, while programming and entering a program, about the legality or effect of some command, it is a simple matter to terminate the build mode and actually execute the command to test its effect. When the question is resolved satisfactorily, file building may be resumed.

SUMMARY OF MODES

Modes are set and turned off globally. The paired commands are:

subs	nosubs	trace	notrace
case	nocase	cols	nocols
verify	noverify	octl	noctl

Instances of the first two modes have been shown copiously in the examples so far.

The verify mode (also controlled by abbreviations "veri" and "nove") simply shows at the terminal the result of each action taken, or command executed.

The trace mode (also controlled by abbreviations "trac" and "notr") has no effect upon commands received from a terminal. But if they are received by a called executive file, the verify mode will be simulated. Inserting trace and notrace pairs in the executive file is an effective diagnostic method.

The columns mode localizes the operational mode of editing actions, to between two defined column positions.

The octal mode permits reference to all 512 different 9-bit characters, by defining their names as the octal values prefixed by an assigned character.

ORDER OF OPERATOR PRECEDENCE

Expressions are evaluated left to right wherever operators are of equal precedence. Otherwise evaluation takes place in this order:

1. Any expression within parentheses is evaluated before elements outside the parentheses (the maximum number of real and implied parenthesis pairs for the HIS TEX processor is 16).
2. Unary operators (+, -, or ^).
3. Scan/split operators (>, <,], or []).
4. Multiplication/division (* or /).
5. Addition/subtraction (+ or -).
6. Concatenation (,).
7. Comparison.

INTRODUCTION

to the

TEX AND REAL APPLICATIONS

Now you'll see some examples from a system for physical mail (electronic mail is interconnected, but not shown here). Many readers will find it useful to study the programs for the algorithms, if not for TEX. Their challenge is to write the same programs in BASIC or any other language, for comparison with these TEX programs for the same procedures. I'll guarantee that TEX is much easier and shorter!

The mail system has these parts (individual programs):

- Creating the name/address file.
- Updating the name/address file.
- Adding or changing indication of membership on a specific mailing list.
- Displaying a mailing list, on a cover sheet or as labels.
- Archiving the various source lists; that is, making an integrated set of permanent copies.

The original database is the telephone directory. People working for a company often consider the accuracy of their entry in it to have priority second only to payroll. To understand the programs to be explained here, the database format needs to be known. It is a linear/sequential file of entries of this form:

```
surname, (given) # tel-no # org. # address # room # bldg
```

The entries are of variable length. "address" is a 4-character mail-station here, but it could be a full address just as easily. "#" is used to represent visibly the Horizontal Tab character. It gets replaced automatically in all display programs.

LINE PRINTER DISPLAY OF THE TELEPHONE BOOK

First let's look at a program called "alphatel", which is the proofing run before photocomposition of the telephone book (directory).

```
!alphatel clear * restart=0 a
!restart0 out:"alphatel' has restart capability." b
filename="telbook,q" call texlib/old if fail call ouch c
out:*lf,"Have the file." rs:"#";*ht b out:"Tabs in." d
call texlib/datehead b ib:*cl:".page 65" e
i:*cl:".repl ~" i:*cl:".tabu 27,37,55,61,69" f
filename="sink" call texlib/resa restart=1 g
out:"Restart1 passed. Now you can, in case of" h
out:"any failure, restart at the last restart" i
out:"passed by entering 'call alphatel!restart'." j
goto !nostart1 k

!restart1 filename="sink" call texlib/old f:".tab" l
!nostart1 i:*cl:"*hd" i:*cl:".space" i:*cl:".space" m
call separate call widow n
out:"Restart1 operation complete",*lf o
b r:"*hd";* p
q
r
.break Organization/ Mail
Name Phone Component Sta. Room Bldg. s
.break t
u
.space v
*null w
b out:"Restart2 (paging) passed." call texlib/resa x
restart=2 goto !nostart2 y

!restart2 filename="sink" call texlib/old z
!nostart2 call texlib/formsink which="a" name="sink" aa
id="(my-ID)" bill="(my-charge)" bb
call texlib/print!howmany nosubs return cc

!restart subs $ dd
goto !restart$restart$ ee
```

Figure 1. The program "alphatel"

In Figure 1, the lines of "alphatel" to be explained are keyed to letters on the right:

- b Programs that use a fair amount of computer time should have restart capability, so that not all of the work is lost in case of a failure. The user of this program is advised that it is restartable.
- c At the terminal we would just say "old telbook" to get that file. If it was busy, or had a password we

Part Three

TEX Language

By Robert W. Bemer

did not provide, a message to that effect would be displayed. Manual correction action could be taken. In executive files such actions must be automatic. So we set the content of the variable "filename" to be the name of the wanted file, and call a program named "old". But this isn't one of our personal files. It belongs to another (but synthetic) user called "textlib". All of its programs constitute a library of service routines and general applications. "old" contains:

```
subs | if filename=eqs:*null goto !fail
ergo !fail old |filename|
fail='f' |*svmd| return
!fail fail='t' |*svmd| return
```

The error forms for "goto" and "call" verbs are "ergoto" (or "ergo") and "ercall" (or "erca"). They are obeyed only if anything following them (on the same line) fails. This permits "failsoft" operation and recovery from failures. In this case a failure to get the requested file sets the variable "fail" to "t", and returns.

A comma and the letter "q", for "query", follow the name of the wanted file. This gets us a snapshot copy of the file even if someone else is then changing it. If even this should somehow fail, "ouch" is called to repair the situation.

- d In any lengthy process, it's sensible to advise the user (even yourself) how it is going. All "#" characters are replaced by Horizontal Tab:

```
"rs:" means replace the string
"," means do it the following number of times
"*" means all
":" means with the character following
```

When the process reaches end-of-file, "b" means back up to the beginning. After this is done, another advisory message.

- e The library program "datehead" prefaces the current file with a message "This display requested by (your name) on (the date) at (the time)". It's always useful to do this, to distinguish among several versions or runs. "ib" stands for "insert

before" (the current line) a line that has the content enclosed in the delimiters.

- f After that line (but still before the first current line), two more lines are inserted. One tells the formatting program to replace all characters "A" with incompressible spaces; the next gives the tab stop positions.
- g Now the content of "filename" is changed to "sink", which is where we wish to keep the intermediate results of our process. The library program "resa" (for resave) puts the current file into "sink", and the variable "restart" is set to 1.
- h Now we can tell the user that he can restart, and how to do it. Suppose something goes wrong just after this point. Per instructions the user would enter "call alphetat!restart".
- dd The program would recommence execution at the label "restart", which sets the substitution (subs) mode with the \$ sign. We can't continue on this same line because all substitution in a line is done before any execution of the line begins. So the subs mode must always be set one line prior to its usage.
- ee Now the value of "restart" gets substituted, and in this case we go to line "l".
- k Line "l" is skipped, and control is at line "m".
- l If we had to restart, the correct file position is located by finding the line starting with ".tab".
- m Three lines are inserted. One is a dummy to stand for the eventual heading, and two cause spaces in the display program.
- n "separate" separates the last name starting with "A" from the first starting with "B", etc., and puts the starting letters in the gaps. "widow" does the paging, ensuring at the same time that no new initial letter group starts unless there are at least three entries in that group before a new page begins.
- o Another "we're still here and working" message.
- p The dummy heading "hd" is replaced by the real one, which is specified in lines "p" through "v". The ".break" commands force new lines.
- u This yields a continuous underline in photo-composition.

- w This simulates the extra Return to show that the replacement is complete.
- x Back to beginning of file, which is put into "sink" in the new and modified form.
- y The restart count is bumped up, and control goes to line "aa".
- aa "formsink" automatically formats the file and puts it in "sink". The ASCII line printer routine is set by "a", and the file to print is identified.
- bb "id" specifies whose file "sink", and "bill" is for timesharing charges.
- cc The only variable left unspecified is "howmany", so the "print" program is called at that entry point. After that starts, the subs mode is turned off, and the program ends.

MAKING THE ORIGINAL MAIL CONTROL FILE

The goal is to create a new file (called "telmail") by extracting the name and address from each line entry. Only programs (not people) access this file for modification, so the "#" character is unnecessary. The file entry format is:

```
name HT address\1.16.5.23.38.14.
```

This means that this individual's name occurs on special mailing lists 1, 16, 5, 23, 38, and 14. The periods delimit the list numbers uniquely. But this is after "telmail" has been processed many times. The original format is:

```
name HT address\.
```

Figure 2 is the program that builds "telmail" originally.

```
!makemail out:*lf,"Starting 'makemail' at ",*time,*lf a
clear * call !setup ht:*ht call !fixup b

!nameit ib:*cl:("This file created ",*date) |official| c
!nameit again |suffix| d
filename=sinkname |make_new_else| goto !nameit_again e
|exit1| f
out:""" ,sinkname,"" contain name and address" |exit2| g

!fixup filename="telbook,q" |get_it| h
out:"Have the file" d:""";* b i
!lineLoop ro:"#";3:*rs scan:*cl:*rs j
r:*cl:(*L">"#",ht,*r">"#",".") f;1 |eof| k
goto !lineLoop l

!setup cant=\b out:"Can't find ", name goto !match\ m
bump=\count=count+1 name="n",count\ t=\in:"List number\ n
what no=t,\is? " tag=*in if (*in>*n):ne:*lin goto\ o
official=out:"For the official 'telmail', just CR"\ p
t=\in:"Else what suffix? " sinkname=*userid,"/telmail" q
suffix=t,\,*in if *lineq:0 sinkname="the/telmail"\ r
exit2=\nocase nosubs out:*lf,"Done at ",*time return\ s
exit1=\if fail out:*lf,"No action is taken.",exit2 t
if_yes=\if *in'Jl:eqs:"Y" \ u
retry=\in:"Try another suffix? " \,if yes v
t=\if fail out:"A file "" ,sinkname,"" \ w
doesnt=t,\does not exist." \ doesnt,\already exists." \ x
get_it=\call texlib/old \ make_new=\call texlib/save \ y
get_it_else=get_it,doesnt,retry z
make_new_else=make_new,does,retry aa
null_cf=\call texlib/new\ put_away=\call texlib/resa\ bb
eof=\if *eof b return\ case subs | return cc
```

Figure 2. The program "makemail"

In explanation of Figure 2:

- b The subprogram "setup" (lines "m" through "c") defines variables to have certain procedures as content. Used by most of the component programs in the mail system, they are explained as encountered. They compact the programs and make them easier to read and to understand.
- The variable "ht" is defined to be the Horizontal Tab character. The program is called at the label "fixup".
- h "telbook" is gotten as the current file, again on a snapshot basis. "get_it" is a procedure. For a different computer we would redefine "get_it" to be the corresponding procedure. Thus the kernel of the mail program is portable.
- i After a "working" message, all lines starting "AAA" are deleted. They are the redundant entries for secretaries listed following the people they work with. Back to beginning of file.
- j A loop operating on all remaining lines. "ro" means "replace occurrence". So the third occurrence of the "#" character is replaced by a Record Separator character. The current line is then scanned on that character.
- k Both left and right parts are kept only as far as the first "#" encountered, and adjoined by the content of "ht", which was a HT character. The two characters "." are placed at string end, and the entire string replaces the original line. The pointer is moved to the following line. If an end-of-file signal is received, the process returns to line "c".
- l Else it's repeated until the entire file is converted.
- c For the record, an identifying line is inserted automatically at the beginning of file. Then the user is asked if it's the official telmail file he's making.
- d If he replies with just a CR to "Else what suffix?", it will be the official file. Else it will be a personal copy for private purposes. E.g., for a list of the rockhounds, or equestrians. That file name is the content of "sinkname".
- e The file this program will make should not exist yet. So an attempt to create it is made. If it fails, a message says that a file of that name already exists, and would you like to change your mind about the suffix?
- f If the user gives up, the creation attempt was still a failure, and the program closes by saying that no action was taken.
- g With success, we are so notified, and the process is wrapped up and complete.

UPDATING THE MAIL CONTROL FILE

People leave, and new people come. Those still there get transferred or moved to new offices. And so the content of the telephone directory changes — particularly the address (mail station in this case). Periodic updates of two types are desirable:

- Frequent microfiche copies, or online interrogation, for the switchboard operator.
- Less frequent photocomposed and published copies for all employees.

The programs of this article were motivated in part by failures to get mailing lists changed or corrected by human procedures, often even after two years. And when I converted the old punch card system for the telephone directory, the process turned up six deceased, a woman who had left four years ago for motherhood, and a fictitious "Fred Fortran" in Manufacturing!

Figure 3 shows the program "update", which updates the file "telmail" for currency and correct address content, while preserving the existing mailing list membership data.

```

!update out:*lf,"Starting 'update' at ",*time,*lf      a
clear * call !setup |official|                          b
!update_again |suffix|                                  c
ht=*ht,*us call !fixup                                  d
filename=*;sinkname |get_it_else| goto !update_again    e
|exit1|                                                 f
out:"Have both files" sort *;*(A25)(A1)                g
out:"These names are new, in case you"                 h
out:" wish to add them to any list.",*lf f;1          i
call !just1 b rs:ht;*:ht filename=sinkname |put away|  j
call !listold out:"",sinkname," updated" |exit2|       k

!just1 |eof|                                           l
scan:*cl:ht if *lm:eq:0 cut;1 goto !just1              m
!new nname=*l f;1 scan:*cl:ht                          n
if *lm:ne:0 out:nname goto !new                        o
if nname:eqs:*l scan:*r:"\." d b;1 a:*r f;1 goto !just1 p
cut;1 out:nname goto !just1                            q

!listold |null cf| b paste b a;*:"|" b cuts:"\.";* b  r
out:"These names are no longer valid."                 s
out:" Rerun indicated lists to remove them.",*lf      t
p;* paste return                                       u

```

Figure 3. The program "update"

In explanation of Figure 3:

- b With line "c", the same procedures used in making the original telmail file.
- d Now the same creation process ("fixup") is applied, except that this time the variable "ht" has the extra Unit Separator in it.
- e The filename is defined to adjoin the new file just made with the old file. We wish to transfer the mailing list identifications from the old file to the new, scrapping the old entries. Again, existence is controlled by the procedures. If we wish to give up, that happens via the "exit1" procedure in line "f".
- g A "working" notice. The sort procedure is called. The Unit Separator, as a control character, collates low to any graphic. Thus when the same person appears in both new and old files, the new entry precedes when the two files are sorted together. The sort verb reads "Sort the current file (*) into the current file (i.e., in situ); an alphabetic field of the first 25 characters is defined as the first sort key; sort ascending (A) on the first field".
- h With line "i", a signal that there may be names in the telephone directory now that weren't in there before.
- j The subprogram "just1" is called to ensure that double entries have the information transferred to the new entry, and to delete old entries without a corresponding new.
- l If end-of-file, return.
- m Scan for "ht", which still contains the Unit Separator. If the length of "middle" is zero, we didn't find it, so the line is an OLD entry. In that case we cut it from the file (it is added to a separate "cut" file, and thus deleted from the current file) and go back to try the next line for a new entry.
- n If we get here we have a NEW entry. "nname" is the new name. The next line is inspected to see if it is a matching OLD entry.
- o If "ht" is found it's a NEW, not OLD, entry. So the previous entry was a new listing in the phone book, and "nname" is printed out according to the

cover message of lines "h" and "i". Back to try again for a pair.

- p Now there is a NEW-OLD pair. Do they match? If so, the tag information is picked up from the OLD entry as *r. The OLD entry is deleted; a backup of one line points to the NEW entry. "a:r" means put *r after the text of the current line. That entry is now fixed, and we go to the next line to repeat the process.
- q If we get here, it's a peculiar coincidence that a new person and one no longer in the directory just happen to be adjacent in the ordering. The OLD one is cut, and the new one printed. Upon end-of-file we go back to finish line "j".
- j Back to beginning of file, which now contains only NEW entries. "ht" is replaced in each line by HT,

TEX is. . . useful for prototyping applications that might eventually be done in compiler-type languages. It checks out design and human interfaces fast. Most debugging is done. . . on live data.

and the file is resaved.

- k Having listed all new (added) names during the process, in case they should go on any mailing lists, we call the subprogram "listold", to print the names to be removed from the source mailing lists.
- r A null current file is made per line "cc" of Figure 2. The cut file is pasted to it. After every line we put a vertical bar character, and back to beginning. Now all lines containing the string "\." are cut, because if such a string is found the entry is not on any list, so why bother to tell anyone? And to the beginning again.
- s With line "t", tells the user what will be listed.
- u "p;" means "print all lines". The cut file is pasted just to clear it out for future processes. Return to line "k" and wrap-up of the process.

```

!chnglist out:*lf,"Changing list at ",*time,*lf goto !sk a
!newlist out:*lf,"Adding list at ",*time,*lf          b
!sk clear * call !setup                                c
!list no |what no| !list no                            d
filename="mail",tag |get_it| count=0 p;2 f;1          e
call !vector |official|                                f
!suf |suffix|                                          g
filename=sinkname |get_it_else| goto !suf             h
|exit1|                                                i
rs:(",",tag,"");*:" b count=0 call !match |put away| j
out:"",sinkname," updated by 'mail',tag,"" |exit2|    k

!vector if *eof |bump| _name=*cl'<" " f;1 goto !vector l
limit=count out:"mail",tag," names vectored." return m

!match if limit:eq:count return                       n
|bump| scan: name:"" if *lm:eq:0 call !exception      o
last=*l,*m f:last if *eof b f:last if *eof |cant|   p
call !components M="A"                               q
W1=W3 A1=A3 L1=L3 W2=W4 A2=A4 L2=L4 max=0 linect=0  r
!lastloop scan:*cl:""                                s
if (*L,*m):nes:last call !high goto !match          t

```

Figure 4. Branched to next page

Figure 4. Vectored from previous page

```

scan:*r:*ht split:*l:0 call !components n=0      u
if L2:ne:0 if L4:ne:0 call !region1 goto !have_it v
if L2:eq:0 if L4:eq:0 call !region4 goto !have_it w
!region23 del=1 M="A" if L1:gt:1 if L3:gt:1 del=3 M="W" x
if |M|1:eqs:|M|3 n=del if n:eq:3 goto !have_it y
if L2:eq:0 W2=W1 A2=A1 W3=W4 A3=A4              z
del=1 if L2:gt:1 if L3:gt:1 del=3 M="W"         aa
if |M|2:eqs:|M|3 if del:gt:n n=del             bb

!have_it n=n*1000+900-!inct if n:gt:max max=n    cc
if ("000",max)'C3:eq:6 call !high goto !match   dd
!inct=!inct+1 f;1 goto !lastLoop               ee

!high t=(900-max[*3] if max:lt:1000 b;t |cant| return ff
b f:last f;t a:(tag,".") b;t return            gg

!region1 del=1 if L2:gt:1 if L4:gt:1 del=3 M="W" hh
if |M|2:nes:|M|4 return                       ii
n=del M="A"                                    jj

!region4 del=1 if L1:gt:1 if L3:gt:1 del=3 M="W" kk
if |M|1:nes:|M|3 n=0 return                   ll
n=n+del return                                mm

!components scann:*r:" " split:*r:1 A3=*l scan:*r:*uc nn
W3=(A3,*l)'<*a A4=*r'j1 W4=*r'<*a L3=W3>*sub L4=W4>*sub oo
if L3:gt:1 scan:W3:" " if *l:ne:0 call !exception pp
if L4:gt:1 scan:W4:" " if *l:ne:0 call !exception qq
return                                         rr

```

Figure 4. The program "newlist"

SETTING UP A NEW MAILING LIST

Figure 4 is the program "newlist", used to add indications for a new mailing list, and to update an existing but changed list. Refinements could be made for faster operation and more elegant decisions for search terminations. But that gets too large to show here!

In explanation of Figure 4:

- c With line "d", the standard procedure to begin.
- d "what_no" (Figure 2, line "o"), permits no reply except digits. It does so by scanning the input for a character not a digit. If the count is less than total length, it's not all digits.
- e The designated mailing list is brought as the current file. The first two lines (identifying data) are printed. Then we move to the first name to find.
- f The subprogram "vector" puts each name of the mailing list into one variable of a vector, the names of which run from "n1" to "ni".
- l If not done, "bump" (Figure 2, line "n") ups the subscript count 1 and makes the content of "name" the current line with any righthand spaces removed. The process loops on this single line until end-of-file.
- m When the whole file is assigned, the last value of "count" is assigned to "limit", so the stepping process can be reconstructed. A message, and return.
- g With lines "h" and "i", a familiar process.
- j The old indicator for this mailing list (number and bracketing periods) is replaced by a single period everywhere it occurs. If it's a new list being added, none are found. The count is reinitialized to use the name vector in the matching process, and it is called.
- n If the count is at limit, the name vector is exhausted. Return to line "j", save the modified file, wrap-up.
- o The name string is broken on the comma, which is critical in two ways: 1) If there isn't any we have a problem, and must execute "exception" (not shown).

- p 2) The comma must be included in "last" for the search, else we might think that "Johns" was found when it was actually "Johnson".

- q If not found, backup to file beginning, in case the file may be in wrong sequence. But if such a last name still isn't found, the "cant" procedure (Figure 2, line "m") is executed. It takes the program back to the label "match", for the next name.
- nn "components" is called to obtain the first and middle names or initials. "M" is set to "A", which identifies the name of initials. Whole names use "W".
- oo "right, including all but the last name, is scanned for *not* space. This handles 0 to n spaces after a comma. W3 and W4 the first and middle names. L3 and L4 will be their lengths in characters, and A3 and A4 the initials of those names.

- pp The first initial *must* be the first character. It could be followed by another capital, a period, a space, or small letters of the complete name (which could also have a period, like "Jas." and "Jno.").

- rr "A3" is the beginning capital of the first name. "right" is broken for *its* "right" to begin with a capital. The initial and residual up to the middle name are scanned from the right for the first letter, removing blanks or other punctuation between the given names. "A4" is the beginning capital of the middle name, and the middle name is found in the same way.
- rr If not an initial, the first name is checked. If it contains a space the exception subprogram is called. Line "qq" does the same for the middle name.

- rr Back to complete line "q".
- r "components" works on names in the address file, where the "3" and "4" subscripts are used. In this case it is serving a vectored name, and the subscripts are changed to "1" and "2". "inct", set to 0, is the count (from the first occurrence of the surname) where the best match is found.

- s "max" is the highest "n" found for any set of given names. **RULE:** If both names are present, count 3 for a full name match, 1 for an initial match, for each name (maximum of 6 is possible). If one or the other middle name is missing, match the existing middle name against the other first name if the first match fails.

- s A loop to pick up the surname in the list "telmail".
- t If no match, the set of wanted surnames is exhausted. Do the subprogram "high" to pick the best fit so far. Then get the next vectored name.

- u The address is stripped, and the given names subjected to "components" after the value is initialized.
- v If both middle names/initials exist, "region1" is called to match them. It continues into "region4" to match first names/initials.

- w If only first names for both, "region4".
- x "del" is the scoring value. It's 1 except if both names are not initials, in which case it's 3, and we compare whole names (W) instead of initials (A).

- y The test is made. If pass, "n" is set to the value. If it is 3, the first names had a full match, and there is no use checking crosswise to a middle name. E.g., "Quitecontrary, Mary Mary".

- z We must match first against the existing middle. This line effects interchange for one of the two conditions, so that lines "aa" and "bb" work for both cases.

- aa The same value setting, initial or full name.
- bb If a match, and a higher value than we have so far, the higher is it. E.g., matching "A. Andrew" to "Andrew", the middle name is a better match.

- cc We get here by falling through for regions 2 and 3, by a "goto" from regions 1 and 4. The value is put in the 4th position by multiplying by 1000; the line count is decremented from 900 (in this case the maximum of a single surname that can be tested. It is decremented so that the first individual with the highest value is the match. If "n" is higher than the old maximum, it becomes the new maximum.
- dd 6 is the highest possible value. Stop looking.
- ee Else up the line count, go forward to the next name to test, and repeat.
- ff "t" is the line count for the maximum. But if we didn't get a value of at least 1, there is no match at all.
- gg To beginning of file again. Find the surname once more, and move forward (count) lines. After that entry append the list number and another period. Back up to the first of that surname and retry.

MAKING THE MAILING LIST TO USE

Figure 5 is a program "display", which makes a cover sheet, or labels, as the directing medium for a specific list.

```

!display out:*lf,"Starting 'display' at ",*time,*lf      a
clear * call !setup |official|                          b
!display_again |suffix|                                c
filename=sinkname |get_it_else| goto !display_again     d
|exit1|                                                 e
!list2 |what no| !list2                                f
cuts:(" ",tag," ");* |null cf|                         g
b paste b d;1 a;:*rs b ds:"\","*rs;* b              h
in:"Want labels? " |if_yes| call !labels |exit2|       i
call !cover |exit2|                                    j

!cover in:"Want in mail station order? "              k
|if_yes| call !ms_order sort *;*(A4)(A1)              l
if *in'1|:nes:"Y" call !no_order                      m
ib:*cl:("Mailing List No. ",tag) i:*cl:" "           n
b p="n" cols=2 n=50 see="n" call texlib/n-up!bf       o
filename="sink" |put_away|                             p
out:"List is in your file 'sink' "                   q
out:"You may use 'texlib/2print sink(n)', or"        r
in:"List at the terminal? " |if_yes| b p;*           s
return                                               t

!ms_order |eof|                                       u
scan:*cl:*ht pre=*l split:*r:1                       v
r:*cl:(*l,(" ",*r)[*3," ",pre) f;1 goto !ms_order   w

!no_order |eof|                                       x
scan:*cl:*ht r:*cl:(*l,"                               y
f;1 goto !no_order                                    z

```

Figure 5. The program "display"

In explanation of Figure 5:

- a Through line "f", standard practices of the previous programs.
- g All lines containing the indication for the wanted mailing list are cut from the current file. A null file is created.
- h The cut lines are pasted, and the first line of the file (a dummy blank) is deleted. A Record Separator (*rs) is put after each line. Then all strings between "\ " and RS are deleted. This destroys all of the list indicators, leaving only name and address.
- i if labels are wanted, that program (not shown) is called, and the process ended.
- j Else the subprogram "cover" is called.
- k An option is given.
- l If accepted, the subprogram "ms__order" is called.

- u At end-of-file, a return upon process completion.
- v Else the line is broken on the HT character. The single letter prefix (in this particular scheme) is split off.
- w The line is reconstructed and replaced. Now the address is in front, the numeric part of the address is right-justified, and the names are all left-justified. Go to the next line, and repeat until end-of-file.
- m If the option wasn't accepted, the lines must still be conditioned.
- y The lines are split. Blank spaces are inserted and truncated so that the addresses will be left-justified.
- z To the next line, and repeat until done.
- n The identification is put into the first line of the list, followed by a blank line.
- o Four parameter values are present for the general-purpose program "n-up", so that it may be called at the label "bf" (for "brief"), and thus avoid interactive questioning. The value "n" for "see" means that we do not wish to see the process working. The name list is formed into two columns of 50 lines per page.
- p The formatted list is put into "sink".
- q We are told that.
- r And how to print it with the line printer if desired.
- s Else we have the option to print it at the terminal.

ADVANTAGES OF TEX

The user gets many bonuses from using TEX, many of which are not always expected:

- Application programs are compact. Most fit on a single page, in one field of view. This aids the human mind in comprehension. No thumbing back and forth.
- That means they also consume less storage.
- And there is less to change when you want to modify a program.

Our Dick Petersen made a database entry program and showed it to the people that had to do the work every day. Each suggested redesign to their preferences, plus some needs Dick had not foreseen. He was back in an hour, with the program running their way (a service few users get from programmers that use other languages)! Naturally they were disposed to use it. And their productivity immediately more than doubled!

- Plan-ahead and structured programming, although surely desirable, are not so vital. With TEX, I usually find it easier to jump right in and build a part, thinking meanwhile about the whole. I can always change both programs and file formats with very little effort, if that appears necessary. In the jargon of the software engineers, one can move easily back and forth between "bottom-up" and "top-down" methods.
- Programs are easy to cannibalize for other, but related, purposes.
- TEX is also useful for prototyping applications that might eventually be done in compiler-type languages. It checks out design and human interfaces fast.
- Responses to data entry prompts are easy to validate for type, size, etc. And it's simple to human-engineer a helpful request for re-entry.
- Most debugging is done quickly on live data, rather than waiting for a lengthy compilation process with vast output.

In ending this series on TEX, I'd like to say that I have been a programmer for over 29 years, and I've never felt before that so much problem-solution power was available for me to use so easily. □



COMPUTERS TODAY™

COMPLIMENTARY
COPY

VOLUME 1, ISSUE 1

Published by McPHERTERS, WOLFE & JONES

1979

25 CENTS

report, contact Edward A. Ross, Venture Development Corp., One Washington St., Wellesley, MA 02181.

completely solid state module using a digital microprocessor and other custom-designed integrated

mentary basis by writing to AFIPS, 1815 N. Lynn St., Suite 805, Arlington, VA 22209, (703) 243-3000.

in either tabular or graphic form, using the map. There are 45 police cars in the AVM program.



Using ASCII

By R. W. Bemer



(Author's Note: This condensation is from INTERFACE AGE Magazine — 1978 May, June, and July issues. For your convenience, references to figures and tables have been retained.)

The data alphabet called ASCII (Figure 1, INTERFACE AGE May 1978)¹ is also International Standard 646 (ISO Code)² and Alphabet No. 5 of CCITT (International Consultative Committee for Telephone and Telegraph). It is used throughout the world, incorporated in billions of dollars of equipment.

But it is not always used correctly and wisely. There are misinterpretations and gaps in definition that permit nonstandard usage. Here are the background, peculiarities, preferred practices and new development for ASCII. For more aid, see the national and international standards given in Table 1 (I/A May 1978). Other detailed articles listed in the references^{3,4,5}.

STICKS 4-7

ASCII, a 7-bit code, is usually represented in 8 columns of 16 positions. The rows are 0000 through 1111

(0-15 decimal), the low-order 4 bits. Columns are 000 through 111 (0-7 decimal), the next higher 3 bits. ASCII developers called these columns "sticks". Each position is represented here by its usual decimal representation. E.g., capital A is in 4/1.

The first positions of sticks 4 and 6 are respectively the "commercial at" and "accent grave". Capital and small Roman letters follow. This 1-position offset is historical (from the UK) and unimportant as long as you remember it.

Beware the three positions following the alphabet in both sticks 5 and 7. In ASCII they are [\] in positions 5/11, 5/12, and 5/13, and { | } in 7/11, 7/12, 7/13. But the ISO and CCITT Codes reserve them for national usage. Table 2 (I/A May 1978) gives their national assignments. Surely you recall that Scandinavian alphabets have 29 letters, not 26? My friend Orjar Heen in Oslo is protective of these positions. He says, "If you Americans want to sell computers and software abroad, don't use the ASCII characters for these positions in your software."

4/0, "commercial at" of ASCII, is also a national usage position, but many nations have it in their primary sets.

Three diacritical marks — accent grave (`) in 6/0, circumflex (^) in 5/14, and tilde (~) in 7/14 — are also in national usage positions.

The circumflex causes confusion. Teletype made it an "up arrow" in an earlier version of ASCII, as an exponentiation symbol for BASIC. But it won't do, because FORTRAN uses a double asterisk! This is preferable in France, because they have such words as *crâne*, *côte* and *coût*, etc.

A problem also exists with the underscore in 5/15. It's neither national nor diacritical; all countries use it just as underscore (for typesetting it's a U.S. convention for italics, but in Italy it means bold-face, except under the last character in a line!). But Teletype's early version of ASCII used it as a "left arrow" — probably as an assignment symbol equivalent to := in ALGOL. The up and left arrow have been carried over from Teletype into many video terminals. Ask your manufac-

turer to cease and desist and retrofit. It's not ASCII and will cause trouble forever.

The last character is DElete in position 7/17. It's here because the binary code is 1111111, which would be all punched holes in perforated tape, and that's the only way to ensure that it can't be misread as some other character. ASCII is a complete set; all positions are assigned to have meaning.

STICKS 2-3

These are for digits and specials. "Digits" 0 to 9; not numbers, not numerals! They're in 3/0 through 3/9 so that the low-order 4 bits are the packed decimal representations. A special 4-bit set was considered for numerical applications, but computer hardware became inexpensive enough to not deprive ourselves of the capabilities of 7- and 8-bit sets.

For packed decimal, where two digits go into each 8-bit group, a word of caution on the plus and minus signs — they are in stick 2, rather than stick 3 with the digits. But the low order 4 bits are distinct, and + should be used only as 1011,

— only as 1101. I mention this because one nonstandard code EBCDIC permits multiple packed decimal representations of + and -. The ASCII representations are not even coincident with any of them, with obvious dangers!

The "currency" positions, 2/3 and 2/4, also have national variations. In ASCII they are # and \$, but remember:

- # is not "number sign" for many countries. Most use "No." or "Nr." for that purpose. When it is "number", it must precede the digits, not follow.

- # is "pound sign" only for the U.S. Others use kilograms. Use the abbreviation "lb." in the U.S. not #.

- The British "pound" has its own symbol, which is why it is in position 2/3 for the UK. Calling # a "pound" sign irks them, especially in software manuals.

- The "dollar" is peculiar to the U.S., Canada, and some others. There are also francs, marks, escudos, pesos, lire, etc. Which is why the ISO Code has a universal currency symbol in 2/4. It's a "scarab", a circle with outside spikes at 45, 135, 225, and 315 degrees.

The problem will get worse when we get into expanded character sets for photocomposition and such. For now, follow ASCII, which says that # is a "number sign".

A few more peculiarities remain. One is in the double quote, in 2/2, and the single quote, in 2/7. That is, you may think it's a single quote, and use it so, but it's really an "accent acute" for vowels. It slants top right to bottom left, to complement "accent grave" in 6/0, which slants top left to bottom right. Some terminal makers, not realizing this pairing, slant accent grave correctly, but put accent acute as a single quote in the unstylized up and down form. My Terminuset is OK.

Don't forget that to a typesetter, in contrast to typewriters, single and double quotes have two forms—opening and closing. In fact, the double quote is made from two single quotes, of either form, because the quote uses very little proportional space. Most video or hard-copy terminals use constant spacing. So double and single quotes must be distinct.

The last variant is the ampersand in 2/6, for which there are many legitimate designs. Neither ASCII nor the ISO Code prescribe a design. But this leads to the next topic—how to represent the ASCII characters in handprinted form so they may be input to computer systems.

HANDPRINTING FOR STICKS 2-7

The classical confusion for many years was between the digit zero and the letter "Oh", but there are other possibilities for mistakes. American Standard X3.45 specifies handwritten character shapes for ASCII, clearing up a longstanding problem. The communications types (and armed services) used to slash the zero. Somehow IBM users got to slashing the letter "Oh" instead, confusing the Scandinavians greatly. Now it's neither (which helps), just a 180-degree rotation of the letter O.

CAPITAL AND SMALL LOWER CASE LETTERS

Many people are accustomed to using upper case (capitals) only. This is a hangover from early line printers and limited (6-bit) sets. It might have been better if those smaller sets had been all lower case. Would you read a book if it were all in capitals? Small letters are much easier and faster to read, so lower case should be the default for single case. FORTRAN and BASIC processors should understand lower case variable names and verbs just as easily as capitals.

Get a terminal with both cases if at all affordable. Second best is making sure that a single-case terminal is retrofittable later. Get a single-case terminal in lower case only, if possible. The computer trade press has reported eyestrain from using computer terminals. Is the reason obvious?

ters, this relationship is the "collating sequence".

The ASCII standard used to say that the collating sequence for both graphics and control characters is defined simply by their binary representations. Later it added a warning that this sequence "cannot be used in many specific applications that define their own sequence." An understatement! The 1977 revision does no better.

It's not that difficult. Suppose you want to know how two files differ and/or are the same. The implied collating sequence (straight binary comparison) is fine. The two files will be in the same order, and can be matched.

Whether straight binary ordering can be used for any other purpose is doubtful. It won't work for signed numbers.

Ordering Numerals

If the sign precedes the digits, ordering by the ASCII collating sequence is worthless, because ordering is decided left to right, and - has a binary value 2 higher than +. Sign following the digits doesn't work either, because the complete decision is made in the leading digit.

To achieve a proper ascending sequence, separate the values into two groups, ordering those with plus signs in ascending sequence, and those with minus signs in descending sequence. Then put the plus group following the minus group. And vice versa for a total descending sequence. This works regardless of where the sign is.

Ordering Alphabetic Fields

Alphabetic ordering is more complex, particularly with both capital and small letters. Again the implied ASCII collating sequence can go wrong.

In the simplest case, two alphabetic items must be compared with the case ignored. Only if they are then equal is case called into consideration to break the tie, and it is also applied successively left-to-right!

In short, the capital and small versions of a letter do not both get full graphic significance. Typing either "y" or "Y" will indicate a "yes" reply, but "N" will not. Because the case distinction is minor, comparisons must first be made on major distinctions, with the minor distinctions used only as tiebreakers. Accenting of letters must also be considered minor, if accomplished via backspace, but this leads us into rules controlled by foreign governments, and won't be considered here.

Real life is even more complicated than this. The ordering and sequencing of characters and words can't always be accomplished by simple binary code comparison. There are constructions such as O'Reilly, l'Informatique, and Smith-Jones — to say nothing of the Juniors, Ills, Esq., etc. The New York Telephone Company's document on this problem runs several pages!

Note that the four information separators (FS, GS, RS, US) are designed to collate just behind Space, in that order. This contiguity means that they can be used as a hierarchy of spaces of different class.

ASCII AND PROGRAMMING LANGUAGES

Standard ECMA-53 gives the subsets and/or modifications of ASCII for APL, Minimal BASIC, COBOL, FORTRAN, and PL/I (Figures 1 through 5, I/A June 1978).

ASCII AND MEDIA

ASCII and Punch Cards

Table 1a (I/A June 1978) gives standards for ASCII on physical media. Figure 6 (I/A June 1978) defines hole patterns for the binary encodings. Figure 7 (I/A June 1978) defines encoding for the hole patterns. Nothing can be done now about the inconsistent relationships. It started with Herman Hollerith's first U.S. Census machines in 1890.

ASCII and Magnetic Tape

Figure 8 (I/A June 1978) represents several relationships, including the assignment of ASCII bit patterns to 9-track magnetic tape.

The jumbled assignment may remind you of the "firing order" for automobile engine cylinders. Despite better technology, data file investment won't let these assignments be changed.

Cassettes and cartridges use no parallelism in recording. The ASCII bits are recorded serially in the track.

ASCII and Communications

The topic of ASCII and communications is complex and large. It is undergoing substantial rethinking, enlargement, and invention.

Table 1b (I/A June 1978) lists many existing communication standards. More are under development. Arguments rage internationally on the merits of packet switching, byte protocols, added value, open-working systems, tariffs, data movement across national borders, X.25 protocol, etc. ATT is offering a new service because they suddenly discovered data-under-voice (DUV). All we can say now is that it is all based upon ASCII, and the proposed protocols are all dependent upon ASCII control characters. It will take years for this to shake out.

ASCII AND THE METRIC SYSTEM

The full ASCII graphic set (both cases) can represent all symbols and prefixes of the SI (International System of Units, the new metric system), except three — the Greek letters "omega" for "ohm", and "mu" for "micro", and the degree symbol for Celsius temperature. These will be provided in 8-bit ASCII. Meanwhile, for these, and also for equipment with only a single case, there is a standard way of representing SI units and prefixes. It's given in International Standard 2955 and American Standard X3.50.

The algorithm is: 1) If you have ASCII with both cases of alphabet, the three missing symbols are handled as "ohm", "u", and "Cel". 2) If you have only one case of alphabet (either upper or lower), "OHM", "U", and "CEL" are also permissible. Some other units get new symbols because case distinction is lost. E.g., S (siemens) and s (second).

ASCII AND KEYBOARDS

Technically, an ASCII keyboard must generate proper codes for the full set of ASCII graphic and control characters. Moreover, none of the graphic characters should have control properties.

There are many types of special keyboards — a two-sided one used like an accordion with the hands in a vertical plane, Touch-Tone and its derivatives, Dvorak, etc. There are no formal standards to relate these keyboards to ASCII. For typewriter-style keyboards, however, two versions given in the American National Standard. One is derived from the usual electric typewriter keyboard, and is up for extensive change due to growth in Word Processing. The other is called the "bit-paired" keyboard. Shown in Figure 9 (I/A June 1978), it is the keyboard of ISO Standard 2530.

Control and Function Keys

The so-called "QWERTY" arrangement is prevalent throughout the Anglo-Saxon world. Even the French "AZERTY" set is being considered for change. But there are hundreds of other keyboard variations. Some have "dead keys" (i.e., the platen or printing element is not advanced when they are hit). This avoids having to use BS for accented letters, but it also creates difficulties in code generation.

There are some general good practices for ASCII keyboards. For ease of use by those experienced with typewriters, all controls not used with typewriters should be located outside the customary touch-typing area. As a specific example, the Break/interrupt key should be located where it takes a definite effort to reach it (not mixed in with the keyboard). Consult ISO 3244 for these considerations.

Function Keys generate sequences of more than one ASCII character. Examples are cursor keys, Erase-to-EOL, etc. They should be located in special clusters. Most importantly, they must all generate ASCII codes for transmission when

2. source voltage is common to all circuit branches.

Continued from Page 5

in character-at-a-time mode. I know of video terminals where the cursors do not generate codes, as they should not while in full page buffered mode; but they still operate in the line mode without generating codes. In this case the screen is alterable, but there is no way of detecting it in the computer.

Many keyboards will have some unlabeled function keys, for do-it-yourself assignment. These should also be clustered separately, and generate code sequences when in line mode.

ASCII AND DISPLAY/PRINTING

When ASCII characters are displayed, it may be on a video screen, paper, or COM (microfiche).

On the video screen there are a number of methods to form the characters, mostly at the manufacturer's preference. They are usually at pica (constant-width) spacing for economy, so graphic quality (such as typesetting) can't be approximated. When lower case is available, the risers and tails extend above and below the line for some screens. In others, they fall within the boundary lines of the upper case characters. They may be shown in inverse video (light background block), or highlighted by differing brightness or blinking. Controls for this work will be discussed later.

For paper copy one usually finds either direct impact of a formed letter, or stylus printing. Either method is suitable for proportional spacing if desired. Recently there has been a general trend towards using the 7x9 dot matrix shapes of ECMA Standard 42 for stylus printers, as shown in Figure 10 (I/A June 1978).

For hard print elements one can get a nearly infinite variety of styles and fonts. Only two, however, are

value of the ordinates during the positive alternation. Average value is not as widely used as the RMS value, but in some instances is more a descriptive value of current or

specifically for computers — OCR-A and OCR-B. "OCR" stands for "Optical Character Recognition", meaning that the shapes are so styled that a computer-controlled scanner can read the characters as printed on paper, encoding them directly from their shapes.

OCR-A is not suitable for human reading. It's the funny looking one with the diamond-shaped letter "Oh". With today's technology there is no need to use anything other than OCR-B, as specified in ISO 1073/2, ECMA-30, and ANSI X3.49, and shown in Figure 11 (I/A June 1978). I have it on my IBM golfball typewriter at home, and on my daisywheel element at the office. So it should be available for most hard elements, including carousel.

CODE EXTENSION — GENERAL PRINCIPLES

ASCII is the basis for codification of the various symbols used throughout the world. Through it, libraries can store encoded versions of printed books. And while electronic mail may be quite simple with ASCII and its Roman alphabet, that's not the alphabet of all countries. The USSR uses Cyrillic, the Japanese use Katakana, and the Arab world uses its own semi-script alphabet. Moreover, to send a mathematics textbook by electronic mail we must encode the formulas and special symbols peculiar to mathematics, which includes many Greek characters!

These is where the ESCape character and ESCape sequences come in. You can get the whole complicated story from ISO Standard 2022 (or ECMA-35) on Code Extension Procedures. Think of reproducing many ASCII Code Tables on the pages of a book, then replacing the ASCII symbols on all but the first page with the

in time. Thus the voltage e_1 is said to lead e_2 by 60° electrical degrees. Another way of saying this is that e_2 lags e_1 by 60° electrical degrees.

The concept of phase difference

other alphabets we need.

CODE EXTENSION REGISTRY

Then make sure that everyone in the world has the same (code) book. (The resemblance to military code books is intentional.) That's done by registering the page number assignment to characters (either a control set or a graphic set, but not both) with the French Standards Body AFNOR — Association Francaise de Normalisation, Tour Europe Cedex 7, 92080 Paris La Defense, FRANCE. The registry set is available from AFNOR for approximately 172 French francs, say \$35. It would be vital for an equipment or software manufacturer to have it. You'll find it perhaps easier to get from ANSI.

The registration procedure is ISO Standard 2375. 19 graphics sets and 3 control character sets are registered so far. Once registered, these assignments may never be changed!

ALTERNATE CONTROLS

Work has been in progress for several years to develop a companion standard for controls for devices such as CRT terminals. In the U.S. this is contained in the ANSI document BSR X3.64, Additional Controls for Character Imaging. In a similar form, this set is before ISO Technical Committee 97 (Computers and Information Processing) as document 2 N 868.

As one example of how these controls operate, the 4-character string ESC [_6_m when encountered, should turn on rapid blink in the field(s) specified on your video screen.

CODE EXPANSION

ASCII has been *extended* by making many related pages of the 7-bit code. It is also possible to *expand* ASCII into an 8-bit code, or even 9-bit and 10-bit if we wished, for that matter. But an 8-bit code is obvious-

ly the most logical one to concentrate on, and this has been under development for several years. Figure 5 (I/A July 1978) is the proposed 8-bit Expanded ASCII Code.

—editor

FUTURE FOR ASCII

Methods are in place for codifying all symbols that people use. They may be language alphabets, signs, drawing symbols, or controls for equipments. Robots, for example. Satellites are augmenting conventional telecommunications systems, so that one can borrow cheaply and permanently from electronic libraries.

To prepare for this, other sets are being developed for registry, many through ISO Technical Committee 46/1, Automated Documentation. A 2-page mathematical symbol set is near submission, as are African sets. Work is started for Arabic, which will take about 5 sets to handle fully, although there is a commercial subst of 94 graphics. A set is proposed for bibliographic controls. Other control sets can come from process control, animation and other graphics applications, etc.*

REFERENCES

1. ANS X3.4-1977, available from the American National Standards Institute, 1430 Broadway, New York, NY 10018. Ask for a brochure on Information Processing Standards.
2. ISO 646, available from ANSI (Ref. 1).
3. R.W. Bemer, "ASCII — the data alphabet that will endure", in *Management of data elements in information processing*, National Bureau of Standards, 1975 Oct., 17-22.
4. R.W. Bemer, "A view of the history of the ISO character code", *Honeywell Computer J.* 6, No. 4, 1972, 274-282.
5. E.H. Clamons, "Character codes: who needs them?", *Honeywell Computer J.* 5, No. 3, 1971, 143-146.

INTERFACE AGE™

COMPUTING FOR HOME AND BUSINESS APPLICATIONS

VOLUME 4, ISSUE 2 FEBRUARY 1979 \$2.00
CANADA/MEXICO \$2.50 INTERNATIONAL \$3.50

91

Business and Computers
Financial Report Writer
Multitasking the 8080
Time and the Computer



NTS mini series **Unit #1 Continued**
Programming Technique #2
The Micro Mathematician



INTERFACE AGE™

COMPUTING FOR HOME AND BUSINESS APPLICATIONS

GENERAL FEATURES	
MULTI-TASKING THE 8080: FAMOS, AN ON-LINE SYSTEM	54
<i>by George Phillipovich, MVT Microcomputers</i>	
A SIMPLE FINANCIAL REPORT WRITER	64
<i>by Fred LaPlante</i>	
BUSINESS AND COMPUTERS: THE PERFECT MARKET	70
<i>by Terry Costlow, Assistant Editor</i>	
TIME AND THE COMPUTER	74
<i>by R.W. Bemer, Contributing Editor</i>	
BUSINESS FEATURES	
BUSINESS EDITORIAL — MICROCOMPUTING AND ACCOUNTING	80
<i>by Phillip N. Payne, Payne, Jackson and Associates</i>	
NOTARY	81
<i>by Jim Schreier</i>	
NEW TERMINAL, NEW IDEA	84
<i>by Carl Warren, Editor-in-Chief</i>	
THE REAL WORLD OF MICROCOMPUTING	86
<i>by Mathew Tekulsky</i>	
MANAGING LAYAWAY ACCOUNTS	88
<i>by Ray Vukcevic</i>	
HARDWARE FEATURES	
CARD OF THE MONTH: MICRONICS INC. — BETTER BUG TRAP	92
<i>by Roger H. Edelson, Hardware Editor</i>	
Z-80: IS IT REALLY BETTER?	94
<i>by Rod Hallen</i>	
NTS MINI SERIES OF BASIC ELECTRONICS	98
<i>by Walter L. Stephens</i>	
SOFTWARE FEATURES	
PROGRAMMING TECHNIQUE — NIGHT 2	126
<i>by Bill Turner, Senior Editor Southeast Region</i>	
BURN YOUR OWN PROMS: BUILDING AND REBUILDING A PROM PROGRAMMER	130
<i>by Alan R. Miller, Contributing Editor</i>	
MACROPROCESSING EASES THE SOFTWARE DEVELOPMENT TASK	136
<i>by Henry Davis, American Microsystems, Inc.</i>	
IN-LINE MACHINE LANGUAGE SUBROUTINES FOR BASIC INTERPRETERS	140
<i>by John P. Newcomer</i>	
FREE RUNNING	
BUSINESS SOFTWARE REVIEW	48
THE COLUMN	15
EDITOR'S NOTEBOOK	4
FROM THE FOUNTAINHEAD	26
INVENTOR'S SKETCHPAD	53
JURISPRUDENT COMPUTERIST	29
LETTERS TO THE EDITOR	6
MICRO MATHEMATICIAN	41
MICRO MEDICINE	32
MIND REVOLUTION	50
WHITE COLLAR MICROCOMPUTER	45
DEPARTMENTS	
ADVERTISER INDEX	144
CALENDAR	21
FIFO	144
MICRO MARKET	142
NEW PRODUCTS	114
UPDATE	17

THIS MONTH'S COVER
Photography by Shelley Wright. Terminal, disk pack, computers, and printer courtesy of Byte Shop, Westminster, California. Produced by B & P Associates.

EXECUTIVE ADMINISTRATION

PUBLISHER **ROBERT S. JONES**
EXECUTIVE PUBLISHER **RANCY A. JONES**
ASSISTANT TO THE PUBLISHER **JOANNA KONDRATH**
GENERAL MANAGER **EVA YAKA**

ADMINISTRATION

PUBLICATION DIRECTOR **MIKE ANTICH**
PUBLICATION ASSISTANT **DENISE JACKSON**
SUBSCRIPTION CIRCULATION **JO ANN FERBUSON**
CIRCULATION ASSISTANT **CHARLOTTE SEVEDGE**
CIRCULATION SECRETARY **TONI DOTY**
ACCOUNTING ASSISTANT **SAYOKO TANISAKI**

EDITORIAL

EDITOR IN CHIEF **CARL WARREN**
SENIOR EDITOR SOUTHEASTERN REGION **BILL TURNER**
ASSISTANT EDITOR **TERRY COSTLOW**
NORTHWESTERN REGIONAL EDITOR **ADAM OSBORNE, PhD**
NORTHEASTERN REGIONAL EDITOR **ROGER C. GARRETT**
HARDWARE EDITOR **ROGER EDELSON**
CONTRIBUTING EDITOR **ALAN R. MILLER, PhD**
CONTRIBUTING EDITOR **R. H. DISTLER**
CONTRIBUTING EDITOR **R. W. BEMER**
CONTRIBUTING EDITOR **SANDRA EVANS**

Editorial Correspondence

Direct all correspondence to the appropriate editor at: INTERFACE AGE Magazine, P.O. Box 1234, Cerritos, CA 90701.

PRODUCTION

PRODUCTION MANAGER **MARGARET FENSTERMAKER**
ASSISTANT PRODUCTION MANAGER **SHELLEY WRIGHT**
ART DIRECTOR **PHIL ORTIZ**
ARTIST **SAMANTHA LEE**
TYPOGRAPHER **MELODY A. MARTENS**

ADVERTISING

NEW ENGLAND REGION **DICK GREEN**
7 Lincoln St., Wakefield, MA 01880 (617) 245-9100
EASTERN REGION **TONY CARLSON**
28 Community Pl., Morristown, NJ 07960 (201) 267-3032
MIDWEST REGION **AL GRAVENHORST & STEVE SKINNER**
1801 N. Cicero Ave., Chicago, IL 60648 (312) 545-6621
WESTERN REGION **BRUCE BERRY & ZACH BONNETTE**
81 S. Lake Ave., Pasadena, CA 91106 (213) 799-7002
COMPUTER RETAIL STORES NATIONWIDE
CALL (213) 799-7002 (COLLECT)

INTERFACE AGE EUROPE

HANS-JÜRGEN GROHMANN
Bismarckstr. 82, 1001 Untermythen, West Germany
Telephone: 8167-368

FOREIGN CIRCULATION

JAPAN CIRCULATION **KAZUHIKO NISHI**
ASCI Publishing, 305 W. TORIO, 5-4 Minami-Aoyama, Minato-ku, Tokyo 107 Japan
Telephone: (03) 467-4919
UNITED KINGDOM CIRCULATION **VINCENT COEN**
L.P. Enterprises, 213 Kingston Road, Hove, Essex, England (0313)
Telephone: 01553-1001
FRANCE CIRCULATION **ROLAND HESSE**
Euro Computer Shop Paris, 16, Rue Louis Pasteur, 92100 Boulogne, France
Telephone: Paris 823-8252
WESTERN CANADA CIRCULATION **BRIAN L.I. WIEBE**
Kempco, 25236 26th Avenue RR5, Abbotsford, B.C. V0X 1A0
Telephone: (604) 856-2301
EASTERN CANADA CIRCULATION **LEZ JANEK**
86-232, Ltd., 186 Queen Street West, Toronto, Ontario M5V 1Z1
Telephone: (416) 598-0288
AUSTRALIA CIRCULATION **R. J. HOESS**
Electronic Concepts Pty. Ltd., 52-58 Clarence Street, Sydney NSW 2000
Telephone: 29-2753
INDONESIA, SINGAPORE, MALAYSIA CIRCULATION **LEE MILES**
The Computer Centre, Pte. Ltd. 5388, 8th Floor Complex, Beach Rd. Singapore F
Telephone: 283-2630

MEMBER OF THE WESTERN PUBLICATIONS ASSOCIATION



AMERICAN SOCIETY OF BUSINESS PRESS EDITORS

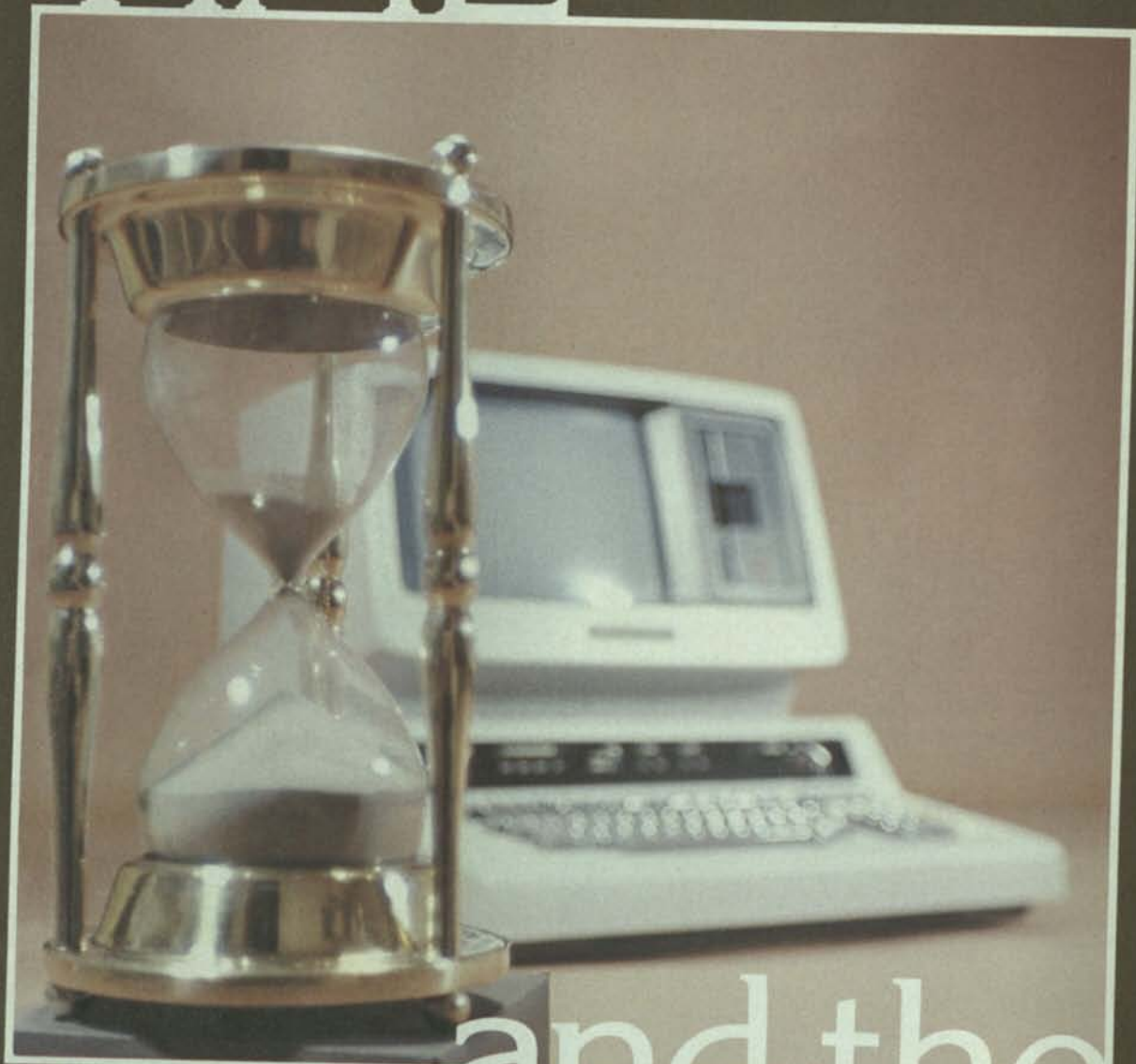


INTERFACE AGE Magazine, published monthly by McPheters, Wolfe & Jones, 16704 Marquardt Ave., Cerritos, CA 90701. Subscription rates: U.S. \$18.00, Canada/Mexico \$20.00, all other countries \$28.00. Make checks payable in U.S. funds drawn on a U.S. bank. Opinions expressed in by-lined articles do not necessarily reflect the opinion of this magazine or the publisher. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsement of the product or products by this magazine or the publisher.

INTERFACE AGE Magazine COPYRIGHT © 1978 by McPheters, Wolfe & Jones. ALL RIGHTS RESERVED. Material in this publication may not be reproduced in any form without permission. Requests for permission should be directed to Joanna Kondrath, Rights and Permission, McPheters, Wolfe & Jones, 16704 Marquardt Ave., Cerritos, CA 90701. INTERFACE AGE Magazine is catalogued in the Library of Congress, Classification No. QA75.5.155, ISSN Publication No. 0147-2992. Membership in Audit Bureau of Circulations applied for.

POSTMASTER: Please send change of address form 3579 and undelivered copies to INTERFACE AGE Magazine, 16704 Marquardt Ave., Cerritos, CA 90701. Second-class postage paid at Artesia, California 90701 and at additional mailing offices.

TIME



and the COMPUTER

By R.W. Bemer, Contributing Editor

Knowledge and use of time is an essential part of computer usage. In payroll programs, for example, so we can get paid periodically. In control programs, as for rockets. And not least in measuring efficiency of the programs themselves.

I'm going to give here all time standards and other aspects that I'm aware of. But first a few warnings:

- Don't embed time characteristics in program workings. There are many horror stories about programs, working for years, that died on some significant change in the date. The original programmer had moved on, of course, and left little documentation. For example, suppose in the 1970s the tens position of the year (i.e., 7) was used for the number of days in the week. Works OK then, but not in the 1980's. And year 2000 isn't far away.
- Don't write programs with provincial time elements. Use international standards, and make them useful for many languages. References and tables for these matters are at the end of this article.
- If from lack of knowledge you cannot program for all conditions, at least put in a stop and display wherever you are unsure.

BASICS OF TIME

The second is one of the seven basic units of the SI (International Systems of Units — the metric system). It may be abbreviated to "sec.", but never use the abbreviation as a symbol. The symbol is "s". E.g.:

3 Gs (3 gigaseconds, or 3 000 000 000 seconds)

It's perfectly possible to do all time calculations with the second only, but it doesn't serve people very well. So we go to calendar time, for which a brief resume is given here:

	second	minute	hour	day	week
second	1				
minute	60	1			
hour	3600	60	1		
day	86400	1440	24	1	
week	604800	10080	168	7	1

After this every relationship is arbitrary. The fortnight might appear logical to the U.S., but to the Spanish it's 15 days, not 14. Perhaps to halve the month more closely.

FORMATS FOR DATE

When asked the time, do you reply 4:30 (four thirty) or 30:4 (thirty four)? 4:30, of course. The larger unit (hour) precedes the smaller (minute). That's the rationale for International Standard 2014¹ and American National Standard X3.30².

The American way has been "month day, year", while the American military and most of the rest of the world used "day month year", not needing a comma. Spelled out, confusion between the two forms was minimal. In digits only, as for a postmark or inside a computer, the confusion was intense, and costly for international trade. Fortunately for compromise, neither form had logic on its side. The international and American standards give time in a sequence that descends to the right:

calendar year - month - day - hour - minute - second

business year - fiscal week - day - hour - minute - second

ISO (International Standards Organization) had these reasons for the descending sequence:

- Ease with which the whole date may be used as a single numeral for the purpose of filing and classification (e.g., for insurance or social security systems);

- Arithmetical calculation, particularly in some computer applications;
- The capability of continuing the order by adding digits for hour-minute-second (subject to the differences in world time zones).

If it is required to separate the date elements for display, the separator is preferably a hyphen (or a space, for second choice), but not the slash (stroke) or point. E.g.:

1979-01-24, not 1971/01/24 or 1971.01.24

In particular, don't drop the first two digits for computer processing, unless you take extreme care, remembering that it's only the "year of the century". Otherwise the program may fail from ambiguity in the year 2000.

COMPARING TWO DATES

The American Standard recommends representing the date in computer store without separators, e.g. 19790124. This makes it easier to compare two dates (which is earlier/later). A simple subtraction does it. Finding the difference between two dates in (elapsed or working) days is more complicated, due to variations in both years and months.

LEAP YEARS

The leap year device is used periodically to adjust the synthetic calendar year to the real astronomical year. Every year divisible by 4 is a leap year, except if it's divisible by 100 but not 400. So 1800 and 1900 were not leap years, but 2000 is. Finer adjustment is coming. Years divisible by 4000 won't be leap years (may your programs run that long). You won't have trouble unless you are computing backward in time, in which case pay particular attention to 1582, which is when the Gregorian calendar superseded the Julian, with a resultant loss of 10 days from October 5 to 15.

Vector strings for the individual and cumulative days in the months (12 per year, of course) are:

031028031030031030031031030031030031
029

031059090120151181212243273304334365
060091121152182213244274305335366

A simple method of calculating whether a year is/isn't a leap year is embedded in Robertson's algorithm³. It uses the integer arithmetic found in FORTRAN, TEX, and some other programming languages. The entry variable "i" is the 4-digit year:

```
a=(i-i/4*4+3)/4      (0 if divisible by 4, else 1)
b=(i-i/100+100+99)/100 (0 if divisible by 100, else 1)
c=(i-i/400+400+399)/400 (0 if divisible by 400, else 1)
leap=1-a+b-c          (1 if "i" is Leap year, else 0) (1)
```

Thus (1-a + b-c) has a value 1 (1-0 + 1-1) for regular leap years, and for 1600, 2000, 2400, etc. (1-0 + 0-0). But it's 0 (1-0 + 0-1) for 1700, 1800, 1900, etc., and all other years (1-1 + 1-1). I have a feeling this one can be bettered.

It's easier in TEX, remainder being an active function. Here the variable "a" contains "if *rmdr:eq:0 leap", and "|" is the substitute character:

```
leap=0 q=i/4 |a|=1 q=q/25 |a|=0 q=q/4 |a|=1 q=q/10 |a|=0 (2)
```

This one handles the year 4000 case, falling through to the next statement 3 out of 4 times without further computation!

ORDINAL DATES

American Standard X3.30 allows replacement of month and day by a 3-digit ordinal day of the year. So does ISO 2711², a separate standard. The date shown before would be:

1979-024 (1979024 internally)

This form is easier to difference, particularly within the same year. Only the leap year factor remains. To compute ordinal day we use the rest of Robertson's algorithm, with two more inputs — month "j" and day "k", 2 digits each. (j + 10)/13 has a value 0 for January and February, else 1.

$$iday = 3055 * (j+2) / 100 - (j+10) / 13 * 2 - 91 + (1-a+b-c) * (j+10) / 13 + k \quad (3)$$

CONVERTING ORDINAL TO CALENDAR DAY

Here we use my modification of an algorithm by Stone¹⁷. The original has a note that a patent was applied for, but the odds are it didn't make it.

The value of "leap" is computed as shown previously, with inputs ordinal day "iday" of year "i" (4 digits). Outputs are month "j" and day "k":

$$\begin{aligned} id &= iday + ((405 + iday - Leap) / 365) * (2 - Leap) & (4) \\ j &= ((id + 91) * 100) / 3055 - 2 & (5) \\ k &= id + 30 - (j * 3056) / 100 & (6) \end{aligned}$$

JULIAN DAYS AND DIFFERENCING

There are many methods of computing the number of days between two days, but the simplest and most reliable is to express any date in the Julian form. No relation to the Julian Calendar, this form has but one unit — the day (just as the metric system has the second as a linear unit).

The best algorithm is the classic of Fliegel and Van Flandern¹⁸. It also uses integer arithmetic without logic. The leap year algorithm is built in; the constant 1461 is the number of days in a 4-year cycle. Inputs "i", "j", and "k" are YYYY, MM, and DD respectively:

$$\begin{aligned} jd &= k - 32075 + 1461 * (i + 4800 + (j - 14) / 12) / 4 \\ jd &= jd + 367 * (j - 2 - (j - 14) / 12 * 12) / 12 \\ jd &= jd - 3 * ((i + 4900 + (j - 14) / 12) / 100) / 4 & (7) \end{aligned}$$

E.g., 1979-01-01 is Julian Day number 2443875. To difference two dates, compute and difference their Julian Day values.

An optional method, if ordinal day is needed anyway, is to compute the Julian day for the first day of the year. Substituting values of 1 for both j and k in the general expression, we get:

$$\begin{aligned} jd1 &= 1461 * (i + 4799) / 4 - 31738 - 3 * ((i + 4899) / 100) / 4 & (8) \\ jd &= jd1 + ord - 1 & (9a) \\ ord &= jd - jd1 + 1 & (9b) \\ jdord &= 1461 * (i + 4799) / 4 + ord - 31739 - 3 * ((i + 4899) / 100) / 4 & (10) \end{aligned}$$

As a curiosity, Julian Day 0 is found for Nov 24 of the year -4713 (yes, minus!), but that's not what the calendar said then. And if you go into more detail with Julian days, remember that they start at noon Greenwich Time.

FISCAL WEEKS

A companion standard, ISO 2015⁸, recognizes the calendar week as an important unit for purposes of planning and accounting, as for delivery dates. So the numbering of weeks has increasing use in international trade and industrial planning. Monday is specified as the first day of the week for business and commercial (not religious) purposes.

Hebdomadal, or septenary, years have an integral number of fiscal weeks. For these, a week divided by the turn of the year belongs to the year having the larger number of days that week. Equivalently, the year of that week's Thursday. Thus some hebdomadal years have 53 fiscal weeks. If based on calendar year, they can start anywhere from December 29 to January 4. Some people try to form fiscal months, of 4 or 5 weeks each, by the same dividing rule. Don't believe it!

Fiscal years are arbitrary, but hebdomadal calendar year is often chosen. In the remainder of the article it's called an "F-year". Figure 1 shows all possible F-year patterns. They're the same 14 of the perpetual calendar business, only the F-years are shown boxed. They're renumbered so that the fiscal constant "fc", which we

must calculate, is the number plus 3. Note the 53 weeks in years 7N, 6L, and 7L.

		Non-Leap Years							Leap Years							
No.	fc	M	T	W	T	F	S	S	M	T	W	T	F	S	S	
						01	02	03						01	02	03
1)	4	04	05	06	07	08	09	10	04	05	06	07	08	09	10	
		27	28	29	30	31	01	02	26	27	28	29	30	31	01	
						01	02							01	02	
2)	5	03	04	05	06	07	08	09	03	04	05	06	07	08	09	
		26	27	28	29	30	31	01	25	26	27	28	29	30	31	
						01	02							01	02	
3)	6	02	03	04	05	06	07	08	02	03	04	05	06	07	08	
		25	26	27	28	29	30	31	24	25	26	27	28	29	30	31
		01	02	03	04	05	06	07	01	02	03	04	05	06	07	
4)	7	24	25	26	27	28	29	30	23	24	25	26	27	28	29	30
		31							30	31						
		31	01	02	03	04	05	06	31	01	02	03	04	05	06	
5)	8	23	24	25	26	27	28	29	22	23	24	25	26	27	28	
		30	31						29	30	31					
		30	31	01	02	03	04	05	30	31	01	02	03	04	05	
6)	9	22	23	24	25	26	27	28	21	22	23	24	25	26	27	
		29	30	31					28	29	30	31	01	02	03	
		29	30	31	01	02	03	04	29	30	31	01	02	03	04	
7)	10	21	22	23	24	25	26	27	20	21	22	23	24	25	26	
		28	29	30	31	01	02	03	27	28	29	30	31	01	02	

Figure 1. Possible F-Year Patterns

CONVERTING ORDINAL TO FISCAL

Because some days of the calendar year may lie in either the previous or the following F-year, we need first an algorithm to determine which of the 14 types the year is. It's convenient to determine a constant "fc", such that:

$$\begin{array}{rccccc} \text{Fiscal Day of Jan 01} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ & fc & 7 & 8 & 9 & 10 & 4 & 5 & 6 \end{array}$$

First Julian Day for Jan 01 is derived from (8), and then:

$$\begin{aligned} t &= jd1 - jd1 / 7 * 7 \\ fc &= t + 7 - (t + 3) / 7 * 7 & (11) \end{aligned}$$

Now the fiscal date is computable:

$$\begin{aligned} fw &= (iday + fc - 1) / 7 & (12) \\ fd &= (iday + fc - 1) - 7 * fw + 1 \quad (=*rmdr+1 \text{ in TEX}) & (13) \end{aligned}$$

F-year "fy" is first assumed to be the same as calendar year "i". But:

$$\begin{aligned} \text{if } fw &= eq:53 \text{ if } (fc + Leap) < :lt:10 \text{ fy} = i + 1 \text{ fw} = 1 & (14) \\ \text{if } fw &= eq:0 \text{ fy} = i - 1 \text{ if } (1) \text{ fw} = 53 - (fc + 1 - Leap) / 6 & (15) \end{aligned}$$

In (15), F-year is 1 less than the calendar year, so formula (1) must be used to see if it is a leap year.

CONVERTING FISCAL TO ORDINAL

Assuming calendar year "i" to be the same as fiscal, formulas (8) and (11) give its fiscal constant. Then:

$$iday = 7 * fw + fd - fc \quad (16)$$

"leap" is found from (1), and:

$$\begin{aligned} \text{if } iday &= gt:(365 + Leap) \text{ } i = i + 1 \text{ } iday = iday - 365 - Leap & (17) \\ \text{if } iday &= lt:1 \text{ } i = i - 1 \text{ } iday = 365 + Leap + iday & (18) \end{aligned}$$

CONVERTING FROM JULIAN DAY

Fliegel and Van Flandern¹⁸ found going from Julian to calendar a bit complicated. Here it is. Going from Julian to ordinal should be much simpler:

```

l=jd+68569
n=4*L/146097
L=L-(146097*n+3)/4
i=4000*(L+1)/1461001
L=L-1461*i/4+31
j=80*L/2447
k=L-2447*j/80
l=j/11
j=j+2-12*L
i=100*(n-49)+i+l

```

(19)

RECAPITULATION

Now our four representations for date are compared:

Unit	Abbr.	Type	Representation
Day of time	DOT	Julian	XXXXXXX
Day of week	DOW	Fiscal	FFFFWWD
Day of month	DOM	Calendar	YYYYMMDD
Day of year	DOY	Ordinal	YYYY000 (or III or ORD)

Fiscal year is represented by FFFF because it's not always equal to YYYY, calendar year. Internal format problems are possible. Three of the representations have 7 characters. For some time the Julian day number will begin with 24.... (If you wish, Smithsonian Day brings it to 6 characters by subtracting 2000001.) But is 1979011 day 1 of fiscal week 1, or ordinal day 11? We could embed separators, as 1979.01.1. I like prefacing date fields with a letter. A same date is:

```
J2444239 F1980011 C19791231 01979365
```

Harry White of the National Bureau of Standards (chairman of both American and International committees on standards for data representation) suggests omitting the "C" prefix in the calendar date. It is thus a default case, being a calendar date when the first character is a digit, not a letter. This permits a common field size of 8 characters. I like it.

Now the sequences of numbered formulas for conversion are summarized. For example, "F-J" means the conversion from fiscal to Julian. When no direct conversions are given, the intermediate conversions are indicated:

C-0	(1)(3)	F-C	f-o _o -c
C-F	c-o _o -f	F-0	(8)(11)(16)(17)(18)
C-J	(7)	F-J	f-o _o -j
O-C	(1)(4)(5)(6)	J-C	(19)
O-F	(8)(11)(12)(13)(14)(15)	J-0	j-c _c -o
O-J	(8)(9a)	J-F	j-c _c -f

THE YEAR AND SHORTER PERIODS

Pay rate is an interesting topic. As rigor stops with the week, salary or other stipend based upon a week, day, or hour gives no problem. Monthly salaries would be rather unfair, due to the variation of up to 3 days, but yearly salaries are common. The question is how often a portion of the salary is paid. There is no problem with calculating a monthly portion — just divide by 12, even though the recipient has to adjust a little. But paying every week or two weeks gives a rate problem.

One solution is to have regular years have 52 1/7 (52.143) weeks or 26 1/14 (26.0714) 2-week periods, and leap years to have 52 2/7 (52.286) weeks or 26 1/7 (26.143) 2-week periods. Or if the employee is expected to stay a long time, constant values of 52.1786 or 26.0893 may be used.

LOCAL TIME OF DAY

Except for the military again (and some airlines), the United States has used 12-hour time (clocks) much more than 24-hour time. Digital clocks will help the trend to 24-hour time. Any problems?

Yes. 12:59 is less than the 24 hours in a day. We've had no problem referring to 12:59 AM and 12:59 PM, followed by 01:00 AM and PM respectively. When we add 12 hours to all those after noon, 12:59 is followed by 13:00, but is 23:59 followed by 24:00 and then 24:01 when there are only 24 hours in a day? It seems logical that one minute after midnight should be 00:01, but is midnight 00:00 or 24:00?

This was one of the major difficulties in trying to settle on standards for time^{10,11,14}. The result is that there are six permissible ways to express time of day numerically:

1. HH
2. HH.hhh--h
3. HHMM (optionally HH:MM)
4. HHMM.mmm--m (optionally HH:MM.mmm--m)
5. HHMMSS (optionally HH:MM:SS)
6. HHMMSS.sss--s (optionally HH:MM:SS.sss--s)

The optional colon spacer gives no logical problem. If not encountered in the third position from the left, it has not been used. If a decimal point is encountered there, the field is decimal hours only. If a decimal appears in the 5th position from the left, the field is hours and decimal minutes. If in the 7th, it is hours, minutes, and decimal seconds. In all three cases, the decimal part is of variable length.

MM and SS have values from 00 to 59, HH from 00 to 23, except that midnight may be 24 in Form 1 only. All 12-hour time forms are followed (no space between) by a meridiem designator, "A" or "P". Some examples in both 12-hour and 24-hour forms are:

Form	12-hour	24-hour	
1	02P	14	
3	12:00P	12:00	noon
5	120000	120000	noon
3	12:21P	14:21	
1	12A	00	midnight
1	12A	24	"
3	12:00A	00:00	"

Date and local time of day may be stored in a single field. Examples:

19791231-235958	(2 seconds to midnight)
19791231-235959	(1 second to midnight)
19800101-000000	(Auld Lang Syne)
19800101000000	(")

Although the example just shown was from the standard, I think it better to actually store the colon(s) and decimal point in the time format. Otherwise one must program their insertion for display. Two times can't be compared by subtraction, as two dates can be, so there's no point to it.

Note:

Midnight is always 00 .. in the ISO Standard. The American Standard permitted "24" to accommodate certain applications of the Department of Defense, but don't you use it! It works only for time accurate to one hour, and it violates other computer standards. It's like having digits 0-1-2-3-4-5-6-7-8-9-X, and you could use either X or 10 at your option.

UNIVERSAL TIME

American National Standard X3.51¹³ relates U.S. time zones to universal time (Greenwich Meridian) and specifies the representations to do so. Time may be specified as universal, in which case 12-hour clock time is followed by A or P (for A.M. or P.M.), a space, and "GMT"; while 24-hour clock time is followed directly by "Z", for Zero time. E.g.:

```
07:09:23P GMT 190923Z
```

Or it may be specified as local, with either a time differential factor (TDF), or a time zone reference. E.g.:

```
140923-0500 140923 EST 02:09:23P EST
```

English	German	Swedish	French	Italian	Spanish	Portuguese
Monday	Montag	måndag	lundi	Lunedì	lunes	segunda-feira
Tuesday	Dienstag	tisdag	mardi	martedì	martes	terça-feira
Wednesday	Mittwoch	onsdag	mercredi	mercoledì	miércoles	quarta-feira
Thursday	Donnerstag	torsdag	jeudi	giovedì	jueves	quinta-feira
Friday	Freitag	fredag	vendredi	venerdì	viernes	sexta-feira
Saturday	Samstag	lördag	samedi	sabato	sábado	sábado
Sunday	Sonntag	söndag	dimanche	domenica	domingo	domingo
January	Januar	januari	janvier	gennaio	enero	janeiro
JAN	JAN	JAN	JAN	GEN	ENE	JAN
February	Februar	februari	février	febbraio	febrero	fevereiro
FEB	FEB	FEB	FEV	FEB	FEB	FEV
March	März	mars	mars	marzo	marzo	março
MAR	MRZ	MAR	MAR	MAR	MAR	MAR
April	April	april	avril	aprile	abril	abril
APR	APR	APR	AVR	APR	ABR	ABR
May	Mai	maj	mai	maggio	mayo	maio
MAY	MAI	MAJ	MAI	MAG	MAY	MAI
June	Juni	juni	juin	giugno	junio	junho
JUN	JUN	JUN	JUN	GIU	JUN	JUN
July	Juli	juli	juillet	luglio	julio	julho
JUL	JUL	JUL	JLT	LUG	JUL	JUL
August	August	augusti	août	agosto	agosto	agosto
AUG	AUG	AUG	AOU	AGO	AGO	AGO
September	September	september	septembre	settembre	septiembre	setembro
SEP	SEP	SEP	SEP	SET	SEP	SET
October	Oktober	oktober	octobre	ottobre	octubre	outubro
OCT	OKT	OKT	OCT	OTT	OCT	OUT
November	November	november	novembre	novembre	noviembre	novembro
NOV	NOV	NOV	NOV	NOV	NOV	NOV
December	Dezember	december	décembre	dicembre	diciembre	dezembro
DEC	DEZ	DEC	DEC	DIC	DIC	DEZ
spring	Frühling	vår	printemps	primavera	primavera	primavera
summer	Sommer	sommar	été	estate	verano	verão
fall	Herbst	höst	automne	autunno	otoño	outono
winter	Winter	vinter	hiver	inverno	invierno	inverno
A.D.	P. Chr.	e. Kr.	apr. J.-C.	d.C.	A.C.	D.C.
B.C.	A. Ch.	f. Kr.	av. J.-C.	a.C.	?	A.C.
period	periode	period	période	periodo	período	período
century	Jahrhundert	sekel	siècle	secolo	siglo	centenário
decade	Jahrzehnt	dekad	décade	decade	decenio	decênio
year	Jahr	år	année	anno	año	ano
6 months	halbjahr	halvår	semestre	semestre	semestre	semestre
3 months	vierteljahr	kvartal	trimestre	trimestre	trimestre	trimestre
2 months	sechsteljahr	två månader	bimestre	bimestre	bimestre	bimestre
month	Monat	månad	mois	mese	mes	mês
week	Woche	vecka	semaine	settimana	semana	semana
day	Tag	dag	jour	giorno	día	día
hour	Uhr	timme	heure	ora	hora	hora
minute	Minute	minut	minute	minuto	minuto	minuto
second	Sekunde	sekund	seconde	secondo	segundo	segundo
date	Datum	datum	date	data	data	data
time	Zeit	tid	temps	tempo	tiempo	tempo
calendar	Kalender	kalender	calendrier	calendario	calendario	calendário
Julian	Julianische	juliansk	julien	Giuliano	juliano	Juliano
fiscal	finanz-	räkenskaps	fiscale	fiscale	fiscal	fiscal
ordinal	Ordnungs-	ordnings-	ordinal	ordinale	ordinal	ordinal
clock	Uhr	klocka	horloge	orologio	reloj	relógio
yesterday	Gestern	igår	hier	ieri	ayer	ontem
today	Heute	idag	aujourd'hui	oggi	hoy	hoje
morning	Morgen	morgon	matin	mattino	mañana	manhã
noon	Mittag	middag	midi	mezzogiorno	mediodía	meio-dia
afternoon	Nachmittag	eftermiddag	après-midi	pomeriggio	tarde	tarde
evening	Abend	afton	soir	sera	tarde	noite
night	Nacht	natt	nuit	notte	noche	noite
midnight	Mitternacht	midnatt	minuit	mezzanotte	medianoche	meia-noite
tomorrow	Morgen	imorgon	demain	domani	mañana	amanhã

Table I. Time Elements

U.S. Standard Time Zones and properties are:

Zone	Standard	NST	AST	ADT	Time
Newfoundland	(Standard)	-0330			
Atlantic	"	AST	-0400	-0300 ADT (Daylight)	"
Eastern	"	EST	-0500	-0400 EDT	"
Central	"	CST	-0600	-0500 CDT	"
Mountain	"	MST	-0700	-0600 MDT	"
Pacific	"	PST	-0800	-0700 PDT	"
Yukon	"	YST	-0900	-0800 YDT	"
Alaska-Hawaii	"	HST	-1000	-0900 HDT	"
Bering	"	BST	-1100	-1000 BDT	"

The Newfoundland Zone doesn't use Daylight Time, nor do Arizona and parts of Indiana. Moreover, it's not unique in being on the half hour from Greenwich. There are 16 other cases, as listed in the International Standard Time Chart of the Official Airline Guide, Worldwide Edition. That's really the authoritative list. Oddly, only a few countries have to use Daylight Time to delude themselves — including the United States, Cuba, Chile, France, and parts of the British Commonwealth. And to most of these it's called Summer Time, not Daylight (Saving) Time.

Halfhour time zones deny extension of the alphabetic zone designations, like EST, because there are more different zones than letters. They're supposed to be mnemonic anyway.

REFERENCES

Standards — Numeric Date

- ISO 2014-1976, Writing of Calendar Dates in All-Numeric Form.
- ISO 2711-1973, Information Processing Interchange — Representation of Ordinal Dates.
- ANSI X3.30-1971, Representation for Calendar Date and Ordinal Date for Information Interchange (\$3.00).
- U.S. Government FIPS PUB 4 (1968 Nov 01), Federal Information Processing Standards Publication, Calendar Date.
- Australian Standard 1120-1971, The Writing of Calendar Dates in All-Numerical Form.
- British Standard 4795:1972, Specification for the Presentation of Calendar Dates for Information Interchange.
- Canadian Standard Z243.20-1977, Representation of Calendar Date for Machine to Machine Data Interchange.

Standards — Fiscal Calendar

- ISO 2015-1976, Numbering of Weeks.
- British Standard 4760:1971, Specification for the Numbering of Weeks.

Standards — Local Time of Day

- ISO 3307, Information Interchange — Representation of Time of the Day.
- ANSI X3.43-1977, Representation of Local Time of the Day for Information Interchange (\$3.00).

Standards — Universal Time

- ISO Draft Standard 4031, Information Processing — Representation of Local Time Differentials.
- ANSI X3.51-1975, Representation of Universal Time, Local Time Differentials, and United States Time Zone References for Information Interchange (\$1.50).

Composite Standards

- British Standard 5249:1976, Representation of Elements of Data in Interchange Using Data Processing Systems, Part I: Representations of Dates and Times.

Methods

- R.G. Tanzen, "Conversions between calendar date and Julian day number", Commun. ACM 6, No. 8, 1963 Aug, 444
- H.F. Fliegel, T.C. Van Flandern, "A machine algorithm for processing calendar dates", Commun. ACM 1, No. 10, 1968 Oct, 857
- R.A. Stone, "Tableless date conversion", Algorithm 398, Commun. ACM 13, No. 10, 1970 Oct, 621
- J.D. Robertson, "Remark on Algorithm 398", Commun. ACM 15, No. 10, 1972 Oct, 918
- F.E. Randall, "Interrogating data-sensitive files", Computer J. 17, No. 4, 1971 Nov, 302-305

Describes usage of qualifiers (IN, BEFORE, AFTER, EVER) for all dates within a period, and qualifiers (DURING, UNTIL, SINCE, ALWAYS) true for at least one day in a period.

TIME ELEMENTS IN OTHER LANGUAGES

The computer business is aware that survival lies in software. So why shouldn't software be built to be salable internationally? With the help of Tables I and II, and some programmed switching, this should be easy.

Notice that some words are spelled with characters other than letters. This is to remind you that these characters of ASCII are replaced, in the other languages, with the following graphics:

]	Å)	ä	(Swedish circle a)
]	U)	ü	(German umlaut u)
]	N)	ñ	(Spanish tilde n)
]	ç	(Portuguese cedilla c)

All other accented letters are formable with the backspace and diacritical marks, e.g., ð. For Table II, the first entry is an adjective (semiannual notice), the second an adverb (do it semiannually). □

English	German	Swedish
yearly	jährlich	års
yearly	"	årligen
semiannual	halbjährlich	halvårs
semiannually	"	halvårsvis
quarterly	vierteljährlich	kvartals
quarterly	"	kvartalsvis
bimonthly	zweimonatlich	-
bimonthly	"	varannan månad
monthly	monatlich	månads
monthly	"	varje månad
weekly	wöchentlich	vecko
weekly	"	varje vecka
daily	täglich	dags
daily	"	dagligen
hourly	stündlich	tim
hourly	"	varje timme

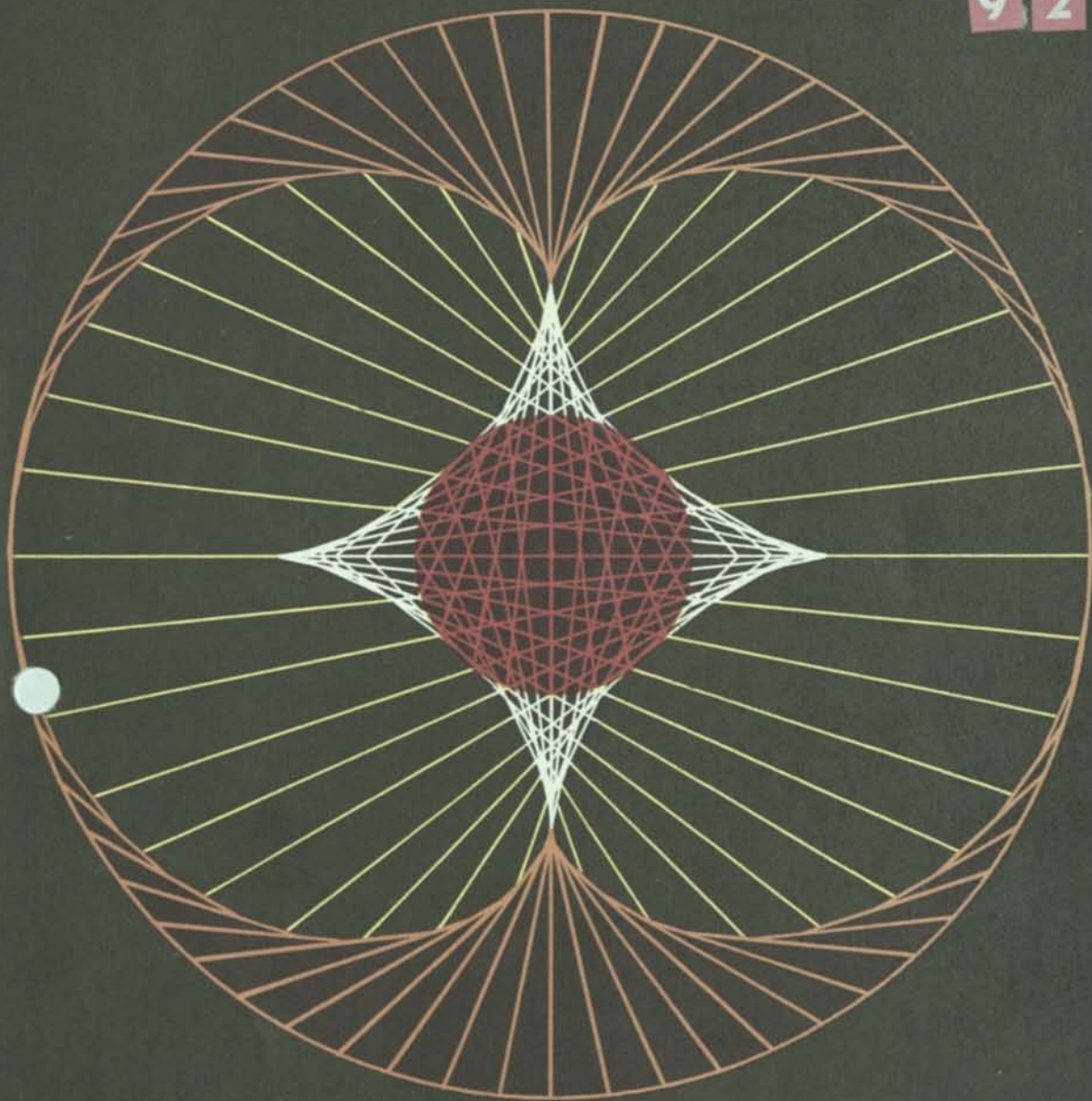
Table Iia. Time Modifiers

English	French	Italian	Spanish	Portuguese
yearly	annuel	annuale	anual	anual
yearly	annuellement	annualmente	anualmente	anualmente
semiannual	semestriel	semestrale	semestral	semestral
semiannually	semestriellement	semestralmente	semestralmente	semestralmente
quarterly	trimestriel	trimestrale	trimestral	trimestral
quarterly	trimestriellement	trimestralmente	trimestralmente	trimestralmente
bimonthly	bimestriel	bimestrale	bimestral	bimestral
bimonthly	bimestriellement	ogni due mesi	bimestralmente	bimestralmente
monthly	mensuel	mensile	mensual	mensal
monthly	mensuellement	mensilmente	mensualmente	mensalmente
weekly	hebdomadaire	settimanale	semanal	semanal
weekly	toutes les semaines	settimanalmente	semanalmente	semanalmente
daily	quotidien	quotidiano	cotidiano	cotidiano
daily	quotidiennement	giornalmente	diariamente	todos os dias
hourly	a chaque heure	-	por horas	horário
hourly	d'heure en heure	ogni ora	a cada ora	a cada hora

Table Iib. Time Modifiers

the mathematics teacher

9 2



APRIL 1979
Volume 72
Number 4



DUNNWOOD SQUARES

R. G. Dunn

The Tenth Step—Metric Time and Angles

E. Lewis Frasier

Regardless of Sex

Grace M. Burton

National Council of Teachers of Mathematics

The *Mathematics Teacher* is devoted to the improvement of mathematics instruction in the junior high schools, senior high schools, two-year colleges, and teacher-education colleges.



Member of the
Educational Press Association
of America

The *Mathematics Teacher*, an official journal of the National Council of Teachers of Mathematics, is published monthly, September through May, at 1906 Association Drive, Reston, Virginia 22091. Dues for individual membership in the Council are \$15.00 (\$7.50 for students), which include \$7.00 for the journal. For an additional \$7.00 (\$3.50 for students) an individual member may also receive the *Arithmetic Teacher*, the other official journal of the Council. Institutional membership rates to the journals are as follows: one journal, \$17.00; both journals, \$34.00. A discount of 20 percent is allowed on ten or more copies of either journal sent to one address with the same expiration date. Single copies of either journal are \$2.50. Remittance should be made payable to the National Council of Teachers of Mathematics. Please add \$1.00 for mailing outside the United States.

Printed in the U.S.A. Second-class postage paid at Reston, Virginia, and at additional mailing offices.

Copyright © 1979
The National Council of Teachers of
Mathematics, Inc.

EDITORIAL PANEL

- LAWRENCE D. HAWKINSON, Henry M. Gunn High School, Palo Alto, CA 94306; Chairman
STEPHEN I. BROWN, State University of New York at Buffalo, Amherst, NY 14260
RICHARD A. HANSON, Burnsville High School, Burnsville, MN 55337
JOHN D. NEFF, Georgia Institute of Technology, Atlanta, GA 30332
THELMA M. SPARKS, Anne Arundel County Public Schools, Annapolis, MD 21401
LEROY C. DALTON, Wauwatosa West High School, Wauwatosa, WI 53222; Board of Directors liaison
HARRY B. TUNIS, Reston, VA 22091; Managing Editor

STAFF

- HARRY B. TUNIS, Managing Editor
CHARLES R. HUCKA, Director of Publications Services
ROWENA G. MARTELINO, Advertising Manager
JOAN R. COOPER, Production Editor
JANE MORRISON, Editorial Aide
JAMES R. TEWELL, Circulation Manager

All correspondence should be addressed to the *Mathematics Teacher*, 1906 Association Drive, Reston, Virginia 22091. Manuscripts should be typewritten, double-spaced with wide margins on 8½" × 11" paper. The original and four copies are required. If the manuscript is not accepted for publication, the original copy will be returned. Priority areas are listed in the editorials that appeared in the September 1977 and September 1978 issues.

NCTM BOARD OF DIRECTORS

- SHIRLEY A. HILL, University of Missouri—Kansas City, Kansas City, MO 64110; President
JOHN C. EGSGARD, Twin Lakes Secondary School, Orillia, ONT L3V 2P5; Past President
BETTY BEAUMONT, San Antonio Independent School District, San Antonio, TX 78210
F. JOE CROSSWHITE, Ohio State University, Columbus, OH 43210
LEROY C. DALTON, Wauwatosa West High School, Wauwatosa, WI 53222
FLOYD L. DOWNS, Hillsdale High School, San Mateo, CA 94403
EDGAR L. EDWARDS, JR., Virginia State Department of Education, Richmond, VA 23216
VERNON R. HOOD, Portland Community College, Portland, OR 97219
GAIL D. LOWE, Los Angeles Unified School System, North Hollywood, CA 91601
JESSE A. RUDNICK, Temple University, Philadelphia, PA 19122
WILLIAM A. STANNARD, Eastern Montana College, Billings, MT 59101
CATHERINE D. TOBIN, Lexington Public Schools, Lexington, MA 02173
JAMES W. WILSON, University of Georgia, Athens, GA 30602
JUNE J. M. YAMASHITA, Kailua High School, Kailua, HI 96734
JAMES D. GATES, Reston, VA 22091; Executive Director

HONORARY PRESIDENT

- JOHN R. CLARK, Professor Emeritus, Columbia University, New York, NY 10027

METRICATION AIDS EDUCATION—AND VICE VERSA

*All you ever wanted to know about
why we're going metric—but were afraid to ask.*

By R. W. BEMER
Honeywell Information Systems
Phoenix, AZ 85005

I learned the customary system and the CGS system (centimetre-gram-second) but am now an avid supporter of the SI (Le Système International d'Unités). But I understand people's reluctance to learn a new system when they have the old one so well in hand and use it so effectively.

For example, let's examine the matter of expressing volume. The SI has only the litre unit, whereas our old system has several. In fact, the historical reason for having several units may have been to express volume in units of one. Instead of saying a given number of litres, we can just say a firkin, a hogshead, or a barrel. And everyone knows that a barrel contains 31.5 gallons—except if it contains petroleum products, in which case it's 42 gallons. And except for malt beverages, in which case it's 31 gallons. (See fig. 1). Of course these barrels are for liquids, but the barrel is also defined for dry content, measured in quarts rather than gallons. Everyone knows that a barrel for dry content contains 105 quarts—except if it contains cranberries, in which case it's 87 quarts (fig. 2).

It's interesting to see the relationship between barrels of liquid and barrels of dry measure. Let's start with the official definition. A dry quart has the same volume as 1.1636 liquid quarts. Calculating 105 times 1.1636, divided by 4 quarts to the gallon,

Editor's Note: At the 9–12 April 1978 meeting of the Board of Directors of the NCTM, a motion was passed to permit individual authors in Council publications to use the spelling they prefer for meter or metre, liter or litre, and their derivatives. R. W. Bemer expressed his desire to use the -re spellings.

Adapted from a lecture given at the February 24–26 meeting of the NCTM in Phoenix, 1977.

we apparently discover a new barrel of 30.54 gallons. Let's call it the *barrel for dry liquids* (fig. 3)!

You may think that this is a variation in one uncommon unit, and that such inconsistencies do not exist in, say, linear measure. Let us take the mathematician's viewpoint, because we are such and seek the lowest common denominator in linear measure. In doing so, we observe (table 1) that the old system is again rich in terms, whereas the user of the metric system has only the metre.

TABLE 1

Mathematicians Seek the Lowest Common Denominator	
1 statute mile =	8 furlongs
=	80 chains (survey)
=	176 chains (football)
=	320 rods
=	880 fathoms
=	1 760 yards
=	5 280 feet
=	7 040 spans
=	8 000 links
=	15 840 hands
=	63 360 inches
=	84 480 digits
=	4 055 040 ounces
=	63 360 000 mils

You think I'm mistaken in showing the number of ounces per mile? Not so. The ounce is equivalent to 1/64 of an inch and is used in measuring leather thickness!

When students think they have found the lowest common denominator for this set, there will always be another to confound it. The engineer's chain, for example, is either 50 feet or 100 feet. But the system is obviously rich in relationships. For example,

If the average horse is 15.84 hands tall (and we know that 1 mile = 15 840 hands), is 1 mile equal to 1 000 horses?

And that's for just the statute mile. How about sea horses and the nautical mile?



Fig. 1. Standard barrels for liquids

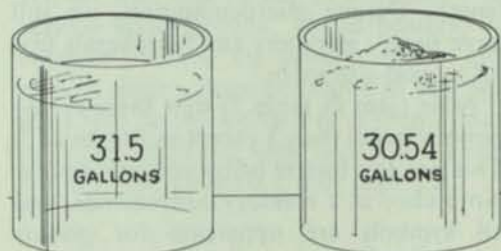
One knot equals 47 (statute) feet, 3 inches. The number of rope knots paid out in 28 seconds equals the number of nautical miles per hour.

Again, as mathematicians, we verify the relationship,

$$\frac{3600 \text{ seconds}}{28 \text{ seconds}} = \frac{1 \text{ nautical mile}}{47' 3''}$$

Thus, a nautical mile equals 6075 feet, but the American nautical mile equals 6076.11549 feet and the British nautical

lieve that it will. The concepts of both number base and exponentiation are built into the system, in the mechanism of prefixes (table 2).



(a) Wet liquids

(b) Dry liquids

Fig. 3.

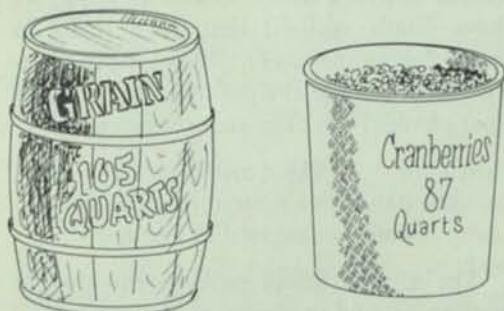


Fig. 2. Standard barrels for dry content

Exa and *peta* may be new to some. They were adopted quite recently to fill out the range. And what a range it is! Sufficient for all galactic measurement, and all the way down to subatomic measurement.

Observe *nano*, which is only 10^{-9} . Do you know how far light and electricity travel in a nanosecond? Let's calculate it. The speed of light is very close to 300 Mm/s (mega-metres/second). Table 2 defines nano as

TABLE 2
Metric Prefixes

10^{+1}	<i>i</i>	10^{-1}
exa (E)	18	atto (a)
peta (P)	15	femto (f)
tera (T)	12	pico (p)
giga (G)	9	nano (n)
mega (M)	6	micro (μ)
kilo (k)	3	milli (m)
hecto (h)	2	centi (c)
deka (da)	1	deci (d)

mile equals 6080 feet. So we're not too exact in our knot tying!

That's enough about the old system; you can find many more oddities yourself. But imagine the time wasted in learning these relationships, and especially the computational difficulties that may arise!

The SI Aids Learning

Will the metric system help students to learn mathematics and other sciences? I be-

10^{-9} and mega as 10^6 . Because $6 - 9 = -3$, which we see from the table is *milli*, the speed of light equals 300 mm/ns (millimetres/nanosecond). Simple, isn't it? Light travels 300 millimetres in a nanosecond—just 5 millimetres short of the old foot.

If you wonder why this example was chosen, remember that computers now operate

Concern over balance of payments is a critical factor in going metric.

at nanosecond speeds and less, so it's a tough problem to get the results anywhere by wires that must be under 300 mm in length. Despite microcomputers, we still have some computers and peripherals that fill a good-size room.

Note (also in table 2) that there are no powers larger than 3 except multiples of 3. This practice fosters better comprehension. Note also, as a memory convenience, that all symbols are uppercase for powers greater than +3.

The SI Abhors Fractions

How about those fractions? Do all students have hand calculators that work in fractions? So they can do problems like

If Sam can dig 32 ditches in 9 hours, and Joe can dig 24 ditches in 7 hours, how long will it take them to dig one ditch—together? If they can keep out of each other's way?

I have often wondered whether such problems have much real applicability—if perhaps they were dreamed up just to teach the difficult art of fractions.

Fractions may not be difficult to you, but they are to the French, who cannot understand where that 0.375-inch dimension came from, because $3/8$ means nothing to them. My company had to import teaching materials from Italy to show the French why such numbers existed, because it is illegal to sell nonmetric materials in

France. The only fraction the French use is one-half. They will order a demilitre of wine, but a smaller amount is not a quarter litre, which would be 2.5 decilitres, but rather an even 2 decilitres. So you see that our fractional system is the awkward octal or hexadecimal one of computers. Theirs goes 2, 1, 0.5, 0.2, 0.1, 0.05, and so on, to form a sequence that repeats in 10s.

Scaling recipes up and down is difficult for most people. Ask yourself what half of $5 \frac{5}{8}$ is. It's actually a very useful figure. You would use it to saw a 2-inch-by-6-inch board in half, because the rough size of 6 inches becomes $5 \frac{5}{8}$ inches when finished. How much easier to ask, "What's half of 140 mm?"

Why Convert to SI?

We'll come back to the value of the metric system to the learning processes, after a digression to question why the United States is converting at just this time. After all, the metric system has been our only legal system for over 100 years, and all our customary units are defined by their metric values to avoid chaos. Could it be that we have finally realized that SI is really so much better and easier that we should convert now despite the possibly high cost of that change? No. The real reason was

1978 April 01

the date when nonmetric products could not be sold in Europe!

The United States government may be getting around to it only now, but General Motors, IBM, Ford, Honeywell, Rockwell, and hundreds of other companies have been converting for a long time. The reason is plain—trade and the balance-of-payments factor, which translates directly into jobs and well-being for the American people.

Kenyon Taylor of the U.S. Metric Conversion Board, appearing on the "Today" show in New York, said that our loss of exports, if we didn't convert to metric design, would amount to about five million jobs!

Conversion and Dollars

General Motors has one answer to the question, "Doesn't it cost too much to change?" They now have over 950 different fan belt sizes that they (and service stations) must stock. When fully converted to metric manufacture, there will be fewer than 50 different sizes! Here's a clear case where a profit is turned on conversion, not a loss.

Metric measure will reduce inventory size.

Your garage will be more likely to have your size in stock if the required inventory is smaller. Don't forget, over 40 percent of the cars on U.S. roads contain metric parts.

In many other cases like this one, forced redesign has yielded economic benefits that were lurking there all the time—if only we hadn't been so blind. They may often exceed conversion costs. In manufacturing, where most conversion costs were thought to lie, ways have been found to minimize these costs and spread them over a period of years.

As metric manufacturing begins, one can compute the corresponding settings on scales calibrated in customary units. Later these scales can have overlays applied. Then new metric machine tools can be purchased without added expense, because such tools do wear out and become obsolete via the newer numerically controlled machines. Gradually, as the percentage of metric manufacture increases, this process can be repeated.

Conversion and People

The impact on people is more critical. There are many diehards who will fight the SI. The best way to rebut them is to avoid conversion entirely (because it's a crutch), and particularly to avoid exact conversion.

Most exact conversion is not justified because of machine tolerances that are already considered acceptable. Those nails are just as likely to be chopped off at ex-

actly fifty millimetres as they are at exactly two inches.

The diehards often use exact conversion to deride the metric system. Somehow football seems to be a favorite target. But the diehards don't have to worry about it. That is one sport that will never be converted, even though the Canadian football field is 100 metres long. All U.S. stadiums have the stands crowding the 100 yards and goal areas as closely as possible, and it would be prohibitively expensive to change. Your children, and their children, will still play football, and they will just think of the yard as an archaic term peculiar to football, just as you and I think of the cubit as an archaic term peculiar to arkbuilding.

During the course of legislation, conversion was also raised as a bugaboo by various union groups. The carpenters, for example, wanted the government to give each of them a huge allowance to buy new tools (see fig. 4). Of course, a journeyman carpenter starts out with tools worth much less than the allowance they requested.



Fig. 4. Metric and nonmetric hammers

Figure 5 shows why the electricians couldn't be too recalcitrant about converting to the SI. They've been using it right along!

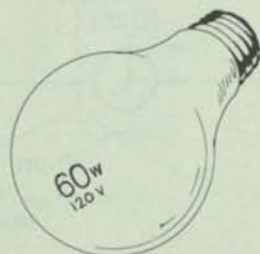


Fig. 5. Metric light bulb

The SI Is Coherent

The chief pedagogical value of the metric system is its coherency, as demonstrated in figures 6 and 7. A cube with a side the length of one 100-mm cigarette has a capacity of one litre; the water required to fill it has a mass of one kilogram.

While living in Paris, I found coherency in my hat size. (In the U.S. it's $7 \frac{5}{8}$, a fractional value that I cannot relate to anything at all.) I wrapped a tape measure around my head, read 61 centimetres, ordered a size 61 beret at Galeries Lafayette, and—voilà!—it fit!

The coherency will also be found in the kitchen, where the confusing pounds, quarts, cups, tablespoons, and teaspoons disappear, to be replaced by millilitres. You won't notice the difference between the 4.9-millilitre teaspoon and the 5-millilitre measure that will replace it. I had to memorize that there were 3 teaspoons per tablespoon, and I still get unsure at times. No memorization is needed when the units are 5 millilitres and 15 millilitres. In Europe, it is common to combine ingredients with a mass scale. Table 3 is a metric recipe for your inspection, if not cooking.

Let's look at a metre stick marked off in centimetres. Imagine it's a dollar in length. Halfway for a dollar is 50 cents. Halfway for a metre is 50 centimetres.

I've noticed that students learn particularly well when you relate things to money. And in this coherent system the same scale can be used to represent temperature (in

degrees Celsius). Figure 8 shows water freezing at 0, boiling at a dollar. Our body temperature is 37 cents, and the thermostat should be set at 18 cents or less. Again, the value ratio of 18 cents to the dollar is well-understood, and so one gets the "feel" of Celsius temperature. The same scale can be used to represent mass—from 3-kilogram babies to 100-kilogram football players. Never convert.

TABLE 3
Metric Recipe

Happy Day Cupcakes	
625 ml	sifted cake flour
375 ml	sugar
15 ml	baking powder
5 ml	salt
125 ml	butter (room temperature)
250 ml	milk
5 ml	vanilla
2	eggs

Measure sifted flour, add sugar, salt, and baking powder. Place shortening in mixing bowl; stir to soften. Sift in dry ingredients. Add 175 ml milk, beat 2 minutes on medium speed (by hand beat 150 strokes per minute). Add eggs, vanilla, and remainder of milk. Beat 1 more minute. Bake cupcakes for 20 minutes at 175°C. Cool, then frost with favorite frosting.

Now let's see how the coherency works in the computational world (Pokorney 1973) by comparing solutions to a problem in both measurement systems.

Problem: A generator supplies an effective 300 kilowatts (357 kilowatts at 84% efficiency). Its driving engine is 30% efficient.

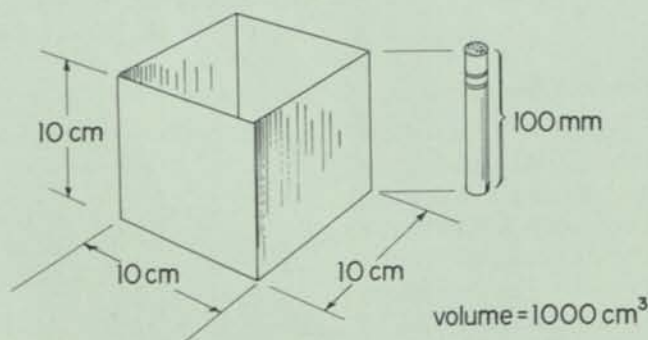


Fig. 6. Length-Volume-Mass coherency
A litre of water weighs one kilogram.

1. What *horsepower* (hp) is required to drive it? For fuel having calorific value of 18 000 BTU per pound, what's the fuel consumption in *pounds per hour*?

2. What *power* is required to drive it? For fuel having calorific value of 42 megajoules per kilogram, what's the fuel consumption in *kilograms per hour*?

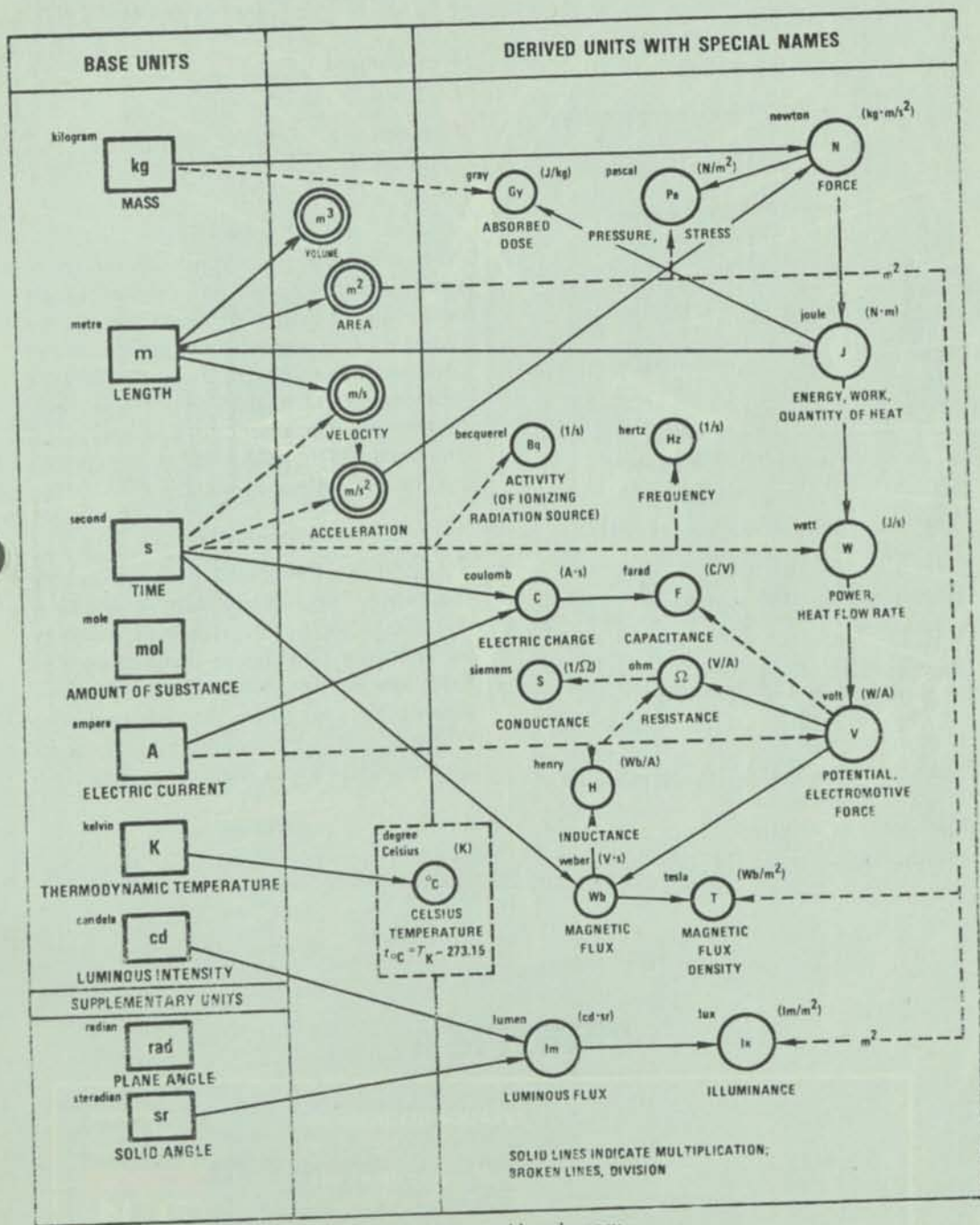


Fig. 7. Systemwide coherency

As you see, the first problem is in customary units, whereas the second problem is in metric units. For both we need the conversion of 3600 seconds in an hour. For the

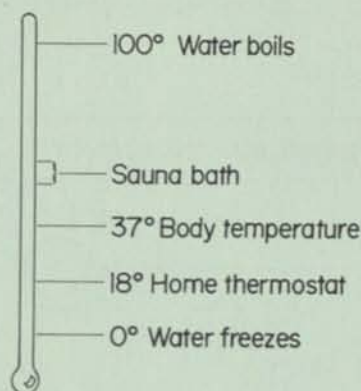


Fig. 8. Relative temperature

customary units, we also need the following:

$$\begin{aligned} 1 \text{ hp} &= 746 \text{ watts} \\ &= 550 \text{ ft-lb/sec} \\ 1 \text{ BTU} &= 778 \text{ ft-lb} \end{aligned}$$

The calculations in customary units are as follows:

$$\text{Horsepower required} = \frac{357 \times 1000}{746} = 480 \text{ hp}$$

$$\text{Power input at 30\%} = \frac{480}{0.3} = 1600 \text{ hp}$$

$$\text{Heat content of fuel} = \frac{1600 \times 550}{778} \text{ BTU/sec}$$

$$\text{Fuel consumption} = \frac{1600 \times 550}{778 \times 18\,000} \text{ lb/sec}$$

$$\begin{aligned} &= \frac{1600 \times 550 \times 3600}{778 \times 18\,000} \text{ lb/hr} \\ &= 227 \text{ lb/hr} \end{aligned}$$

The calculations in metric units are as follows:

$$\text{Power input at 30\%} = \frac{357}{0.3} = 1190 \text{ kW}$$

$$\text{Heat content of fuel} = 1190 \text{ kJ/s}$$

$$\begin{aligned} \text{Fuel consumption} &= \frac{1190 \times 1000 \times 3600}{42 \times 1\,000\,000} \text{ kg/hr} \\ &= 102 \text{ kg/hr} \end{aligned}$$

Note how easy the second calculation is, due to coherency; one watt (power) equals one joule (quantity of heat) per second (time). In this case we could substitute and combine into a single expression, where the meaning of each term would be comprehensible. The absence of conversion factors unclouds the relationships, thus bettering the student's understanding.

Conclusion

The SI is a powerful tool for unlocking the barriers to understanding. Welcome it and use it, and your teaching productivity will increase. You may not need to work so hard, and the remaining work will be more pleasurable as your students are not "turned off" by computational complexities. Here's to a Metric America.

REFERENCE

Pokorney, Joseph L. "Metrication and Systems Design." *Honeywell Computer Journal* 7(1973):123-26.

A

AmericA

Editorial

New Software Ensures Integrity of Files While Allowing Open Access to Data

By Robert W. Bemer

File security and integrity are, indeed, difficult problems when one thinks of the conventional situation with two elements — a person and a file. But they are separable problems. Whereas security is a consideration for all files, integrity usually is not a problem for read-only files (except by the person or program creating them).

Oddly, in the extensive work done on file security and integrity, a few systems have catered to the case of three elements — a person, a file, and an intelligent go-between program, the prototype of which is the "daemon" of the Multics system. Here we actually wish to give file access to many persons, while still protecting integrity. The interposing program acts as a surrogate to do this. In theory, and in some practice, the indirection is compound. The exterior program, talking to the user, can call upon secondary programs and/or databases, which in turn can subcontract.

Here are some examples of where this type of function is desirable:

- A "trivia" game where a player with a certain level of success is deemed understanding and qualified enough to add his own questions.
- A field trouble reporting system for software.
- Personal data files that anyone may inspect, and correct data on one's self.
- Mail systems.

To be effective, the controlling program requires much intelligence, plus alteration capability that may be exercised flexibly, and in a very short time. This puts heavy requirements upon the programming language used. Until two years ago, I had never seen this capability in any of the programming languages in use. (Even APL, lest its fans think I have overlooked it.) Now it exists in the TEX language, which Honeywell has placed in the public domain, for standardization under ANSI Committee X3J6, Text Processing. In a lesser degree, the UNIX operations system has many of these capabilities, for those familiar with it.

The indirect file access problem is very similar to those of CAL (Computer-Assisted Learning). When a student blows the program, or gives an unforeseen (and therefore unconsidered) response, it must be altered very quickly to account for that case.

User Identification

Before getting into specific examples, a general problem of validation must be discussed. A random number generator plays a large role in preventing replication of queries. One time it may be "Where do you part your hair, Robert Bemer? L, C, or R?" Another time "What school did you attend in 11th grade?"

I know that privacy laws forbid many identifying questions, but I have a list of many that may be asked quite legally. And then there is the first qualifying question: "Do you voluntarily agree to submit question-answer pairs that may be used to identify you to this system, knowing that the questions could be asked of others, and that the answers may be exposed by any security breach?"

The Trivia Game

Mike Copeland, a contributor to Honeywell's DM-IV System, has experimented with the trivia game. He seeks answers to the fuzzy set problems for databases (where the query is not quite right, but close enough), and to the addition of data arising from uncontrolled sources. I have added the same to our general library of TEX programs, to demonstrate many of the features of that language.

Note these points of integrity control:

- The file that the users access is only a call to a driving file under another user-ID, which in turn accesses the database.
- If that is transported to another system (with TEXTLIB), access fails, and the user is told so.
- The program and database are reloadable through SAVE tapes made each night. A day's added questions are no loss, if that process is needed.
- Unless the user is a psychopathic vandal, adding questions via the program is so easy and interesting that he won't

Robert Bemer is a consultant-computer systems with Honeywell's United States Information Systems Group, Phoenix, Ariz. His remarks are excerpted from a talk he gave earlier this year at Interfile '79 in Chicago.

try to discover the actual path mechanisms.

— If he would like to write programs using similar techniques, we'll make them available to him directly. No need to crack the file security.

— All modifications of the Q-A file is conditional. If everything isn't just right, the program can re-ask or move on.

— The program prompts for variants of the answer, all of which are stored as possible answers.

Honeywell has been running an experimental program with the Washington School District in Phoenix. From their terminals, the students discovered the trivia game by themselves, and have put in a large number of their own questions, knowing absolutely nothing about the mechanics of the system software.

Field Reporting System

Bob Brandt has constructed an ingenious field reporting system for software problems. It begins by providing all field representatives one of several user-IDs for a central computer system.

Given such access, one may customarily create any files as desired, and control their privileges — giving or denying various types of access permissions to users with other IDs, selectively or generally — to read, to write/modify/add, to query, and even to delete or purge the entire file. But in this case there is another user-ID which has all privileges for their IDs. When the field rep dials in, it looks like this:

ISER-OD? NE-REGION
PASSWORD?

- tex call star/entry

"STAR" is the controller ID: "entry", its interactive file-creation program, asks the user if help is required. Then a form-filling process begins — questions, answers, and answer validation. The description of the software problem is entered in free form; "entry" formats it automatically. When all questions have been answered to the satisfaction of the program "entry", it creates a file in the ID "star" on the user's behalf, but immediately locks it for any writing. This is because the total file system records the user's ID as "originator", which would normally give him all access permissions. "Write-lock" overrides all such permissions.

On the hour, another program from "star" surveys the newly-created files, making and saving copies with itself as originator (owner), and destroys the original copy. If the user has created any other files in his own ID, the control program may destroy them, too, for that is not the purpose of the access rights so granted.

The user knows the nominal name of his original copy, and of the replacement copy. He may list either to see the results, but he may never again modify it in any way. In effect, this is a specialized mail system where the message is forced to be in a controlled and unalterable format. Any corrections to entry data must be made at time of origination.

I think that many files access applications could benefit from this technique. It does demand the availability of the text-processing language to edit the system controls.

A Consultant's Database

I have a small database on people in the computer field. It contains only name, telephone number, title, address, and a few descriptions. It's a relational database, but pointerless — being a linear file powered by TEX programs. If unmechanized, it would fall in the category of a Christmas list. It has no descriptions that I would consider a problem for anyone's privacy, but I will permit anyone to alter their own information, even to wiping out the total entry. After all, I use it to organize conferences and sessions, and to provide gatekeeper-like references for others, etc. If people don't want me to refer to them for a better position, etc., that's their problem.

On the other hand, when I can get an automatic mailing out for the contents, I can permit the people in the databases to correct their own address and other information if they wish. That would keep it more up-to-date and have more integrity that I could ever do by myself.

So here is the other side of the security coin. Rather than keeping you out, we welcome you in, for many good reasons. But only because adequate controls are now possible.

I hope that the new capabilities exposed here will lead to new types of applications of computers and databases. Properly used, it should certainly mean the end of the closed shop type of operation.

Computer Business News

W. David Gardner — Editor
John F. X. Drohan — Managing Editor

Susan D. Carlson — New England Editor
David A. Copthorne — Associate Editor
Deborah de Peyster — West Coast Editor
John Greitzer — West Coast Editor
Robert J. Kozma — Financial Editor
Michael Lowe — Associate Editor
Karen Slater — Associate Editor
Sam Wetmore — Chief Copy Editor
John Rhea — Contributing Editor
Carl L. Howard — Director of Photography

EDITORIAL HEADQUARTERS:

370 Lexington Ave., Suite 308, New York, N.Y.
10017. Telephone: (212) 686-9530.

New England Bureau: 797 Washington St., New-
ton, Mass. 02160. Telephone: (617) 965-5800.

West Coast Bureau: 407 California Avenue, Palo
Alto, Calif. 94306. Telephone: (415) 329-8220.

OTHER EDITORIAL OFFICES:

Washington, D.C.: 621 National Press Building,
529 14th Street N.W., Washington, D.C. 20045.

England: Steve Burton, IDG Publishing Ltd.,
140-146 Camden Street, London NW1 9PF
Phone: 01-267-1201. Telex: 851 264737.

W. Germany: CW Publikationen, Freid-
richstrasse 31, 8000 Munich 40, Phone: (069) 34-
90-61. Telex: 5215350.

Asia: Hidetsuna Sasaki, Dempa/Computerworld
Company, Dempa Building, 1-11-15, Higashi
Gotanda 1-chome, Shinagawa-ku, Tokyo 141,
Phone: (03) 445-6101. Telex: J2424461.

Australia: John P. Costello, Computerworld Pty.
Ltd., 3 Bruce Street, Crows Nest, NSW 2065.
Phone: 929-5388, Telex: SECCO AA25468.

Brazil: Eric Hippeau, Data News, Computer-
world do Brazil, Servicos e Publicacoes Ltda.,
Rua Alcindo Guanabara, 25/10th floor, 20031
Rio de Janeiro, RJ Brazil. Phone: (021) 242-7396
or 263-8046. Telex: 11-32017(WORD BR).

CW COMMUNICATIONS/INC.

Board Chairman/ Publisher	Patrick J. McGovern
President	W. Walter Boyd
VP-Research	Jack Edmonston
VP-Marketing	Roy Einreihof
VP-Editorial	E. Drake Lundell
VP-Finance	William P. Murphy
VP-Circulation	Margaret Phelan
VP-Production	Lee Vidmer

SALES	
Vice-President	Roy Einreihof
Administrator	Lynn Victor
VP-Research	Jack Edmonston
Promotion Manager	Noreen Kimball
Market Research	Kathryn V. Dinneen

CIRCULATION	
Vice-President	Margaret Phelan
Circulation Manager	Barbara Jeannet
Customer Serv. Rep Assistant	Peggy Kelly Carmela Priante

PRODUCTION	
Vice-President	Lee Vidmer
Production Manager	Peter Holm
Production Assistant	Donovan White Thomas Monahan

● The Sixty-Sixth Infotech
State of the Art Conference

London 26-28 November 1979

INFORMATION TECHNOLOGY

INFOTECH LIMITED EDUCATION AND TRAINING DIVISION

Infotech Limited is the world's largest independent supplier of technical education and information on data processing. The Education and Training Division of Infotech mounts a continuing series of Training Courses, State of the Art Conferences, State of the Art Tutorials, Buyers' Forums, Industry Briefings, and Special Seminars. The Education and Training Division also sponsors the European Computing Review, the major annual event for the European computing community.



INFORMATION ACQUISITION, STORAGE AND RETRIEVAL, IN THE OFFICE ENVIRONMENT

R W Bemer

Honeywell Information Systems

Phoenix

AZ

R W Bemer obtained a degree in Mathematics at Albion College, Michigan. His distinguished computing career began with RAND Corporation, and he has since obtained experience with a variety of firms including IBM, Univac, Bull and General Electric. In 1970 he joined Honeywell Informations Systems as Staff Consultant and Editor of the Honeywell Computer Journal. In 1974 he became Senior Consulting Engineer with Honeywell, and in recent years he has headed the development within the company of an interactive enquiry system based on the relational model.

INFORMATION ACQUISITION, STORAGE AND RETRIEVAL, IN THE OFFICE ENVIRONMENT

INTRODUCTION

Our theme is convergence. This implies that communications have been successful for quite a while. So have computers. And perhaps we should put them together to change the way offices are run. Hardware costs are low enough now.

But only maybe. First we might look at some previous inspirations of this kind. In 1957 the big thing was going to be automatic language translation. Much money and effort was spent, until Mortimer Taube showed why it could not be done. It was not that computers lacked power; but rather that people do not write or speak in a logically constrained manner.

In the 1960s it was MIS (Management Information Systems). We were going to put all of our business data together, model some alternatives, and pick one that would maximise return on investment. Again, it was not that computers lacked power; it was many other factors, such as the fact that secretaries and not managers do the typing, making it hard to validate data, let alone put quantitative weights on it.

This leads to my central point, that most offices are not ready for a United Airlines-type database, and never will be. They do not have the same type of requirements, and the workers will probably never think that way.

The offices of today, and yesterday, already have databases. They enter data, they store it, and they retrieve it. But there are certain drawbacks as follows:

- The data is stored on paper, which is now more expensive than electronics.
- Data is often duplicated, and subject to error and loss.
- It is usually stored on the basis of 0 or 1 indexing methods, infrequently on 2 or more methods. Relationships are hard to perceive.
- Although mental indexing often augments the physical, it fails when the indexer retires, moves, dies, etc.
- Some part of the database is often stored in a different location from that of the prospective user of the data.

ACQUISITION

When the new office systems fail, storage and retrieval will probably not be responsible. Acquisition is more likely to be the cause. Whether you use part of someone else's database, collect your own data via transactions, or enter it locally from paper records, you must have logical data and structure, and not nonsense. Storage is a physical process for data whose logical characteristics have been determined in its acquisition.

The database administrator's first job is to serve as entry gatekeeper, sampling and testing the data as attempts are made to transfer it from entry buffers into the real system. Otherwise, conditions will become lax, and the system will degrade. It is at the acquisition portal that indexing must be done, that descriptors must be inserted, and that tests must be made for differences that are not real. (Data that appears different but really is not, is much more frequent than the reverse.)

The following two steps are required:

- 1 Automatic testing for validity and logic during entry
- 2 Post-acquisition conditioning, that is, testing the entered data as a whole. For example, one would not validate spelling as each word was entered. It is better to wait and test the whole set against a dictionary.

Another possibility is to test the data against synonym data, to see if it can be represented by a synonym (token) rather than the full form. This is valuable in compacting database size and speeding up searches. A simple maxim is: 'If a decision worth making is made with certain data, it's worth validating the data first!'

Conversion

For offices that already have some form of database (payroll is common), some conversion will be required. Perhaps from EBCDIC to ASCII code, for communications and timesharing. Many have little structure indicated, and structure will have to be found and incorporated. Text processing programs will be best for this. I have not changed my mind since 1974 when I said that by the 1980s business function concepts will be built into hardware, and that 'programming will be a subset of text processing and should be thought of that way' (001).

STORAGE

Keeping to the assertion that storage is only a physical process, we can inspect the physical forms available, and the possibility of *multiple* representations.

We are usually advised that three representations are available: hierarchical, network, and relational. Those are logical properties, and will not be treated here.

Linear files

I expect that only linear files will be acceptable to office workers (002). They are comprehensible to the average person, by methods as simple as a store dump, in a way

that pointered and chained files are not. Even for relational databases, the third normal form is not vital; people are unaccustomed to think that way. If a computer program knows or can find all references to where an item occurs, it can update all occurrences from the list.

Though the method is little used as yet, linear files have the capability of semantic labels. That is, the data and its types and properties can be physically stored in a contiguous mode for retrieval. This permits selection by both content and type of content. A string processor will do this for hardware that is not content-addressable.

Each line of a linear file may be given a prefix or suffix. Where a prefix ends, and the content begins (or where content ends and suffix begins) is determined by a pre-assigned character. The search window is opened or closed when it is encountered.

In Figure 1 the prefix character is \backslash .

```
Jones, Sam#263-4467\18.46.9.23
D22S3G1\Sam Jones
V5V4V1\if sal:gt:3400 then perform "hisal"
```

Figure 1: Prefix and suffix usage

- The first line is suffixed. It shows that Sam Jones is on mailing lists 18, 48, 9, and 23.
- The second line is prefixed. It shows that Sam Jones is in department 22, section 3, and is male.
- The third line shows that the program statement exists in versions 1, 4, and 5 of the program.

Proponents of pointered databases might argue retrieval speed advantages, but this is not necessarily so if the string search is tuned in hardware to the same efficiency as addressing, and parallelism is employed in searches (003).

Linear files also permit several database administrators in the office, not just a specialist upon whom your business safety rests.

Co-files

The people/computer cost ratio has influenced development of computer usage methods. The first major impetus in databases was to devise pointer methods so there would be only one copy of an item of data in the database. This helped to keep the database 'clean', i e, ensuring complete retrieval because variants of identical data would not occur due to faulty encoded representation, for example, a database wherein there were three high-level executives in a company with similar names - Wohlegemuth, Wohlgeomuth, and Wolgemuth.

In some instances this can cause high overhead in people costs to control the database entry process, so that all data is structured within the database operational method. Tom Gilb has some comments on this point (004) as follows:

'The (existing database) languages encourage the use of complex file structures, which results in a greater net loss of human time, machine time and money than any documented gains can justify.'

'The obsession with languages has already obscured the need for technological know-how in the really critical areas of database design such as reliability, maintainability, and portability.'

'There is an easily documented lack of fundamental training in database (that means 'file', of a special sort) technology. People know they don't know enough, so they jump to the incredible conclusion that they have to get a complex file organisation system to 'help' them.'

But both processing and storage (memory) costs have decreased drastically since these methods were devices in about 1964. One may wish to consider 'co-files', where the data are intentionally duplicated to provide faster processing. Co-files are, of course, only feasible for databases with a low change to display (C/D) ratio. Again, United Airlines cannot use them easily. Two co-file methods are:

- 1 Files containing the same data, but in a different physical structure for faster search and/or extraction.
- 2 The main file contains the data in encoded form (usually called 'tokens'), while the co-file contains what needs to be substituted for the token when humans must read the data.

Method 1 is understood easily in the punch card context: make a copy of the deck/file; keep the original in its sequence (ordering); sort the second copy into another sequence; use the appropriate deck for the demand that arises. Stefferud (005) says 'In short, the information is sorted according to the answers to expected questions'.

It is this principle of conceptualising processes via separate linear files that will cause Kneitel's title 'EAM (Electric Accounting Machines) will rise again!'

It would not be feasible to use people to do this, even if the first deck changed slowly. Nor would it warrant the cost of the cards involved. But electronic storage carries no such overhead for destruction, and the frequency of the change can control the automatic re-creation of the specialised second form, or co-file. Of course, in this form, as in EAM card decks, either deck can contain only a subset of the total data, as long as they form the entire set when keyed together.

There may be a natural tendency for old computer *cognoscenti* to suspect these remarks. If so, they just have not worked in the real office environment. Let us consider what percentage of office files change, and how often:

- I have a consultant's database on people in and around the computer field. When a change or new addition comes in, I mark it on the hardcopy summary. Once every one or two weeks I add these into the database and make a new listing. But the relational database queries use a special file created from these files, not the originals. The process does have some cost attached, so I do not use it when adding a descriptor to an entry. But when enough changes have accumulated to be significant to the percent-wise integrity of my search, I ask for the 'new' option when calling the program 'WHO'.

Figure 2 shows the source form for an entry. Figure 3 shows a sample interrogation, and Figure 4 the results. Note that this is not an answer to a single query, but rather a subpicture of the database, giving much more real information.

```
{1Morgan, Howard}
{2215-243-77312}
{3191743}
{4Prof. H. L. Morgan|Dept. of Decision Sciences|The Wharton
  School|Univ. Pennsylvania|Philadelphia, PA 191744}
{5univ-PA-Wharton-office-automation-WP-
  calendar-UltraText-CACM-ACM-database5}
```

Figure 2: Entry in linear database

```
-call qannotator/profile
Starting 'profile' at 12:30:17
Major descriptor? France
Want minors?      yes
CR only, when done
Descriptor?       stwe
Descriptor?       hdwe
Descriptor?       IBM
Descriptor?       COBOL
Descriptor?       AFNOR
Descriptor?       stds
Descriptor?
```

Figure 3: Typical query

- At Honeywell Information Systems in Phoenix, the telephone database is updated weekly on Friday afternoon. On Monday morning a new microfiche copy awaits the telephone operators. A hardcopy phone book is photocomposed every two or three months, or after drastic revision. Statistically, this works out very well.
- Daily changes are often adequate. A Chicago bank found that providing a daily microfiche of account records cost a third as much as on-line terminals for enquiry.
- Cases can doubtless be found where intervals of an hour down to 10 seconds or so will be optimal.

The new office

When a new electronic database capability, augmented and improved by distributed storage of its components is established, what should be done with the data that is entered? Should it be accumulated indefinitely? This could be justified, on the basis of ever-lower hardware and storage costs, but should not be implemented because it will increase the path complexity to really important data.

A sensible method is to archive old data to microform if there is any expectation of needing it in the future, and to destroy it if there is not. A monthly cycle would be ideal for me with a fiche containing all of my in and out correspondence, plus indexes

Profile for -France-

Benay, Joannes	1			
Boss, Jean Paul	1			
Bourgain, J.	1	4	6	
Carteron, Jean				
Chasles, Francoise	1	3	4	6
Combelic, Donn				6
Dreyfus, Philippe	1			
Frontard, Raymond			5	6
Genuys, Francois	1	3		
Grimanelli, J			5	6
Guilhamou, J.		3	6	
Lepicard, Georges	2			
Maisonrouge, Jacques		3		
Marguerite, Andre	1			
Sallé, Francois	1		6	

1=stwe	4=COBOL
2=hdwe	5=AFNOR
3=IBM	6=stds

Ending at 12:30:55

Figure 4: Response to the query

to reports that I was invited to read (instead of being sent a copy with a cover list).

This means that electronic marking options should be provided as follows:

- Do not put on fiche
- Put a pointer on fiche
- Put the document on fiche.

Conversely, there will always be a use for paper or its equivalent. It is still more acceptable legally, and often cheaper than providing some workers with terminals and communications access.

Some kinds of office data purposes are:

- Informative
- Transient (meeting tomorrow)
- Production schedules
- Bulky, for reference
- Continuing, with updates
- Require retrieval (design decisions)
- For legal record.

RETRIEVAL

Retrieval of stored data may be needed for the following:

- Legal data
- Billing and financial audit
- Business analysis and optimisation
- Education and training
- Correspondence
- Correlation of data for prediction and invention
- Personal relations
- Payment of costs incurred, such as, payroll, material, shipping, rent, etc
- Address information for communication
- Historical events.

For these and many other purposes, retrieval may be done on the basis of:

- 1 Document names
- 2 Document characteristics, descriptors
- 3 Document content
- 4 Non-documentary data content.

It is important to avoid loss in retrieval. Some clear (although often ignored or misunderstood) requirements must be met to use any of these methods. They are as follows:

- 1 Unless an elaborate index is prepared and maintained, names should have information content. Of course they must be unique. Incorporating the date helps to ensure this.
- 2 Catalogue the descriptors. Refine them to make sure they are meaningful. Do not overlap, or have duplicates (e g, 'mgmt' and 'management').
- 3 Control content by concordances.
- 4 Run fire drills, and other tests, on stored data at off-peak times, to make sure it comes out when really needed.
- 5 Batch requests when possible, even for pointered databases. Put the most frequently requested data up front.

Suppose a mistake in entry went unchecked, and 'Wisconson' was recorded somewhere in the database. If retrieved by document name, it will come out that way, and humans will compensate (possibly with laughter). If retrieved by the descriptor 'state', the same will occur. But it will not work for retrieval by content, unless deliberate variants are known and given.

That is where preconditioning comes in. The end use of storage is retrieval for real use. This implies that it will pay to condition data *before* storage, for the following:

- Proper file naming
- Correct spelling
- Control of grade level (GM's STAR)
- Ambiguities

- Multilingual capability
- The Fog Index (per Rudolph Flesch)
- Use of tokens.

The last is where the 'Wisconsin' problem could have been avoided. It is a fuzzy set problem, which can be solved by a concordance or partial string searches. We have the option to correct it, or always use 'WI'. One can go further, and store only the (subscripted) zip-code in the database; if the city-state part of the address needs to be printed, the file of printable lines is collated to the zip-code correspondence file. This also reduces database size, giving faster search.

The invisibility problem

Working with electronic files poses some new problems. Consider files that are the (inherent) property of a single user. The user should, of course, set up a catalogue and subcatalogue structure in a well-planned and hierarchical manner. The file names should be significant and associated directly with the catalogue names.

The only catch is that you forget the several mnemonic associations. How do you 'thumb' through to jog the mind?

The first requirement is a program that builds a file to indicate the catalogue structure and membership, such that each line of the file is the name of a specific file. With this one can build monitoring programs to inspect each file automatically, and extract pertinent identifying information. My program 'TAKES' builds, from the list of files, a new file containing the name of the file followed by the first five printing lines of the file. This makes for very effective visual inspection. Often you say 'How could I have forgotten to eliminate that file? It was just temporary, and of no permanent value.'

If the files are for processes, and have a special section for explanation of the action, the same procedure can build a file which gives the names of the several process files, and what they do.

These 'samplers' are useful to take inventory when the formal mechanisms fail.

How to retrieve?

I am not convinced that a database, perfectly structured or not, is subject to unambiguous query. Nor am I overconfident of the office worker's ability to phrase unambiguous queries. Nor am I confident of an immense intelligence in database retrieval systems to prompt with questions that will coerce the user into unambiguous statements.

We have had some success, however, with creating a large number of programmed processes that are conditioned by the answers to questions that the processes first pose. These, then, are the capabilities available to the user, provided only that he or she knows about them. This knowledge is provided by an inventory system, much as the engineer studies the parts catalogues. Figure 5 shows a query from a user who wants the fiscal calendar for 1984. Figure 6 shows the reply, indicating the most likely program for the purpose. Figure 7 shows the menu of descriptors the user could have been shown, if unfamiliar with the procedure. Figure 8 shows the query for what the program actually does, and Figure 9 is the actual run.

```

-call texlib/inventry
Want a keyword menu? (64 lines) no
Keyword? print
Keyword? fiscal
Keyword? year
Keyword? calendar
Keyword?

```

Figure 5: Is my need fillable?

TEXLIB programs for your keywords:

Hits	Program	Keywords
4	ap/fiscalyr	-print-fiscal-year-calendar
1	ap/n-up	-print
1	h/2print	-print
1	h/filprint	-print
2	l/today	-fiscal-year
3	u/date	-fiscal-year-calendar
1	u/explist	-print
1	u/print	-print

Figure 6: Reply to query for fiscal calendar

*lcl	content	hold	operation	sorter
*lin	control	hyphen	ordered	source
2014	convention	identify	order	space
2print	convert	impressive	ordinal	spacing
3I	copies	improve	originate	special
4J	copy	increment	outline	spelling
80	correct	index	overstrike	split
ASCII	correspond	infix	packaged	spurious
ISO	cpy	insert	page	square
Julian	crt	interactive	paper	standard
.....				

Figure 7: Keymenu, if requested

```
-call texlib/ap/fiscalyr!explain
```

FISCALYR will generate the
International (and American)
Standard fiscal calendar for
any year of the 20th century.

Figure 8: Will it do what I need?

```
-call texlib/ap/fiscalyr
For the year 19?? 84
Create as a current file (reply 'c'),
or just print? (CR)
```

```
Fiscal Year 1984

FW   M T W T F S S
1    02 03 04 05 06 07 08
2    09 10 11 12 13 14 15 JAN
3    16 17 18 19 20 21 22
4    23 24 25 26 27 28 29
5    30 31 01 02 03 04 05
6    06 07 08 09 10 11 12 FEB
7    13 14 15 16 17 18 19
....
```

Figure 9: Result of program

Formatting retrieved data

It will be a long time before the office worker is willing to abstract data and build a report in his head or on paper. The retrieval system had better do that for him. Managers, according to Kneitel (006), demand formatting of the retrieved data, in that they 'like to see everything ... on a single sheet of 8.5 x 11 paper held vertically'.

But he cautions against filling it up like a dump. '... getting a prospective reader's attention ... is done with headlines, illustrations and attractive layouts with lots of "white space"'. It is easy to see that the popularity of report generators will remain undiminished.

Time

Many retrieval actions for the office will occur upon demand, but even more will be initiated on a time schedule. So one needs a clock program supporting a master timer file list. All processes (even demand or deferred) should occur only through this list, because it is the best audit point for security and history.

The program must be able to differentiate users and terminal devices. This is needed for control of calendars and mail. Also for warnings and wakeup calls. The user, engrossed in some project in the electronic office, needs interruptions like:

```
Time is 09:45
You have a meeting with Coburn at 11:00
Needed: The Havemeyer file
```

CONCLUSION

A recent Diebold study, made in anticipation of widespread office automation, cautions that complex systems will not work, particularly if they deviate from known and used

methods. This is not a new insight. David Kahn has described the solving of the Linear B writing (around 1400 BC, the earliest writing that we can read) (007). He tells us that:

'Several tablets were found with a totalling entry on the bottom line. Some tablets were indexed on their edges so that the bookkeeper would not have to pull out a whole batch to get the one he needed.'

As this is still a common practice, we must indeed assume that computers and communications must blend, not intrude.

The physical file structure for the electronic office should be linear and visible. This was characteristic of many pre-CODASYL databases, as well as the many new databases being formed via microcomputers.

Success in usage will come from involving as many office workers as possible in the knowledge and workings of the database process, and not from supporting a specialised priesthood.

REFERENCES

- | | | | |
|-----|--|-----|---|
| 001 | Data Congress News
IFIP Stockholm (7 Aug 1974) | 005 | STEFFERUD E
<i>Convergence of technologies points toward the automated office</i>
Proc IEEE/NBS Conf <i>Trends in applications: 1978 - Distributed processing</i>
Gaithersburg MD pp 6-11 (May 1978) |
| 002 | ROBINSON S L
<i>Non-CODASYL DBMS not horrible prospect</i>
Database J vol 8 no 3 pp 16-17
CR 34114 (1978) | 006 | KNEITEL A M
<i>EAM will rise again</i>
Infosystems vol 24 no 7 p 88 (July 1977) |
| 003 | BEMER R W
<i>Alternatives to the generalised DBMS</i>
Proc Infotech State of the Art Conf <i>On-line databases</i> pp 150-161
Infotech International Ltd Maidenhead UK (Nov 1976) | 007 | KAHN D
<i>The code-breakers</i>
Weidenfeld and Nicholson London p 919 (1967) |
| 004 | GILB T
<i>Gilb's methodology</i>
Computer Weekly (8 Sept 1977) | | |

.....973-4807#04338#R72
.....973-2701#24852#R72
.....973-3793#17553#R72
.....973-2979#15645#R72
.....973-2052#20466#R72
.....997-3001#35556#R72
.....973-2361#18907#R72
.....973-3529#14093#R72

Search File - TEST1

Abajian, Robert	973-3719#ABC/	FED#CT19-MC5#(none)
Abraham, Ronald F.	973-4807#XYZ/	Eng#CT05-B131#2M160
Abramson, James L.	973-2701#XYZ/	Mfg#CT05-B64#1C295
Abshire, Ruth E.	973-3647#XYZ/	P&PM#CT05-B167#2K202
Acuna, Ernest G.	973-3405#XYZ/	Mfg#CT05-F10#FB129
Adams, Duane	973-3793#XYZ/	Eng#CT05-B92#2K200
Adams, Gary D.	973-2979#XYZ/	SD#CT05-B125#2A296
Addis, Raymond	973-3843#XYZ/	Eng#CT05-B170#2A242
Adler, Wanda L.	973-3412#XYZ/	Mfg#CT05-A76#1B136
Aguilar, Ron N.	973-4753#XYZ/	IRM#CT05-C33#1P210
Ahlo, Ainsley A.	973-2052#XYZ/	E&CR#CT05-B118#2A245
Aiken, Eric G.	997-3001#ABC/	FED#CT21-T11#(none)
Akins, Glen A.	973-4842#XYZ/	Eng#CT05-B170#2A242
Alaface, Angelo A.	973-3007#XYZ/	Eng#CT05-B145#1P349
Albano, Tony	249-5512#ABC/	FED#CT06-K88#173
Aldano, Richard	973-3529#XYZ/	SD#CT05-C90#1A347
Aldridge, Don F.	973-4753#XYZ/	IRM#CT05-C33#1P210
Aldridge, Sherri	973-4985#XYZ/	Mfg#CT05-B17#1C221

File to Search - TEST2

```
call texlib/xp/collator
Starting COLLATOR at 08:15:55 on 79-05-30
Name of search file? test1
First column of key? 7
Last column of key? 14
Need any data except key? no ←
Name of file to search? test2
First column of key? 21
Last column of key? 28
Starting to merge!
Want the reject file, too? yes
Reject file is in SINK
Selected file(s) in current file (*)
Ending COLLATOR at 08:16:34
```

Job History

Ahlo, Ainsley A.	973-2052#XYZ/ E&CR#CT05-B118#2A245
Abramson, James L.	973-2701#XYZ/ Mfg#CT05-B64#1C295
Adams, Gary D.	973-2979#XYZ/ SD#CT05-B125#2A296
Aldano, Richard	973-3529#XYZ/ SD#CT05-C90#1A347
Adams, Duane	973-3793#XYZ/ Eng#CT05-B92#2K200
Abraham, Ronald F.	973-4807#XYZ/ Eng#CT05-B131#2M160
Aiken, Eric G.	997-3001#ABC/ FED#CT21-T11#(none)

Selected File

Subject File

Albano, Tony	249-5512#ABC/	FED#CT06-K88#173
Alaface, Angelo A.	973-3007#XYZ/	Eng#CT05-B145#1P349
Acuna, Ernest G.	973-3405#XYZ/	Mfg#CT05-F10#FB129
Adler, Wanda L.	973-3412#XYZ/	Mfg#CT05-A76#1B136
Abshire, Ruth E.	973-3647#XYZ/	P&PM#CT05-B167#2K202
Abajian, Robert	973-3719#ABC/	FED#CT19-MC5#(none)
Addis, Raymond	973-3843#XYZ/	Eng#CT05-B170#2A242
Aguilar, Ron N.	973-4753#XYZ/	IRM#CT05-C33#1P210
Aldridge, Don F.	973-4753#XYZ/	IRM#CT05-C33#1P210
Akins, Glen A.	973-4842#XYZ/	Eng#CT05-B170#2A242
Aldridge, Sherri	973-4985#XYZ/	Mfg#CT05-B17#1C221

Reject File

```
call texlib/xp/collator
Starting COLLATOR at 08:21:22 on 79-05-30
Name of search file? test1
First column of key? 7
Last column of key? 14
Need any data except key? yes
Name of file to search? test2
First column of key? 21
Last column of key? 28
Starting to merge!
Want the reject file, too? no
Selected file(s) in current file (*)
Ending COLLATOR at 08:22:58
```

Job History

.....973-2052#20466#R72
 Ahlo, Ainsley A. 973-2052#XYZ/ E&CR#CT05-B118#2A245
973-2361#18907#R72
973-2701#24852#R72
 Abramson, James L. 973-2701#XYZ/ Mfg#CT05-B64#1C295
973-2979#15645#R72
 Adams, Gary D. 973-2979#XYZ/ SD#CT05-B125#2A296
973-3529#14093#R72
 Aldano, Richard 973-3529#XYZ/ SD#CT05-C90#1A347
973-3793#17553#R72
 Adams, Duane 973-3793#XYZ/ Eng#CT05-B92#2K200
973-4807#04338#R72
 Abraham, Ronald F. 973-4807#XYZ/ Eng#CT05-B131#2M160
997-3001#35556#R72
 Aiken, Eric G. 997-3001#ABC/ FED#CT21-T11#(none)

Selected File

AU1610	YBDX	[CPT1]	
AU1611	XVMX	[CPT1]	
AU1613	YACR	[A-C6]	
AU1614	XZOT	[0EV3]	
AU1616	HXHZ	Consolidated Valve Industries, Inc.\PO Box 247\Matthews, NC 28105	
AU1620	FILR	[EAT1]	
AU1621	XXYZ	[CPT1]	
AU1623	XXYZ	Hyster Co.\2902 NE Clackamas St.\PO Box 2902\Portland, OR 97208	
AU1624	XXYZ	[A-C6]	
AU1625	ZSDV	[CPT1]	
AU1626	XXYZ	[EAT13]	
AU1631	YACR	[CE1]	
AU1639	YBDX	[EAT13]	
...			
..			

Excerpt from UL Database
(Part no, Type, Supplier)

A-C6	ALLIS-CHALMERS CORP.\P0 Box 512\Milwaukee, WI\53201
CE1	Clark Equipment Co.\Industrial Truck Div.\525 North 24th St.\Battle Creek, MI 49016
CPT1	Towmotor Corporation, Subs. of\Caterpillar Tractor Co.\7111 Tyler Blvd.\Mentor, OH 44060
EAT1	Eaton Corp.\Controls Div.\191 E North Ave.\Carol Stream, IL 60187
EAT13	Eaton Corp.\Industrial Truck Div.\11000 Roosevelt Blvd.\Philadelphia, PA 19115
OEVS	Otis Material Handling Div.\Otis Elevator Co.\8000 Baker Ave.\Cleveland, OH 44102

Excerpt from UL Address File
(Token-to-Address)

OFFICE AUTOMATION AND INVISIBLE FILES

R. W. Bemer

Honeywell Information Systems
Phoenix, AZ, USA

For "Automazione e Strumentazione", OA Issue, 1980 March?

For most businesses, records and files are the life blood, moving through the arteries of the office as power for actions and transactions. The use of files is such an important function that it needs detailed study and analysis of the way it will change if the office is to be automated.

It is assumed that such automation of the office means not only facsimile, high-speed copiers, and new PBXs, but also computers. It follows that the records and files will be converted in many instances to electronic form. That is, as either bit patterns for graphics, or encoded characters for text, stored on the commonly used computer media. And all of these media -- magnetic, optical (video disk), bubble lattices, etc. -- have the property that you cannot pick up the recording medium and comprehend the content visually. In short, most of the files of the automated office will be invisible most of the time, and that's a problem.

Let us remember why it is useful to be able to inspect files visually:

- o We see their content in totality.
- o The human mind can analyze and make decisions while inspecting the file.
- o The human mind can compensate quickly for fuzzy data (such as misspellings).

These features are so useful that some method to replicate their function must be provided for the electronic office, else we shall have to go back to paper files again, losing many new advantages such as:

- o High-speed search and abstraction of the required data.
- o More completely retrieved data, due to storage on the basis of more indexes, and/or with full text search capabilities. Relationships are found easier. All of which is useful to optimization of business actions.

15 95

- o Simultaneous retrieval from different physical locations.
- o Elimination of some redundancy, and reduction in error and loss.
- o The librarian knowledge is not lost if an employee relied upon for mental indexing resigns, moves, retires, or dies.
- o The storage costs are much reduced (the cost of paper is rising, while the costs of electronic media is declining swiftly).

The File Management System

All file usage is dependent upon the filing system -- how data is put away for retrieval -- the controls for creation, cataloging, and access of files. Preparing for the electronic searcher is much different from preparing for the human searcher. Of course the human is still the ultimate searcher, but in the automated office the task must be delegated first to the electronic searcher, which must in many cases operate without supervision or hints.

The invisibility problem demands an organized place to put things so you can get them back! For interchange, and for successful retrieval for your own purposes, it is useful to attach all kinds of semantic characteristics to files. In the past, overdue accounts may have been indicated by red labels on the file folders, or by placement in a certain drawer. Now they're all in one electronic drawer, which has no color. So these characteristics or clues must be signified in another way -- attached names, indices, attributes, groupings, permissions and locks, pointers and references, etc.

Acquisition - General

It is essential that acquired data be tested for data structure and qualities. If logical structure is missing, it must be found and incorporated. Text processing programs will be best for this. It is at the acquisition portal that indexing must be done, descriptors inserted, and tests made for differences that are not real. Particular care must be taken that data of the same class must be named the same, and that data classes that are different are not named the same.

Acquisition - Conversion

Very few automated offices will be those of new companies, dedicated to paperless methods from the outset. Most will be transformations of existing offices, with conversion of the existing files and file management system. This will be a most difficult task. If it were possible to adequately inventory the paper files, they could be placed on microfiche via camera methods, meanwhile creating an electronic file that indexes them by fiche location and content type. In a very few cases it might be desirable to convert some existing paper documents to electronic files, perhaps to get familiar with and check out the new methods of usage. This may be done by direct entry or by optical reading.

Some offices may already have a substantial amount of electronic files arising from data processing. These may require conversion to the ISO Code (no automated office can afford to live with more than one representation of the alphabet, or to have it differ from the alphabet of communication links). It is also probable that the DP form assumes that a computer program procedure will identify the location and type of data in the files. This is unsuitable. The explicit information keys must be added. A COBOL-type picture clause is a minimum requirement; semantic labels for the data are even better, and will be described.

Acquisition - Purchase/Rental

Not all office usage will concern your own files. You may wish to use data from the files of someone else (there are hundreds of computerized databases available at a fee). You may also wish to join that group with your own databases, e.g., a catalog of your products. Note that there is a difference between interrogating a foreign database with its own procedures (that cater to its own file management system), and making a copy of a portion or all of a foreign database to use under your own file management system. They are quite likely to differ. Although my experience with standardization for computers is extensive, I know of no file management system existing or proposed as a national or international standard for computers. This means that each filing system may demand an import/export system to make data portable between systems.

Acquisition - New Entry and Transactions

The same warnings apply -- test for validity, and add as much identifying information as possible during the process.

Linear Files

I expect that only linear files will be acceptable to office workers. They are comprehensible to the average person, by methods as simple as a store dump, in a way that pointers and chained files are not. Even for relational databases, the third normal form is not vital; people are unaccustomed to think that way. If a computer program knows or can find all references to where an item occurs, it can update all occurrences from the list.

Linear files also permit several database administrators in the office, not just a specialist upon whom your business safety rests. The more people that can (legitimately) peruse your files with understanding, the less they are likely to be sabotaged without warning.

Semantic Labels

It is possible to attach semantic labels to each line of linear files. I.e., the data and its types and properties can be stored physically contiguous for retrieval. This permits selection either by content or by type of content. A string processor will do this for hardware that is not content-addressable.

It works this way. Each line of a linear file may be given a prefix (or suffix). Where a prefix ends, and the content begins (or where content ends and suffix begins) is determined by a preassigned character. The search window is closed (or opened, for a suffix) when it is encountered.

```
Nardelli, O. J.#263-4467#C61\18,46,9,23
D22S3G1\O. J. Nardelli
V5V4V1\if salary:gt:3400 then perform "hisal"
```

Figure 1. Prefix and suffix usage.

In Figure 1 the prefix character is "\".

- o The first line is suffixed. It shows that O. J. Nardelli (tel. 263-4467, mail station C61), is on mailing lists 18, 46, 9, and 23.
- o The second line is prefixed. It shows that O. J. Nardelli is in department 22, section 3, and is male.
- o The third line shows that the program statement exists in versions 1, 4, and 5 of the program.

A Data Processing DBMS for the Office?

The people/computer cost ratio has always influenced development of computer usage methods. The first major impetus in databases was to devise pointer methods so there would be only one copy of a datum in the database. This helped to keep the database "clean", i.e., ensuring complete retrieval because variants of an identical datum would not occur due to faulty encoded representation.

It is possible for this to cause high overhead in people costs to control the database entry process, to structure all data within the database operational method. Tom Gilb has studied these dangers [1]. He argues:

"The (existing database) languages encourage the use of complex file structures, which results in a greater net loss of human time, machine time and money than any documented gains can justify."

"The obsession with languages has already obscured the need for technological know-how in the really critical areas of database design such as reliability, maintainability, and portability."

"There is an easily documented lack of fundamental training in database (that means 'file', of a special sort) technology. People know they don't know enough, so they jump to the incredible conclusion that they have to get a complex file organization system to 'help' them."

Mayford Roark [2] says "In one of our activities ... 70 percent of the instructions being executed resulted from DBMS overhead. ... In another of our activities ... nearly half of the file requirements were required for pointers." As Executive Director of Systems for the Ford Motor Company, he should know, after approving about 3000 computer projects, leading to some 50 000 applications.

Perhaps some offices will find a complex pointered database useful, particularly when the change to display (C/D) ratio is high, as for an airlines reservation system.

Linear Co-files

Both processing and storage (memory) costs have decreased drastically since these methods were devised around 1964. I prefer "co-files", where some data is intentionally duplicated to provide faster processing. Co-files are, of course, most practical for databases with a low change to display ratio. This is because there is need for periodic

reconciliation, and perhaps the remaking of special co-files for specific purposes. The three basic co-file arrangements are:

1. A-B and B-A (two replications).

Containing the same data, but in different physical structure for faster search and/or extraction. The B-A file would be used whenever a "B" ordering is required, as for against a B-C file, for example. This is understood easily in a punch card file context.

2. A-B and B-C (one replication).

The main file contains the data in encoded form (usually called "tokens"), while the co-file B-C contains what needs to be substituted for the token when humans must read the data. Imagine A as a part number and B as a manufacturer code. To order we would need to know the manufacturer address, etc., which is C. If a manufacturer makes many parts, we avoid error by single reference (sort of a pointer, itself), at the expense of a small amount of storage. This is usually compensated for in processing time.

3. A-B and A-C (one replication).

A is the link between B and C, often used for relational database work.

On the last type, we had relational databases in punch card days, which may surprise recent computer graduates. They were handled by co-file (card deck) methods for a reason that applies to automated office usage -- the record length was limited to 80 characters. So are many terminal screens!

To find elements with properties "X and Y" or "Z", we put the elements/property deck in a sorter and dropped the cards with property "X". These were rerun to drop on property "Y". All other than these were run to drop on property "Z". Those that did not drop did not match the condition; all others did. The latter were then placed in a collator to match against the element/address deck, for example. Thus we used one cofile to match the relationship, another to get other properties for that subset.

Of course, it's more complicated in real business, but it is this principle of conceptualizing processes via separate linear files that will cause Kneitel's title "EAM ((Electric Accounting Machines)) will rise again!" [3] to come true.

Retrieval - Principles

Poppel [4] says that "A corporation often houses pertinent and even crucial documents that managers and professionals either have forgotten about or are simply unaware of. An electronic search of an indexed system for all pertinent information on a given topic can ensure that no such information gaps arise in the evaluation and decisionmaking process."

Yes, it can, but only if the files are properly cared for and validated. A few years ago I was to demonstrate some relational database capability to a certain U. S. government agency. To prepare, I tested a portion of their database. It appeared to be all capital letters, and studded with spaces, as though an old-fashioned keypunch had been used to enter it. After using a text editor to remove all redundant spaces, the mass storage requirement for the resulting file was just half that of the original. But that was not the most significant finding. A concordance of the file indicated that a certain company had three high-level executives with sound-alike names -- Wolgemuth, Wohlgemuth, and Wohlegemuth. Of course that was not true. The database was dirty. That electronic search might find only a third of the references desired!

This means that electronic search must have the capability to handle "fuzzy sets", with pattern-matching devices of some complexity for full text search. For example, in playing with the "trivia" database recently I responded "Hilary" for the first climber of Mt. Everest. It should have been "Hillary", but the computer replied "Close enough". The search program was "forgiving", an important attribute of file handling for the electronic office.

Realworld databases are produced by people of the real world that enter and process the data. For output of maximum benefit we must provide a maximum of tools and convenience for both input and retrieval. It is not always known in advance what is expected to be extracted from a database upon query or display of some subset.

5. Run fire drills and other tests on stored data at off-peak times, to prove that the retrieval mechanism works.
6. Make use of the unused upper half of the 120 code for your own purposes. E.g., a character to mean "start of an address field", and another for the end of the field. Make sure these indicators can be entered readily with the source data.
7. Use phrases for database elements that appear frequently just as a code book "AMT" might mean "deny credit to the AMT (Amputation) until current accounts are settled". This isn't just for file compression. Such "resorts" make for a cleaner and more searchable database.

Retrieval - Principles

Files are arrays. Their contents may be referenceable, in which case some data may be found via an index, a key, or a map. Or their contents may be anonymous, in which case specific data is found by its content, via a pattern match.

Data not retrieved is detrimental to office operation. There are many ways to guard against such loss:

1. Take care in naming of files. Try to give the names information content. Make them unique, perhaps by incorporating the date of their creation. Cluster them in subcatalogs and sub-subcatalogs.
2. In the catalog of files, augment the file names by as many keywords or descriptors as possible.
3. Catalog the descriptors. Refine them to ensure that they are meaningful. Do not overlap, or have duplicates (e.g., both "mgmt" and "management").
4. Run concordances against files to find hits on certain words or phrases. Note that these are not necessarily the same as the descriptors.
5. Run fire drills and other tests on stored data at off-peak times, to prove that the retrieval mechanism works. Cluster them in subcatalogs and sub-subcatalogs.
2. In the catalog of files, augment the file names by as many keywords or descriptors as possible.
3. Catalog the descriptors. Refine them to ensure that they are meaningful. Do not overlap, or have duplicates (e.g., both "mgmt" and "management").
4. Run concordances against files to find hits on certain words or phrases. Note that these are not necessarily the same as the descriptors.
5. Run fire drills and other tests on stored data at off-peak times, to prove that the retrieval mechanisms work.
6. Make use of the unused upper half of the ISO Code for your own purposes. E.g., a character to mean "start of an address field", and another for the end of the field. Make sure these indicators can be entered facily with the source data.
7. Use syncryms for database elements that appear frequently, just as a code book "AXXYZ" might mean "Deny credit to the XYZ Corporation until current accounts are settled". This isn't just for file compaction. Such "tokens" make for a cleaner and more searchable database.

8. For files that are tables or other forms with quantitative values, test for range and validity upon individual entry. Later, test the entire set of such data for consistency.

9. Do the same for spelling, against a stored dictionary.

I think that it is reasonable to archive old data to electronic form if there is any project that it will be needed in the future, and to just destroy it if there is not. Notice that the physical layout and visual referencing on the microfilm eliminated the pointers of the invisible file system.

This process should be undertaken periodically, but the cycle time will vary considerably. Monthly is my choice to run a microfiche containing all of my in and out correspondence plus indexes to reports and memos that I was invited to read. In lieu of bringing a copy with a cover sheet, this for each item I have the choice of not putting it on fiche, putting it on in entirety or just putting on a pointer. What this requires is a file that is a list of the other file names plus descriptors. I mark my choice for each file electronically on this file which then drives the making of the fiche itself.

Other archiving is possible in electronic form, but on cheaper media. Magnetic tape is still useful because all it requires for archival form is the plastic reels, not the expensive disk storage drives and electronics. About 70 per cent of my own files are so solid now that I am better off to have them on tape.

In rare cases my archival will be back to paper, for various reasons.

I have dealt with the archival choices so far as though there were but one copy of a document in electronic invisible form. Actually that is not usually true, any more than there is only one copy of a paper document. The convenience of one's terminal facilities having copies more than electronic paper copies do. Heavily used workbooks will have a terminal, whereas a copier serves many people. The problem is that you so easily forget that you have made them, as there is no paper form on your desk to remind you.

Archiving

Offices acquire new data continuously. As the size of the accumulated databases grow, the path to any particular set of data must necessarily get longer and relatively more expensive. Theoretically, on the basis of ever-decreasing hardware and storage costs (as for a video disk), we could preserve it indefinitely. Indeed, some planners of automated offices do not see microform as a long term solution because of these video disks. I do not for the cost of storage, but because of the cost of access.

I think that it is reasonable to archive old data to microform if there is any prospect that it will be needed in the future, and to just destroy it if there is not. Notice that the physical layout and visual referencing on the microform eliminates the pointers of the invisible file system.

This process should be undertaken periodically, but the cycle time will vary considerably. Monthly is my choice to make a microfiche containing all of my in and out correspondence, plus indexes to reports and memos that I was invited to read (in lieu of being sent a copy with a cover list). Thus for each item I have the choice of not putting it on fiche, putting it on in entirety, or just putting on a pointer. What this requires is a file that is a list of the other file names, plus descriptors. I mark my choice for each file electronically on this file, which then drives the making of the fiche (CCM).

Other archiving is possible, in electronic form, but on cheaper media. Magnetic tape is still useful, because all it requires in archived form is the plastic reel, not the expensive disk storage driver and electronics. About 70 per cent of my own files are so seldom used that I am better off to have them on tape.

In rare cases, my archival will be back to paper, for various reasons.

I have dealt with the archival choices so far as though there were but one copy of a document in electronic, invisible form. Actually that is not usually true, any more than there is only one copy of a paper document. The convenience of one's terminal facilitates making copies more than electrostatic paper copiers do! (Nearly every worker will have a terminal, whereas a copier serves many people.) The problem is that you so easily forget that you have made them, as there is no paper form on your desk to remind you.

One of the most important programs for the new office is one that "walks" your catalog of files, building a list of all that you "own". Each line of the generated file is the name of a file, in its full catalog format to show structured membership. This list is now used to drive a program that looks at each of the files in turn, building a new file that is an expanded index -- it contains the file name and then the first n printing lines of each. This is used for visual inspection. Many times you will note temporary or redundant copies that can be purged immediately. This is often done automatically, too, by a differencing program to compare two files.

This is not a minor factor. Using these processes we salvaged 9 disk drives worth of space on one 32-drive system. Had we not done so, it would be 9 additional drives and packs, another I/O controller, and more floor space.

Extra and test files are not the only ones that consume storage. So do files that are not needed online anymore. The same programs to walk the file structure can extract other data. The last-used date, the last-changed date, and amount of usage, for example. Or if the files are processes, carrying their own explanation of what they do, a new file can be built for a list of file names and their uses. Such "samplers" are useful to take inventory when the formal mechanisms fail.

As a last note on archiving, it is often useful to archive in ultimate structure, which may not be the original. For example, I would not want a review book to be archived as it is printed; I would rather have each review adjacent to the original works when they are archived. Applying this to business, many files may contribute to a major decision. They should be grouped physically with that decision, in case some of the pointers were forgotten.

About Redundancy

We hear from many experts that a network or hierarchical database is valuable to reduce redundancy. In ComputerWorld's Special Report on Databases, 1979 Oct 29, a F. H. Wu is headlined "Redundancy, Inconsistency Eliminated", and says that "Although DP files are duplicated for 'security reasons', data redundancy is considered a waste of storage space". And a M. A. Vogel says that "with a DB system, all of the business data is organized in a single file, eliminating duplication and redundant data".

used it as a guide to the telephone directory. The two files were ordered by name and address. In the belief that adjacent line pairs would represent the same person. Nothing could be less true! There were people in the directory that were not paid

I'm not sure that this is necessarily a desirable goal:

1. Suppose the pointered connectivity is lost when power fails?
2. Suppose you are the victim of a malicious database administrator, or a penetrator?
3. Suppose that you wish to have variants for the same data?

On point (2), imagine that a catastrophe has occurred in the office, and a certain person will get full blame if some memorandum is examined. That person might try to alter all copies of the memo. But can he be sure, with copying machines in such common usage? History can be rewritten if there is only one copy of the book. A pointered database is more vulnerable to malice. The memo need not be altered. Just cut the pointer, which has no historical record. Indeed, much computer fraud is accomplished in this manner. This cannot be done in the case of a file which has permissions and historical data attached. Figure 2 shows such a file status, obtainable upon demand:

```
File Name-TODAY
Originator-Bemer
Date Created-780123
Date Changed-781207(12,445)
Last Date Accessed-791212
Number of accesses-877
Max File Size-10 LLINKS
Current File Size-5 LLINKS
File Type-Linked
General Permissions-R,E
Specific Permissions-None
```

Figure 2. Catalog Card for a File.

On point (3), there are instances where aliases are in fact desirable. I retell my experience with making a telephone directory for our plant.

While the revision procedure became fairly stable, it was still unstable with respect to organizational affiliation. A reorganization should not demand manual changing of the entire directory. So I extracted name, cost center, and employee number from the payroll file, and used it as a cofile to the telephone directory. The two files were ordered on name and merged, in the belief that adjacent line pairs would represent the same person. Nothing could be less true! There were people in the directory that were not paid

from the Phoenix payroll. And there were name mismatches, primarily because those people wanted a listing other than their legal name, which I think is a moral right.

For example, a Mr. Schallmo was Irvin for his paycheck, Irving in the directory. When I called to find out which was correct, he answered with "This is Mo"! I asked if everyone called him "Mo", and he assured me that they did. However, he wants it "Irving" in the book because "I like it better".

The automated office will have even more need to cater to people, their preferences, and their personal rights. So aliases must somehow be accommodated. The female Lynn may suspect that she will be taken for a boy, and so wants it Lynne in the directory. Queenie May G. prefers a plain G. May in the public scrutiny. And then there is the song about "A Boy named Sue".

Safety, Security, and Surrogates

The filing system plays another important part in office automation. A file may contain data, but it may also contain, as a program, a process useful to the office. If the filing system has an adequate access procedure, ranging from usage by anyone down to usage by only a single person (and then under rigid rules), those access rights may be applied to the processes!

This method of control was introduced to me by Duncan MacGregor, manager of Management Information Systems for Honeywell Ltd. in Toronto, Canada. He uses it to control his computer installation. The operating system for his computer reacts to textual commands, but the general user is not taught these. Instead, MacGregor created a subset of files that were processes, the names of which are the working vocabulary of his users. Some of this vocabulary can be restricted to a subset of users, via the permission and access attributes of the file system.

I call such programs "surrogates", because a "surrogate" is a person acting in place of another, a substitute or deputy. Often used in a legal sense, with good reason. If you have little knowledge of the law, you get a lawyer to plead to the judge and/or jury.

In the days of closed computer shops, programmers acted as surrogates and intermediaries for the real users, presenting the problem to the computer on behalf of the clients. Note that auditability was applied only to the programmers, and security was more controllable.

In timesharing, the human surrogate is seldom used. The client is customarily given direct use of the computer via a terminal. As many system facilities are open as the user cares to learn from manuals or by questioning other users. This is not a necessary mode of operation. It does not meet the needs of auditors. Nor will it work for the automated office.

Our files containing processes are PROGRAM surrogates, versed in talking to both operating system and office worker. They query the worker as to what he wishes to do, and then enter demands against the computer operating system to get that done. The worker has no need to learn operating system jargon, just as the layman need not learn legal jargon. And the system is more orderly. These surrogate programs (similar to the "shell" of UNIX) may be made ever more intelligent and obedient. Properly programmed, they are model users, doing everything correctly that is needed and desired for our automated office. To them, the files are visible. Functionality may be added as required, even specialized to a particular office.

Now you will see where we can obtain safety and security for our invisible office files. The "storage" surrogate is trained (programmed) to make sure that all the necessary retrieval information is recorded before it will put any file away. All traces and pointers must be supplied, keywords given to describe the content, and all catalog information furnished to deposit in the appropriate places. If not supplied, it will simply refuse to store the named data. With this guarantee of an automatically controlled process to move the data from initial temporary status to permanent storage, we achieve safety. It's unlikely that the file will be misplaced and/or lost.

Part of that data is who (or what function) is permitted to look at the data again. Thus the reading permissions are peculiar to the data, not under the control of some so-called "database administrator". Control is in the hands of more than one person, and that is how we achieve security.

Of course much has been made of encryption methods in the last 5 years, particularly on or over unsafe media like communications. There are different gross levels of encrypted security, whether in electronic chips or in programs. But key management remains a problem, and many users that originally requested such security are not now willing to pay its cost, in either money or performance. Surrogate programs provide analogous methods, such as layers of internal passwords. They cost little, and may be either alternative or additive to encryption schemes.

Knowledge and Surrogates

In every office there is some specialization of knowledge. For the answer to a certain class of questions, everyone asks Tony Pizzarello (to name a certain friend of mine). Many such "gatekeepers" obtain their specialized knowledge from reading what passes by them in their jobs. They synthesize new knowledge from the relationships they perceive. A successful automated office must provide either a means for these persons to continue the activity, or an electronic equivalent for that retrieval process.

The common expectation is that the office worker will be able to make queries in his own natural languages. I am curious. I doubt that even a perfectly structured database is subject to unambiguous query. I have only to look at the failure of language translation by computer, two decades ago, and still failing. I expect that the average person will phrase enquiries ambiguous to computer search, and doubt that the database retrieval systems will have the intelligence to correctly prompt with clarifying questions. For example:

"How many black and white rugs were sold in July?"

A fairly good retrieval system will ask "July of what year?" before proceeding. But what will it do if the database covers black rugs, white rugs, and rugs that are both black and white?

We have had success, however, with the menu principle -- the same choice mechanism used for data entry. We have created a number of surrogates whose processes are selected by the answers of the user, whose only knowledge requirement is that the surrogate in fact exists.

Each surrogate has an explanation system and an inventory system (like a parts catalog). Figure 3 is a terminal display for a user exploring the "AUTHOR" surrogate (the user's actions/responses are outlined). Figure 4 shows a query from a user who then wants to know how to format a document. Figure 5 is a portion of the file used by the inventory program. Figure 6 is the reply, to show the most likely program for the purpose (in this case, only one -- a void might indicate a need to construct a new facility). Figure 7 shows the menu of descriptors the user could have been shown, if not familiar with the process. Figure 8 shows a request for a description of what the program really does, and Figure 9 is the actual usage.

Pathways to author/lecture

-call author/explain

Do you know how to use "AUTHOR"? n

AUTHOR is a userid with the specific purpose of helping you to write technical papers and documentation. It will inform and direct you to the proper tools, such as:

- o INVENTORY - to find applicable programs.
- o WHERE - gives the pathname (via subcatalogs) for accessing programs known to AUTHOR.
- o EXPLAIN - for briefings on programs known to AUTHOR.
- o LECTURE - for brief individual lectures on various facets of the publications processes.
- o COMPOSE - an improved formatting program, with indexing, table of contents, and hyphenation capabilities.
- o SPELL - to check the words of a file against a master dictionary.
- o FLESCHE - to compute the FOG index of a file -- length of words, sentences, paragraphs, etc. Will also compute the high school grade level of the vocabulary in a file, like the General Motors STAR program.
- o TEXROFF - A preprocessor which converts source files, with prefix formatting indications, to the infix formatting commands of COMPOSE.

Want further details (28 lines)? n

Want a list of files known to AUTHOR (13 lines)? y

These are the programs/files in, or used by, userid AUTHOR -- 79-11-09 version.

- alfatize	conrun	fill	print	spell
- 2print	convert	filprint	ragged	tab
- aprint	copycols	flesch	roff	tablefix
- asort	crtbs	highcols	roff.doc	texroff
- banner	cutcols	inventory	roman	tuneroff
- billdraf	date	keymenu	rowtocol	update
- build	dict	keywords	ruleform	where
- calc	diff	lecture	seqcheck	wordfile
- chg.case	doconccr	modcols	si	write
- compose	elimdup	n-up	sort	
- concord		prefix	spell-it	

For information about any file or catalog known to AUTHOR, type the selected name --- lecture

Pathname is author/lecture

(Figure 3 continued)

AUTHOR/LECTURE gives you a brief reminder and useful data on these topics:

```

call author/lecture!spacing    for display spacing
                                !paper    for paper size info
                                !layout   for various formats
                                !com      for microfiche data
                                !photo    for photocomp advice
                                !all      to print all lectures

```

For information on any other file or catalog, type its name now; otherwise CR --

Figure 3. Exploring the AUTHOR Surrogate.

Figure 5. The file AUTHOR/KEYWORDS (index vs. attributes).

Programs for your keywords:

```

File Program Keywords
-- author/sale  !status!size!page!layout

```

Figure 6. Reply to query.

Zajant	concedence	fold	month	sort
ASOTI	content	form	multiple	source
B	control	format	wy	spacing
English	conventional	generation	new	spell
GA	conversion	grade	new	spelling
Julian	convert	graphic	notation	subcatalog
Roman	copies	group	number	subject
SI	copy	highlight	numerical	symbol
STAR	correct	hyphen	numeric	table

Figure 7. Keywords, if repeated.

```

-call author/inventory
Want a keyword menu? no
Keyword? calculate
Keyword? page
Keyword? layout
Keyword?

```

Figure 4. Is my need fillable?

```

author/calc          -calculate-page-layout-pitch-column-size-
author/explain      -explain-TEX-program-operation-
author/              -find-TEX-program-keyword-desired-inventory
author/keymenu       -keyword-list-select-
author/keywords      -keyword-by-program-inventory-
author/lecture       -advice-data-subject-paper-format-spacing-
author/letter/write  -write-custom-letter-file-database-desc-
author/where         -author-find-path-name-pathname-subcata-
. . . .

```

Figure 5. The file AUTHCR/KEYWORDS (names vs. attributes).

Programs for your keywords:

Hits	Program	Keywords
3	author/calc	-calculate-page-layout

Figure 6. Reply to query.

2print	concordance	fold	month	sort
ASCII	content	form	multiple	source
B	control	format	my	spacing
English	conventional	generation	name	spell
GM	conversion	grade	new	spelling
Julian	convert	graphic	notation	subcatalog
Roman	copies	group	number	subject
SI	copy	highlight	numeral	system
STAR	correct	hyphen	numeric	table
. . . .				

Figure 7. Keymenu, if requested.

-call author/calc!explain
AUTHOR/CALC is called to calculate the commands that CA/COMPOSE should be given to get a desired final page layout.

Options include page orientation and size, number of columns, % reduction, pitch, and crowding or not.

Figure 8. Will it do what I need?

```
-call author/calc
8.5x11 paper size? n
ISO A4, then? y
Long side (V)ertical or (H)orizontal? v
No. columns per page? 2
Pitch (chars/inch)? (10 or 12) 12
% photoreduction? (CR if 100%) 85
Want page numbering? y
(St)andard tolerances? or as (I)ight as possible? t
Set for 75 lines of 48 characters.
```

Figure 9. Result of program.

Recombination, Formatting, and Display

Creation of new reports from multiple existing documents is a common office usage. Presently the copies of the documents are marked visually and graphically to indicate the rules for formation of the new product. Will we have to produce hardcopy from the electronic files and reenter the same data as a new document? Or is there a means to manipulate the components of the several documents symbolically to form the new product without creating any hard copy or duplicating the data?

The answer is that the latter is possible and practical. It's done with the new text processing languages. Success is so evident that formal standardization is occurring in the United States, and the international standards project is about to begin. This is the next step beyond the report generators so popular in data processing.

Experience has shown that the general office worker can learn and use this language easily. The experts can make programs for repetitive office processes, for use by anyone who can answer questions! A financial executive of the Hoffman-LaRoche company, after learning and using it, said "We don't need your programmers anymore!"

Terminals

The video screen terminal is much preferred for the office, rather than hardcopy terminal. It is quieter, displays much faster, and can show a full form for data entry. Usually its only problem is unfamiliarity. Its display, like the electronic files, is intangible.

For the video terminal to become accepted, new demands must be made upon the filing system and the operating software. As an example, think of how many times we compare two pieces of paper. Where is that small difference? -- the office worker must be provided with a file-differencing functionality. What names appear on both? -- a split screen arc software for looking at two files at once should be provided.

There is the matter of finding the proper place in a file, particularly a large one. Scrolling is very useful for short distances, but requires high character transfer rates. It may be more useful to provide a clue string to pass to the host computer, which then finds it and transmits the text surrounding it to the terminal. A text processor will usually search more accurately than the human eye. Moreover, it can provide all instances of some clue string. This feature should be provided in function keys.

In addition to filling in forms, the terminal is also used to initiate processes, which often require choices to be made. A first-time user of a process will be frustrated if the questions asked are not clear enough or complete. The expert will be annoyed if required to see those long questions every time, particularly at a hardcopy terminal where they appear at typing speed. Only a mnemonic prompt is needed or wanted. The software should first ask if the user is a novice, intermediate, or expert user. Based upon the reply, it selects a graded set of questions. A response of "?" to any question should cause a reissuance of the question in its form for the previous level of experience.

And at any point a response of "HELP" should allow the user to find out about any aspect of the system that is not understood. Even a complete manual of instructions should be a system file available for display.

Conclusions

Only a glimpse, and some perhaps curious detail, has been provided of the severe problems that file invisibility might cause in the automated office. I have not seen any complete study, possibly because the focus so far has been upon the glamorous side -- electronic mail, conferencing, and calendars. Possibly, too, because the real automated office does not exist anywhere yet, and the pains have yet to be reported.

This paper was intended to draw attention to a possible oversight in the soft-colored dreams of the future office. Early reports on word processing were mainly about the resistance encountered in introduction, and about ways to ameliorate this. I believe that the automated office will see this tenfold, because it involves everyone, not just a portion of the secretaries.

When office automation is adjudged successful, it will be found that as many office workers as possible were involved in the knowledge and workings of the file management process, not leaving the task to a database specialist to whom the office might become ransom. The file system itself will be linear and uncomplicated. And the devices used, computers and communications, will be unobtrusive and undemanding of any substantial change from known and used methods.

REFERENCES

1. T. Gilb, "Gilb's Methodology", Computer Weekly, 1977 Sep 08.
2. M.L. Roark, "Evolution in computer systems", Keynote Address, DataBase Directions - the Conversion Problem, NBS/ACM Workshop, 1977 Nov 1.
3. A.M. Kneitel, "EAM will rise again", Infosystems 24, No. 7, 1977 Jul, 88.
4. H.L. Poppel, "The automated office moves in", Datamation, 1979 Nov 25 Special Edition, 73-77.

Computer Security Journal

Winter 1982, Volume I • Number 2

Contents

	Editorial Advisory Board	5
	About the Authors	7
	Editorial	11
	Commentary:	13
	Federal Agency Contingency Plans: A Failure	
	ACM Establishes Technical Security Group	
	A Note on "Developing Standards for Operating System Security"	
Carl Hammer	Managing Computer Security	17
Warren R. Schmitt	Data Security Program Development: An Overview	23
Rolf Moulton	A Strategy for Dealing with Computer Fraud and Abuse: A Case Study	31
Terrence J. Boyer	Contingency Planning: An Opportunity for DP Management	41
Robert W. Bemer	Incorrect Data and Social Harm	51
John D. Tangney Peter S. Tasker	Safeguarding Today's Interactive Computer Systems	57
Rein Turn	Security Issues in Transborder Data Flows	71
Jeffrey A. Meldman	Privacy Expectations in an Information Age	81
Joseph L. Sardinas, Jr. Richard J. Asebrook	"Bridging the Gap" between DP Professionals and Auditors	91
Robert Bigelow	The Security Officer's Role in the Legal Protection of Software	99
A. B. Thom	Hardware Security	105
	Book Reviews:	109
	Advances in Computer Security	
	White Collar Crime	
	Information Security: European Attitudes, Controversies, Needs and Solutions	
Donald A. Marchand	Privacy Protection and Power: Computers and Privacy in the Next Decade	

Incorrect Data and Social Harm

Robert W. Bemer

ABSTRACT: After discussing with examples some causes of incorrect data, Mr. Bemer considers the problem of data invisibility and then reviews seven methods that will prevent, or at least reduce, data incorrectness. This article first appeared in *Data*, a Danish computer magazine, and is reprinted with permission.

Causes of Incorrect Data

Incorrect data can arise from:

- Mistake by the supplier.
- Falsification by the supplier.
- Erroneous entry, undetected by limits or type.
- Non-removal when invalid.
- Insufficiency of qualifiers or relationships.
- Updating amongst multiple copies.
- Arbitrary alteration.
- Program malfunction.
- Destruction of access path.

Some examples or case histories are best to illustrate some of the causes.

Non-removal when Invalid

Although the words "social harm" are a part of the title, I make no claim to be able to judge degree of harm quantitatively. Any presumed ranking would cause much argument. Take the case of Steven Karagianis,¹ who was arrested in his automobile because of a warrant for previous arrest which, although voided, was not yet removed from a computer database. It was coded "CW" for "Cleared Wanted", but the inexperienced interrogator of the database thought that meant "Currently Wanted".

Early the next morning he was found dead in his cell, having committed suicide. His mother said that he had been in jail once before and was terrified of ever doing so again. Some would consider this extreme social harm; others might argue world overpopulation and the acceptable loss of one person, weak in a Darwinian sense.

Insufficiency of Qualifiers or Relationships

The case of Frank D. Booth² is more demonstrative of social harm. On the way to his father's funeral he stopped by the side of the road. A state policeman, running a routine check on the car license, was told that it was stolen. He therefore approached the car with drawn weapon, and somehow Booth was shot dead. His only problem was that his 1975 license plate had the same number as the 1971 license plate of the real stolen auto. The appropriate color, required to distinguish between the two, was not in the database.

¹*ComputerWorld*, 26 April 1976, p. 2.

²*ComputerWorld*, 19 December 1975, p. 1.

Updating Among Multiple Copies

As the concept of distributed databases gains momentum, we all hope that the system designers incorporate safeguards to ensure correct data at all nodes of the network. For example, Computer A is the main source for a database. Computer B makes a copy for local use. Is its copy changed automatically and immediately whenever a change is made to the database in A? Or only upon every n th change in A? How is the time of change attached to each datum? Or is it possible to alter data in A without a history of the change? If so, how can B know when to correct an erroneous transaction?

Arbitrary Alteration

These questions are not idle. Take the case of the taxation department of one of the world's largest countries. So large, in fact, that it required an entire computer to hold and manipulate the file of name-to-address relationships.

And another entire computer to do the same for the file of name-to-monetary relationship. Because people do move, it must be possible to cause a change in the name-to-address file. Now even those with but slightly larcenous minds can see possibilities.

A group of employees conspire to steal. The second computer is interrogated to find the names of persons with substantial refunds due. The first computer receives orders to change the addresses for that group to various addresses of the conspirators. No audit trail mechanism exists. Refunds are sent to the addresses of the conspirators. The checks, with forged signatures, are cashed. Immediately an order is sent to the first computer to change the address again, back to the original. Still no audit trail.

Now comes the taxpayer to complain that he has not received his refund. The first computer is interrogated. Isn't this your address? Yes, it is. And that is where we sent your check, which we have here, signed by you. A forgery, you say? Not our fault. You shouldn't have mislaid the check when you got it.

The ring operated effectively in a technical sense. It was exposed only when conscience led one member to mental breakdown, running through the department screaming a confession for the \$6 million taken.

One of our biggest fears today is that terrorists will be able to use atomic bombs. Few people are worried lest terrorists gain access to any of the databases upon which our lives so depend! It is sometimes not even thought particularly heinous to commit a data crime.

Six men conspired to alter credit records in the TRW database supplied to industry.³ They worked through an inside operator who actually modified the data. The judge gave them trivial sentences (the sternest was 60 weekend days in jail, and a fine of \$3,000) probably because he could visualize the harmful effects of murder, but could not understand the harmful effects of altering data.

It doesn't take much to visualize the possibilities for data crime with terrorist rather than monetary purpose. The route data for computer-controlled trains could be altered to cause accident. The parole and release data for prisons could be altered. Nor should we forget that many a skilled computer programmer was trained in prison.

Program Malfunction

Because I have been a computer programmer for 31 years, it seems ironic that I should be affected so directly by incorrect computer data. More than most people, apparently. In earlier life I was divorced in the State of California. As there were chil-

³ComputerWorld, 13 September 1976, p. 1.

dren, I made periodic payments for support until such time as the youngest became of legal age. Then I stopped, and heard nothing further for six years. Until the state agency sent me a bill for \$5,096.

After a few failures in telephone contact, time passed without further billing, and I assumed that somehow they had corrected the mistake. Two years later I received a demand for \$397, with a threat of arrest for failure to pay. While pondering this, a new demand came two weeks later — for \$54,643! This was too frightening, so I enlisted the aid of a reporter from the Los Angeles Times newspaper.

Conversations began between the reporter, myself, and people that run the computer. To the last they insisted that there was no way to change the data without order from a court of law (but there wasn't any with jurisdiction). The story appeared.⁴ The data has been corrected, and I don't owe anything (officially). But have the programs to create the data been corrected? I don't know.

The fault was that the program included no means for updating the age of the children with passage of real time. It assumed that they stayed the same age forever, and therefore the payments should go on forever.

Was this social harm? It certainly wasted the effort of many people to correct it. There is no guarantee that it is not happening to other people in the same way. And had my heart been weaker, the shock of a demand for \$54,000 might have caused real harm to my family.

Destruction of Access Path

We know that data can be encrypted and decrypted for security purposes. Furthermore, some methods of database access are in themselves an encrypting of the data (e.g., CODASYL). In either type, data can be rendered inaccessible by throwing away the key. In the first case, the unscrambling key; in the second, the pointers. We live in a peculiar world. There are copiers, on one hand, which make it difficult to destroy incriminating evidence because one is never sure about how many copies were made, and where they are. There are computers, on the other hand, where conventional wisdom is the opposite — to keep only one copy of data, avoiding redundancy like plague.

Pointered databases are vulnerable to malice, just as one can alter history if there is but one record of events. All the data criminal has to do is cut the pointers, which often have no historical record elsewhere. Indeed, much computer crime is done in this way. If there is no way to access the data, or even know of its existence, what difference does it make if it still resides among millions of electronic bits? It matters little if data is correct or incorrect if it is not accessible. This is not a new concern of mine. At NordData '73 I asked "But what happens when we put perfectly valid data in? Can we get it out again?"⁵

The men that stole the physical database (and its backup copy) from Imperial Chemical Industries⁶ for ransom were amateurs. They were captured, and there was physical evidence. They would have been much more successful had they converted the pointer set via computer to an encrypted form and sold the key for ransom.

Data Invisibility

An underlying difficulty is that the primary state of databases is electronic, thus invisible. It is possible to bring that data to a secondary state by printing the data in the

⁴Los Angeles Times, 2 November 1979, part II, p. 1.

⁵R. W. Bemer, "Computers and our society," *Honeywell Comput. J.*, vol. 8, no. 1 (1974), pp. 49-54.

⁶ComputerWorld, 31 January 1977, p. 2.

stored form. But this is only a physical manifestation, as every programmer who takes a dump of the store knows. Unless the logical form is identical to the physical form in storage, this printing will be unintelligible.

To really see the true data requires printing it in the logical form. If this is not the same as the physical storage mode, as so often happens, certain programmed processes are required for transformation to real visibility.

The invisibility problem is expressed eloquently by Prof. Theodore Sterling, of Simon Fraser University in British Columbia, Canada.⁷ He says:

"We cannot help but express two concerns. The trend is toward replacing by electronic signals much of the visible record left from transaction between man and system. From the frequency with which individuals encounter errors and from observing the difficulty with which these errors are corrected, the question naturally comes to mind what will happen when the visible audit trail, such as checks, bills, and receipts, is replaced by electronic signals? We need to be especially concerned when these electronically managed transactions take place in great numbers at faraway places, thus increasing the distance between the individual and the transaction which may, in many ways, have important consequences for his life

"Another and perhaps even stronger concern is with present practices for designing large program packages. The tendency is to construct massively integrated systems of programs, something we have termed *global*, constructed with an eye toward cost effectiveness rather than flexibility and correctability. The result has been giant programs which turn out to be peculiarly resistant to modifications when conditions change and to the correction of unusual errors."

Sterling's studies revealed a new twist in consumer mentality:

"Cash transactions were preferred because it was felt that the gain in service by using credit would be countered by painful 'hassles' in the case of errors. Such observations form an interesting sidelight on the preference of consumers to return to a cash society, just the reverse direction of the business community."

Preventive Methods

A partial list is attempted here of some methods to prevent or at least reduce data incorrectness. It is not comprehensive, but it may suggest further thinking on such an important matter.

Entry Validation

A person should never be allowed to access a file directly, add to it or modify it, and put it away in altered form. All such actions should be taken only under control of a program that acts on the behalf of a database. These are called "surrogate" programs.⁸ They are processes that can talk to both the user and the operating system. They may be made ever more obedient and intelligent. Programmed well, they are the best and most capable of users, acting on behalf of less experienced human users.

Forms mode entry programs fall into this surrogate class. In most of them, the screen area has attached properties that will inhibit data from being entered if it does not have those properties. But this is not enough. Programmed control beyond this is

⁷T. D. Sterling, "Consumer difficulties with computerized transactions: An empirical investigation," *Commun. ACM*, vol. 22, no. 5 (May 1979), pp. 283-89.

⁸A "surrogate" is a person acting in place of another, a substitute or deputy. Often used in a legal sense, with good reason. If you have little knowledge of the law, you get a lawyer to plead to the judge and/or jury.

often necessary. What is permitted to enter in one screen area may also be a compound function of what is entered in another area. Much attention should be directed to checking all data entry and change for completeness, limits, range, and types.

Standards

ISO Technical Committee 97, Computers and Information Processing, has a subcommittee for the standardization of data elements. A logical outcome of this work could be the registration of standard databases, with identifier numbers. Other databases could then reference these numbers as a part of their own definition.

For example, I once worked on a database to reduce its size by replacing references to business entities with tokens (such as stock exchange letters). There were many references to Carlyle Compressor Corp., Division of Carrier Air Conditioning. There were also many references to Carlyle Compressor Corp., same address, but not mentioning the division status with Carrier. This can be solved with a co-file giving the token-to-address relationship, and then just putting (CAR1) in the database, assuming that (CAR1) and the address is a part of a standard and registered database.

Many national and international agencies are involved in making standards for data elements. It remains only to develop an international registry, just as one has been made for the ISO Code (CCITT) Alphabet 5).

Tools

Many programmed tools should be available to the database administrator. Among them:

- Ordering programs and duplicate checkers (a flagging of duplicates might have prevented Booth's death).
- Concordances to run against the database, to detect misspellings, variants, aliases, fuzzy sets (how does a search program know that Wohlegemuth, Wohlgemuth, and Wolgemuth are all the same person?).
- Subsetting and display of items of the same classes (I periodically extract all descriptors from a relational database, order them, eliminate duplicates, and then check visually for pairs that should not differ, i.e., the retrieval keys should be unique).
- Subpictures or profiles of the database (that's how I found a "software" descriptor missing for Capt. Grace Hopper).
- Surrogates that permit individuals to correct data about themselves.

Periodic Views

A database administrator has many prerogatives and privileges, not the least of which is unique knowledge of what the database contains, and how the data is extractable. This can put the administrator in a position to ask ransom, unless you have a database organization that permits as many concerned people as possible to inspect (part of) the database for correctness. This should be a periodic scrutiny, and the best people to do it are those directly concerned with the entry process.

Data safety and correctness depend strongly on the possibility to take multiple views of the data. An employee given a list relating to people might say:

No, there's no one named Fred Fortran in our department. Are you sure that someone from the data processing group isn't playing a trick? After all, Fortran is the name of a well-known programming language.

And Sally resigned last week. She's pregnant.

That man's name should not be on the list for welfare payments. He died two years ago.

An employee, given a list of what she sold last month (by item class, descending by amounts), might say:

Oh no, I didn't make any sale of \$120,000. I certainly would remember that if I had. Anyway, not in the perfume department.

What's this about an elephant gun? We certainly don't carry those in lingerie.

Audit and Attack

Here the fire drill principle should be exploited. Don't wait for the real fire to find out if you can escape or not. Instruct the auditors to play real war games — to subvert the database system in any possible way. When they break security, build program surrogates to deny those access paths.

Data Protection

Surely computer storage is getting significantly cheaper all the time. But let us not be hasty to pocket that quick and apparent profit. Let's invest some of it into carrying auxiliary data with each data element. Who made the element have this content, when was it changed, and who authorized the change? Carried to logical extreme, perhaps each data element should have the protection, via auxiliary data, that an entire file does in most of today's systems. Here is a sample catalog entry:

File Name — TODAY
Originator — Bemer
Data Created — 790123
Data Changed — 791207(12.445)
Last Date Accessed — 800512
Number of accesses — 877
Max File Size — 10, LLINKS
Current File Size — 5 LLINKS
File Type — Linked
General Permissions — R,E
Specific Permissions — None

This should be dynamic, to record *all* changes made since the last archiving. I know of no way to prevent incorrectness absolutely, but this is what it will take to make it amendable.

Purge

Assuming the associated data of the previous section exists, it is then simple for programs to be written to walk the data structures to purge, from the database, all data not meeting certain criteria. Data older than *n* years, for example, which has not been revalidated. In the case of Mr. Karagianis, the code "CW" was added to his record. By program, the same attempt at addition could have resulted in automatic purge.

Finally . . .

The technical literature of computers for the last three years shows a great number of papers about proving that programs operate correctly. Whether or not this is possible is the subject of wide debate. No matter. It is still desirable that they do operate correctly, however that may be achieved. Papers on the subject of proving data correctness are fewer. This is a pity, for many ills arise from incorrect data. Surely correct programs operating upon incorrect data will lose that useful property.

**PROCEEDINGS
DAY 2**

**STATE
OF THE ART
REVIEW
1980**

98



R W Bemer

Honeywell Information Systems
Phoenix
AZ

FILE MANAGEMENT FOR OFFICE AUTOMATION

SCOPE OF THIS PAPER

J Carlisle, President of Office of the Future, Inc, told the 1980 National Computer Conference in Anaheim, California (001) that 'the emphasis in office automation systems ... should fall on services that allow managers to retrieve needed information quickly and easily from a database rather than through expensive and often wasteful 'phone calls'. This paper gives an organised accumulation of some file management techniques needed to reach such a goal. It considers:

- 1 Surrogates: the intermediary control programs for all user-initiated processes.
- 2 Acquisition: purchase, trade, entry, or conversion.
- 3 Storage methods: random versus sequential, redundancy.
- 4 Storage media: visible permanently or on demand.
- 5 Archival and purging methods: single or multiple levels of storage, aspects of perpetual hoarding.
- 6 Retrieval methods: inverted files and storage costs, synonyms and fuzzy definition, single queries versus subpictures, update and change synchronisation.
- 7 Maintenance and reliability tools: surrogates, assurance of data completeness and correctness.
- 8 Human factors: personal profiles of interest, usage assistance.

SURROGATES

One way to get the 'feel' of an automated office is to pull all of the file folders out of the cabinets, strew them haphazardly on the floor of an out of the way room, and lock the door. Drill two holes for access by infinitely long arms, and try to find what you need. If you do not have the lock, a blindfold will do.

Gone are your former aids, the coloured tabs, the assignment to a certain emergency drawer, your secretary who knew how to find it if you did not. Perhaps a total loss would not occur, but even a 10% loss is likely to hurt your business. So what you need for that locked room is an 'inside man' - an electronic inside man.

He is called a 'surrogate', from human usage, where one person acts in place of another, as a substitute or deputy. The concept is often used in a legal sense, with good reason: if you have little knowledge of the law, you get an attorney/barrister to plead to the bench.

In the days of closed computer shops, programmers acted as surrogates and intermediaries for the real users, presenting the problem to the computer on behalf of the clients. Only the programmer was audited, and security was more controllable. With direct access, as in timesharing, the human surrogate is seldom used now. Users have access to as many system facilities as they care to learn from manuals or by talking to other users. This is a safety and security hazard for office automation. It also requires special instruction, which is bad for casual users.

It is better to interpose surrogate programs which, being files, have various protections and access privileges. Their names become the vocabulary of the user. When called, they query the client as to what he wishes to do, and then bid his commands against the database. There is no need to learn a special jargon, and the system is more orderly and controlled. These surrogate (a term to be preferred to that of 'shell') programs may be made ever more intelligent and obedient, and while executing they meet the auditing and recording needs of the office.

No file management action (creation, inspection, modification, or destruction) should occur except via a surrogate. This ensures the security and integrity of your files and provides protection against even your own database administrator.

ACQUISITION

Data elements and tokens

I once modified a copy of Underwriter Labs' (the US product safety guarantor) database of products and manufacturers. The purpose was to reduce its size by replacing the business and address elements with tokens (such as stock exchange symbols). (CAR8) is a typical token, standing for the eighth location of Carrier Airconditioning Corporation. The database was reduced to one-eighth the original size, even though a new co-file of token-to-address relationships was created. Needless to say, not all strings replaced by any one token were identical in the original. There were misspellings, bad updates, and random variations, all things to be found in any database not so controlled.

There is much standardisation work in progress on the subject of data elements. States, countries, oceans, ports, zip codes, time, longitude, latitude, and business entities are all the subject of encoding. Because the registry method I proposed for variations in the ISO Code (ASCII is one) has worked out well, I would propose that the registry method could apply to standardised databases, each with an identifier number. Other databases could reference these numbers as a part of their own definition. This was proposed informally at the 20th Annual Meeting of CODASYL, May 1980, and appears in (002).

Second party supply

An office generates most of its own database, but there are still external sources of

data that it could use. Stock market statistics, laws and regulations, housing inventories, monetary conversions, weather predictions, and trucking schedules and capacity are but a few that come to mind.

Many such databases are now on the market for on-line interrogation, via their own file management system and structure. If, however, they were to be acquired for integration with the file system of an automated office, they should be obtained only in flat, linear file structure. Otherwise the file structure is built in, and would be difficult to remove. Also, an update method must be agreed between supplier and user.

Conversion

No matter what the source for your office automation database (even your own current electronic files), some aspects of conversion will be involved. The matter of alphabet encoding is easy, as from EBCDIC to ISO Code because you are now in the field of communications and distributed databases. Word processing may be the biggest stumbling-block because of variations in file structure and those invisible embedded codes that were never standardised between even two manufacturers. Be patient - the new text processing languages will ease this problem, once the generic format identifiers can be agreed upon.

STORAGE METHODS

The file attribute file

The typical file attribute file of DP systems contains primarily external information, such as:

- Name of the file
- The originator
- Date of creation
- Date and time of last change
- Date of last access (reading or processing)
- Number of accesses
- General permissions
- Specific permissions
- Storage requirements (in units such as bytes)
- Maximum size to which file can be expanded
- Type of file storage mechanism
- Alphabetic encoding method (ASCII, EBCDIC, Kanji, etc).

General permissions are those granted to anyone, from among options such as READ, WRITE OVER, APPEND, COPY, etc. Specific permissions are those granted to specific users, over and above the general permissions.

The automated office needs to be concerned also with attributes and records on partial content. The file attribute file should contain:

- The capacity to expand 'date of last change' to an audit trail of each change, with the person causing the change, and their authorisations

- Persons who should receive mail message notification of such change
- The capacity to expand 'date of last access' to an audit trail of who looked at it
- Semantic typing of the file (whether it is a letter, announcement, design history, minutes, calendar, schedule, or a ranking of the importance of the document to the business)
- Keywords for the content (names, document identifiers, cross references, descriptors, customer group etc).

This process can become tedious, even self-defeating, if applied totally for all electronic documents. Many of these elements must be permitted to be null for certain classes of documents. For example, if the general permissions include all options, there is no need for a specific permissions entry. Or if the file is an announcement of the company picnic, there is little need of an audit trail of changes.

Intrafile attributes and protection

An audit trail for change can be applied to specific elements in a file. Examples of elements that could well require this protection would be the figures in the company's annual report. Of course such a critical document should never be trusted to a single copy for protection; this is a case where eliminating redundancy is not a desirable aspect of DP systems. One or more copies should be made and stored under different names. History is changeable if there is only one copy of the history books, so it is wise to distribute copies for protection.

Linear file and co-files

The file management records described will suffice for random as well as sequential files, but it is the opinion of the author that the office will be more comfortable with linear files only. They can be displayed by the simplest of methods. The present speed of computer operations permits rapid formation of co-files, which may be either a single file in a different order for usage, or combinations of elements of several linear files for operations on sets (relational databases).

A known useful property of linear files is line number, either implicit or actually attached. This may be expanded to include more than line number. The prefix may include tags for priority, type, origin, security, or any other purpose.

STORAGE MEDIA

Carlisle (001) also had some words for exponents of 'paperless offices'. He said they are about as realistic as 'paperless bathrooms', and that 'paper records are part of the well-entrenched patterns people have developed for being effective, and you just cannot suddenly take those patterns away'.

It may be true that the paper to electronic ratio will decrease, but somewhere between the chiselled words on statues and satellite data there is a varying line of demarcation between always visible text and visible on demand text.

The 'paperless' exponents seem to forget the distinction between *targeted* information and broadcast information. When mail is sent to a single person, or small class of people, it is targeted information. The receivers are informed unwillingly, and must make deliberate access without benefit of descriptors (in the YOU HAVE MAIL message) or advertising.

The hard copy (paper) audience is the broadcast audience. A sender has no particular way of knowing who will turn the dial slowly and, hearing a useful melody, sharpen the setting to hear it to the end. The video terminal moves the data relative to the eye. In contrast, the eye moves relative to the paper, and paper will win when the interest is low.

It would also be difficult to reorganise a file access pattern as quickly as paper sheets can be reordered. The best solution is to develop methods whereby paper can coexist comfortably with electronic files. The first step might be an optical reader. If you had locations without direct access to your system, paper would still be advisable, as the OCR reader could convert it after it arrives by mail.

Another visible medium which may prove to be as hardy as paper, and should even thrive in the automated office, is microfilm. It currently costs \$0.10 a copy, \$2.50 for the master. Hardly a system today can put that many words on a screen at that price, and none may approach that cost for a decade. Moreover, a microfiche gives access to many pages with little effort; the arm movement is less than typing in a query. Data can be retrieved in context, to scan back and forth, or the same data can be placed in multiple contexts. If the contents of microfiche are generated by COM from a database, many different organisations can be made for ease of use, eg, a record of all mail and memos on a topic until final action is taken.

ARCHIVAL AND PURGING METHODS

We are often told that storage is getting ever cheaper, that bubbles and videodisk will make the cost virtually zero, so that we can keep everything forever in direct access storage. Such optimists imply that there will be no future need for microfiche or paper or other semi-permanent forms of display and recording. This is hard to believe for the following reasons:

- 1 In some cases the law says 'no'. Much of the content of this paper is directed to methods for preserving the integrity of electronic databases, so there must be some possibility of subversion. Books have been published on computer crime, which depends upon alteration of databases.
- 2 Eventually the pointer system, which separates the current and valid data from the debris, becomes congested and costly to run. This is called information pollution. Debris includes not only obsolete and historical data, but also erroneous programs, test datafiles, traces, dumps, etc. It depends upon programs, which might have bugs in them. If files are deactivated by changing the pointer mesh, then the mesh must be changed by program, at which time a power outage or new bug might make *everything* into debris.
- 3 Given a large enough database, just getting to the right data via the pointers can be quite costly.
- 4 The physical pointer system is not the only pointer system needed. There are the

pointer meshes that reflect the organisation of the subset of data of interest to each office worker.

- 5 Some purging is in the interest of security and protection against malfunction. Wrong and superseded data is better not left in the system after some period of time, less accidental access to it might lead to belief in it. Suppose the boss changes his mind, and you find the next to last version of a directive and follow it? Your surrogate programs have the knowledge to 'patrol' the database structure to purge all files not meeting certain requirements or criteria. Is it likely that a programmer is still at the same address, if the recorded data is over 10 years old?

RETRIEVAL METHODS

Organising for retrieval

D Beer (003) and others say that true information retrieval is being able to ask questions without predefined answers. Beer faults certain retrieval packages because they require specific addition of a group of keywords, and the search is made only on these keywords. He prefers no addition, and a search on full text content, after subtracting out a stop word (noise word) list. This view could be supported if the content always contained the associative material, but often it does not. A person's mail address and telephone number does not indicate that he was once a house guest of mine. The workable compromise seems to be to add keywords or descriptors that seem appropriate but are not in the original entry, which did not contain that semantic information.

Full text or inverted files

The proponents of full text search are probably quite right when it comes to searching something like a database of law and regulations. One is unlikely to know exactly, during the storage process, what is to be searched for in the future, particularly because of the language used, which is likely, experience shows, to be confusing or perplexing to any but the originators.

Most office documents, while also likely to contain ambiguous and stilted language, are much easier to categorise for future retrieval, provided a comprehensive classification is given, with good rules and suggestions for indexing. In this case, inverted files are easier to use (and visualise) than pointers. They require more electronic storage, but those costs have decreased. E Codd says that the payoff for the extra storage for set-oriented (relation) models turns on the nature of the user's application.

The inversion surrogate

The difference between full text search and inverted files is minor. The inverted file is best made by a full text search at entry and change time, so this could be called preplanned full text search.

There must be a way to enter any document into storage. This should be done only under programmed control, where the program surrogate puts the copy into permanent store. It also takes all the proper steps to permit the document to be retrieved properly and

The 'paperless' exponents seem to forget the distinction between *targeted* information and *broadcast* information. When mail is sent to a single person, or small class of people, it is targeted information. The receivers are informed unwillingly, and must make deliberate access without benefit of descriptors (in the YOU HAVE MAIL message) or advertising.

The hard copy (paper) audience is the broadcast audience. A sender has no particular way of knowing who will turn the dial slowly and, hearing a useful melody, sharpen the setting to hear it to the end. The video terminal moves the data relative to the eye. In contrast, the eye moves relative to the paper, and paper will win when the interest is low.

It would also be difficult to reorganise a file access pattern as quickly as paper sheets can be reordered. The best solution is to develop methods whereby paper can coexist comfortably with electronic files. The first step might be an optical reader. If you had locations without direct access to your system, paper would still be advisable, as the OCR reader could convert it after it arrives by mail.

Another visible medium which may prove to be as hardy as paper, and should even thrive in the automated office, is microfilm. It currently costs \$0.10 a copy, \$2.50 for the master. Hardly a system today can put that many words on a screen at that price, and none may approach that cost for a decade. Moreover, a microfiche gives access to many pages with little effort; the arm movement is less than typing in a query. Data can be retrieved in context, to scan back and forth, or the same data can be placed in multiple contexts. If the contents of microfiche are generated by COM from a database, many different organisations can be made for ease of use, eg, a record of all mail and memos on a topic until final action is taken.

ARCHIVAL AND PURGING METHODS

We are often told that storage is getting ever cheaper, that bubbles and videodisk will make the cost virtually zero, so that we can keep everything forever in direct access storage. Such optimists imply that there will be no future need for microfiche or paper or other semi-permanent forms of display and recording. This is hard to believe for the following reasons:

- 1 In some cases the law says 'no'. Much of the content of this paper is directed to methods for preserving the integrity of electronic databases, so there must be some possibility of subversion. Books have been published on computer crime, which depends upon alteration of databases.
- 2 Eventually the pointer system, which separates the current and valid data from the debris, becomes congested and costly to run. This is called information pollution. Debris includes not only obsolete and historical data, but also erroneous programs, test datafiles, traces, dumps, etc. It depends upon programs, which might have bugs in them. If files are deactivated by changing the pointer mesh, then the mesh must be changed by program, at which time a power outage or new bug might make *everything* into debris.
- 3 Given a large enough database, just getting to the right data via the pointers can be quite costly.
- 4 The physical pointer system is not the only pointer system needed. There are the

correctly upon demand.

The surrogate entry program is called and given the name of a document to enter, which is in an immediately accessible file. The surrogate calls the list of keywords or other descriptors to another immediate access file. It cycles down the list, picking up each word in turn. Pointing then to the document copy, it searches for the keyword. If it cannot be found anywhere, it returns to the keylist and deletes that word, then picks up the next. The process is then repeated. If it is found, the surrogate brings the file which is named by the keyword to immediate access, and adds to it a line entry which is the document identification, with the number of times the keyword was found in it (this is for threshold access). On exhaustion of the list, the surrogate is left with a list of all keywords found for that document.

So far it has been assumed that no new keyword would be placed on the list. One may add a new keyword arbitrarily, or because it has not appeared before (such as the identification for a new document). The surrogate has optional ways to find the second case. A recommended method is an initial modification of the document to individual words, which are then sorted in the same order as the keyword list. A noise word list is then subtracted from this new list. The remainder is then subjected to the keyword search. Keywords are deleted as found. The remainder is displayable for the librarian. New keywords are selected, and that remainder contains candidates for the noise word list.

The editing surrogate

Beer (003) says that 'the user may amend any document in the database. If a permanently 'registered' document is amended, the system will amend all files automatically'.

The rudiments of such editing surrogates are to be found in the 'OOPS' command of several text editors, which implies that all seemingly direct actions are actually indirect and filtered by program. The link between modified files and the inverted keyword files may also be handled by an editing surrogate.

The inverted files created by the previous process reflect the document at first entry, but it may be modified by text editors. The editing surrogate should store a copy of the file before editing. When a resave of the modified file is requested, the versions are subjected to a logical comparison for set membership. The file 'old-and-not-new' is tested for keywords. Any found have their inverted file appropriately reduced. The file 'new-and-not-old' is tested for keywords to see if any should be added to the keyword list, creating an appropriate new inverted file.

MAINTENANCE AND RELIABILITY TOOLS

Managerial extraction of information from data requires that the database be as complete and correct as possible, and that the possible access paths are always taken. The time to control these factors is before, not during usage. Some computer operating systems automatically run hardware diagnostics when sufficient capacity is idle; it might be useful to run data fire drills as well.

C Kossack, reviewing a paper on medical information practices (004), has a prophetic caution for the automated office:

'I have been depressed by the frequency with which I have encountered sophisticated computerised information systems, only to find that their files are no longer current. The energy, talent, and devotion required to keep such files current, although initially available in adequate supply, apparently erode with time, and the files become of only historic interest.'

Data completeness

Here the concept of subpictures of the database comes into play. Database completeness can be controlled by periodically running a program 'profile' for some Boolean specification of descriptors and keywords. A profile for 'US' and 'navy', with several minor descriptors, showed a small subpicture of a database. There was no hit for Capt Grace Hopper in the 'software' column. A hurried addition was made.

For informative-only databases about people, one way to ensure both completeness and correctness is to permit the individuals themselves to have surrogate-controlled access to correct their own data. This would not include the payroll file or biographies.

Data correctness

For data elements that have range limits on values, or type restrictions, surrogates can monitor the values upon entry. The field control capabilities of many forms entry programs are effective in this way, but they can reflect only a single parameter control, not a combinatorial one. The latter must be programmed into surrogates that scan some database subset periodically.

It is useful to run concordance programs against the databases, to detect misspelling, aliases, and fuzzy sets. It is better to do this as an off-line process, for perspective, rather than build the mistake detection into the working searches.

Data accessibility

Whether data is correct or not is of little importance if it is not accessible. The data dictionary is indispensable, but so is a process dictionary. That is, some files contain programmed processes for the office. Their existence must not be hidden from the user.

Factories have toolcribs for tools not in continual use, thus not needing replication for each user. The automated office needs such a toolcrib. There must be mechanisms for deposit, cataloguing, inventory, and coalescence (making multipurpose tools).

HUMAN FACTORS

Profile of interests

Just as many offices require identification before admission of visitors, there should be no casual users of office automation systems. Each should be registered. With such registry, a substantial amount of identifying information is requested, which cannot fail to be helpful to the user. It is used to build an associated profile file of

characteristics peculiar to that user, such as:

- Personal identifier to the system
- Primary password for initial system access
- Access permissions: what the user is permitted to see and use in the way of files, programs, tools, and major subsystems
- Native language preference for system communication
- Skill level for system communication
- Automatic start-up conditions, ie systems and modes in force and such aspects as terminal characteristics (brightness, contrast, audible key depression, alarm intensity, etc).

To this general profile, it should be possible to append particular preset profiles for specific applications and usages:

- Native language preference for the application
- Skill level for this application (novice, intermediate, and expert will usually suffice - if there is confusion at any level, a question mark response drops back to the next level of understanding).

Native language is stressed because it is useful for safety and understanding. It is usually quite easy if one uses only the words that map one-to-one for many languages. Caterpillar Tractor and International Harvester are among the many organisations that use automatic translation for instruction manuals around the world.

Retrieval display

To antagonise the office worker entirely, obtain the retrieval system from a DP vendor. Most of them have a peculiar punch-card flavour that is abominable to people. Some systems, for example, will tell you that they found 000000028 such documents, not reiterating what 'such' meant. The command formats given by Beer (003) are also a bad human interface. From his examples:

Q Electronic	Q=query
14	Means 14 documents somehow about electronics
A Manufacture	A=and
1 72	Means 72 documents have either, but only one has both descriptors

The responses are dependent upon placement or layout. The second command implies the existence of the first. The construction of responses that are readable by and acceptable to humans is actually a trivial exercise; it merely needs to be carried out. That is why the most important part of any computer-based system is the dialogue processor.

REFERENCES

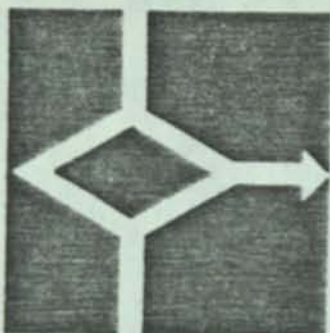
001 *Automated office seen irrelevant
to executives*
Computerworld p 24 (6 Feb 1980)

002 BEMER R W
Incorrect data and social harm
DATA Magazine Copenhagen Denmark
(To appear)

003 BEER D
Rep in Computing (24 Apr 1980)

004 KOSSACK C F
In CR 35942 Computing Reviews
vol 21 no 3 (Mar 1980)

1961



CONGRESSO ANNUALE
ANNUAL CONFERENCE

A.I.C.A. ASSOCIAZIONE
ITALIANA
PER IL CALCOLO
AUTOMATICO

1981

PAVIA 23-25 Settembre 1981

atti
volume I



Programma per l'automazione dei
processi di lavoro in un ufficio

Aut. 10000
Società per l'Automazione dei Sistemi
Programma, Bologna, 1968

Il presente lavoro è stato realizzato nell'ambito di un progetto di ricerca finanziato dalla ANP (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive) e dalla ANS (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive).

AUTOMAZIONE DELL'UFFICIO

1. Introduzione

Il presente lavoro è stato realizzato nell'ambito di un progetto di ricerca finanziato dalla ANP (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive) e dalla ANS (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive).

2. Obiettivi del progetto

Il presente lavoro è stato realizzato nell'ambito di un progetto di ricerca finanziato dalla ANP (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive) e dalla ANS (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive).

Il presente lavoro è stato realizzato nell'ambito di un progetto di ricerca finanziato dalla ANP (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive) e dalla ANS (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive).

Il presente lavoro è stato realizzato nell'ambito di un progetto di ricerca finanziato dalla ANP (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive) e dalla ANS (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive).

Il presente lavoro è stato realizzato nell'ambito di un progetto di ricerca finanziato dalla ANP (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive) e dalla ANS (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive).

Il presente lavoro è stato realizzato nell'ambito di un progetto di ricerca finanziato dalla ANP (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive) e dalla ANS (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive).

Il presente lavoro è stato realizzato nell'ambito di un progetto di ricerca finanziato dalla ANP (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive) e dalla ANS (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive).

Il presente lavoro è stato realizzato nell'ambito di un progetto di ricerca finanziato dalla ANP (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive) e dalla ANS (Associazione Nazionale per lo Sviluppo delle Imprese e delle Attività Produttive).

Problems and Solutions for Electronic Files in the Office

R. W. Bemer
Honeywell Information Systems
Phoenix, Arizona, USA

Abstract

Many solutions promoted for the expected problems of office automation may be premature or ill-advised, although this is often denied, leaning on the tenet of "conventional wisdom". This paper examines some alternatives on such topics as file structure, forms mode, full buffer transmission from screens, and scrolling. The economic and the human tradeoffs are examined.

Some Suspicions

For a field as new as office automation, there seems to be already a body of accepted wisdoms that may deter success if not restudied and subjected to constructive criticism. It is my intention to begin the questioning.

Viewing Electronic Files

Sometime this year a John Wiley series book by Bruce Christie has appeared, or will appear, entitled "Face to File Communication". I commend the phrase, and envy the invention of it. This is one of the key aspects of office automation, which field could well founder if this subject is not treated well. I would make only one suggestion about the title. Perhaps it would be better to consider the triangle "file to face to fingers to file", to close the loop neatly.

It would be wrong to think of electronic files as all representable on A4 paper with characters spaced 4 per centimetre. Such is the influence of the non-SI United States that common data processing output throughout the world, via line printers, is 66 lines of either 132 or 136 characters (sometimes 160). Such pages cannot be viewed in their entirety on screens having 24 lines of 80 characters. And so manufacturers are developing screens with higher resolution, and sometimes greater size.

Now I ask a perhaps heretical question. Is this the correct way to go? If manufacturers will persist in thinking of forms orientation, with a rigid physical linkage between the files and the areas of a screen, I am certain that it is not. We all see exemplars of very wide physical documents in the office — Gantt charts, yearly summaries with several entries per month, etc. And documents longer than than 66 lines.

The proper way to view such files is by "windowing" — simulating a window moving freely over a printed document of virtually any size. The programming of windowing is not an easy task. The leftmost display position on the screen must be computed to cater for (imagined) characters to the left of the screen, an area that may contain accented characters and underscored words (accomplished by backspacing), or horizontal tabulation controls, or nonprinting (and thus nonspacing) characters. The payoff is high, however. One benefit is that we may view files as though the window were the eye.

Rex Malik says [1] "It is beginning to look suspiciously like a fact that the 80-character line has some inbuilt characteristics to it which come out of the physiology of eye/brain coordination and that it naturally fits our ability to process substantial amounts of information at any one time." In fact, the evidence has been with us a long time. An axiom in typography has been to limit lines to "two and a half alphabets", or 65 for the Roman alphabet.

A system called "Square Span" was proposed about 40 years ago. It involved placing text within approximately square outlines. One would read across a few boxes and then drop to the next row of boxes. It was designed for fewer and more regular eye movements. Reverse the pattern to read down, and you have newspaper columns, which are in no danger of becoming extinct in form when they become electronic files to be read via Viewdata or Telidon!

A great benefit of the windowing method arises from the fact that the controlling program must know continuously where the window is placed relative to the file or document. Thus we may have direct access to any field or item in the file without the need of pointers or other database access methods. Imagine viewing data records 1000 characters long, windowing right to some desired field, and then studying that field throughout the file. Imagine how easy it would be to update by direct alteration.

Moving the View

So far I have discussed only the stationary view, not what the eye sees as the view is changing to the next position. Werner Buchholz wrote a famous paper entitled "Fingers or Fists" [2], a study to find out whether computer processors should be primarily decimal or binary internally. So far, the fists seem to have won.

A corresponding question for displays (although perhaps not so momentous as the other) is "Papyrus or Books?" This is a question for the human factors of VDU terminals, specifically the matter of scrolling. One might think that books would have won, just on the basis of evolution in the printing industry.

But no, scrolling seems to be another "conventional wisdom" to designers of video terminals. Possibly they saw how hardcopy terminals used continuous-form paper, either rolls or fanfold. In the video world, nothing could be further from the truth. For every usage one knows or can imagine, a page approach is possible — and usually is better. Scrolling merely permits me to introduce a new factor with an acronym — the Tough-On-The-Eyes factor (TOTE).

Scrolling capability, although useful, should not be used for the initial creation of a page of display. The simple distinction to be made is whether or not a display of text has already been read. If not, it is very difficult to read (especially at high transfer rates) if each new line appears at window bottom and forces a scrolling up. If a page is displayed by writing from the top down, the reader may choose a granularity of stable text for his reading, even if some chaotic action is taking place further down the screen.

However, if the text has already been read, it is tolerable to do some disjoint scrolling to effect a rearrangement of the screen with the purpose of not rewriting lines that already exist there (insertion, deletion, cut, and paste). While this is being done, the viewer can maintain a mental picture of what is there, having already read it.

The following are basic operations for manipulating files with video terminals. They can all be done by filling the screen from the top down, rather than by scrolling up from the bottom of the screen.

- o Moving back and forth a number of screen "pages" in a file.
- o Finding a place in a file by text content.
- o Moving the screen window right or left for a file wider than screen width.
- o Insertion of lines of characters, or "building".

Viewing Multiple Electronic Files

The "conventional wisdom" seems to be to show multiple files in different sub-areas of the screen. I'll be bold enough to challenge the ergonomic reasons for always doing so. An easy way to test the urgency for split screens is to imagine different instances of working with two pieces of paper, deciding in which of those instances it is imperative that the two pieces be side by side, in simultaneous view. I think you will find that in many instances the human memory is retentive enough to be able to reference them separately.

There are some instances when split screens, or multiple windows, are useful on a permanent assignment. In general, however, a smaller display detracts from the information content available. I prefer to use more than one window on a dynamic basis only, in an overlay mode, restoring the original contents at the option of the user. The arrival of electronic mail is a good example. While reading it, the part that was overlaid is of little concern. If this seems a trivial thing, remember that the overlay size can be variable, not a fixed portion of the screen.

In those cases where permanently split screens are a necessity or benefit, I very much prefer splitting a screen vertically rather than horizontally, when the aspect ratio is nearly 1:4 (20 lines by 80 columns). Again, the newspaper-type column, particularly for menus and operating directions. Otherwise the split enthusiasts should consider using two separately-driven screens.

Economics of Viewing Files

For files and messages that are only for reading, there are few economic, but many ergonomic, considerations. The characters must be transported from residency to display. There is some room for economy in expanding long spaces at the display, rather than sending them as groups of space characters. Embedded tabulation characters help. But big gains are not there.

In contrast, when displayed files are to be changed, added to, or annotated, there are big economies to be made over today's usual technology. When video terminals were introduced for file manipulation, the "conventional wisdom" was that an associated buffer, in the terminal, would reflect the current state of the screen. Periodically that buffer was to be sent to file residency and replace what was there previously. The file and the picture of the file would then be resynchronized.

It did not take long for the industry to recognize that this led to extreme and expensive overloads. If, for example, one wished to put a copy of the top line on the screen just below it, all lines in the buffer from that point on have been changed, and the buffer must be sent again. That is too expensive, when the microprocessor, in the terminal, can effect the change itself and send only the information that it did so (probably 4 characters).

IBM met this problem with the 3270-type terminal, where fields are defined by pre-empted positions on the screen, and only the changed fields are transmitted. It works fine for forms, but not for free text (as in a letter); certainly not for windowing!

Duncan MacGregor (Honeywell Ltd. in Toronto, Canada) and I have solved the problem by using terminals in a half duplex asynchronous mode, in cooperating processes with the microprocessor. Function keys are assigned to processes (in many cases using numeric tags). The microprocessor does what it can do, and only the function key codes and editing keystrokes are transported to residency. There another processor modifies the file to match the picture on the screen.

What or Who Controls the Viewing?

A major question in viewing electronic files concerns modification and data entry. The "conventional wisdom" opts for "forms entry" modes. The forms entry mode program is the driver or controller; it controls the user. I am not sanguine. Data entry can be done by query; the form itself does not have to be displayed for data entry. Philips has a new data entry terminal with only a single line of 40 characters displayable — enough for the question and for the answer. The answers can be placed by program in the real form, and if that happens to be larger than screen size, one can window to view it when filled in. Or even change it. It is more important to control the data that is entered than to attempt to put it directly in place immediately. Few forms entry screens will reject 1982-02-67 in a numeric field.

The larger payoff in office automation will come from the added information that management can derive from a static amount of data. This is not a zero sum game. If the executive can view easily all of the pertinent elements of some subset of electronic files, then he can make better decisions to conserve resources. He must be able to drive the system from the outside in, uncontrolled by some program written by someone that does not have his knowledge and judgment.

Summary

Effective use of electronic files in the office will require that the user be permitted, and have the facilities, to access those files in every way that was possible with paper files — PLUS access to all the structure knowledge for those files. All this must be done economically in both equipment cost and elapsed user time. The user must have the tools available to select and limit the amount of data that must be displayed.

REFERENCES

- [1] Rex Malik, "A new medium after McLuhan", Computing, Vol 9, No. 21, 28, 1981 May 21.
- [2] W.W. Buchholz, "Fingers or Fists?", Commun. ACM 2, No. 12, 3-11, 1959 Dec.

Computing prior to FORTRAN

by R. W. BEMER
Honeywell Information Systems
Phoenix, Arizona

ABSTRACT

The life of the programmer in pre-FORTRAN days is characterized in modern terminology, indicating how strongly FORTRAN has changed the programmer's condition and working habits.

Computing prior to FORTRAN was a very different world than it is today. The early computing era was characterized by a lack of programming languages, and the programmer's life was a constant struggle to make the most of the limited resources available. The early days of computing were a time of great challenge and opportunity, and the programmer's life was a constant struggle to make the most of the limited resources available.

TABLE 1—Paper distribution by author last location

	1957	1958	1959	1960	1961
Author	10	20	4	12	20
				12	10

COMPUTER DESIGN & EDUCATION

The first programming, and in fact a few schools. When you read a programmer's story, you don't get a degree in computer science right with you. You get the programmer's story, but you don't get a degree in computer science. The programmer's story is a story of a man who has spent his life in the computer business, and he has a lot of experience to share with you.

A lot of people say that programming is not really taking the life out of the world. It's just a matter of getting the computer to do what you want it to do. But the programmer's story is a story of a man who has spent his life in the computer business, and he has a lot of experience to share with you.

Although there may not have been enough educational facilities to support specific demand, the education people felt at that time was a good one. The computer business was just getting started, and the programmer's story is a story of a man who has spent his life in the computer business, and he has a lot of experience to share with you.

FORTRAN PROGRAMMING

Programs were written in a "hard-wired" program. The code was written in a language that was not very different from the code that was written in the early days of computing. The programmer's story is a story of a man who has spent his life in the computer business, and he has a lot of experience to share with you.

For the CDC the program was extremely at the time. The programmer's story is a story of a man who has spent his life in the computer business, and he has a lot of experience to share with you.

1. Introduction
2. Computer
3. Program
4. Program
5. Program
6. Program
7. Program
8. Program
9. Program
10. Program
11. Program
12. Program
13. Program
14. Program

The included names of early programming language has been covered by many authors to indicate a general history of the language and its evolution in more general terms.

REFERENCES AND PUBLISHED PAPERS

References to published papers in pre-FORTRAN days were for the most part, those that were used in the "old" days. The references to published papers in pre-FORTRAN days were for the most part, those that were used in the "old" days.

During the middle of the paper included in the table are

TABLE 1—Paper distribution by author last location

Year	CC	1957	1958	1959	1960	1961
1957	Author	10	20	4	12	20
1958	Author	10	20	4	12	20
1959	Author	10	20	4	12	20
1960	Author	10	20	4	12	20
1961	Author	10	20	4	12	20

The 25 years since the introduction of FORTRAN covers most of programming as we know it, certainly in volume of usage. To minimize any possible communications gap, I have chosen to describe how it was before that watershed event by means of some of the terminology and buzzwords of today:

1. Conferences and published papers
2. Computer science education
3. Stored programming
4. Structured programming
5. Program portability
6. Performance measurement
7. Communications and timesharing
8. Compilers
9. Data independence
10. Software piece parts
11. Software packages

The technical history of early programming languages has been covered by many authors (it became a popular subject), so I'll confine my contribution to more general areas.

CONFERENCES AND PUBLISHED PAPERS

Publication of software papers in pre-FORTRAN days was far less prolific than now. And it wasn't yet "software." Papers on software techniques prior to FORTRAN are given,²⁻⁴² as found (mostly) in Youden's "Computer Literature Bibliography 1946 to 1963."¹ They're given in best chronological order. To avoid duplication, sources with multiple papers are referenced separately, and the individual papers are given decimal notation.

Doing an analysis of the paper content of the early Joint

TABLE I—Paper distribution of early JCCs

Year	JCC	Hard-ware	Appli-cations	Soft-ware
1951	Eastern	16	2	0
1952	Eastern	26	0	0
1953	Western	8	11	0
1953	Eastern	18	4	1
1954	Western	8	14	0
1954	Eastern	9	7	2
1955	Western	6	16	1
1955	Eastern	6	9	1
1956	Western	18	10	6
1956	Eastern	29	0	0
1957	Western	28	4	3

Computer Conferences (the only continuing national meetings of that era) yields the counts shown in Table I. The last entry is the meeting at which FORTRAN was presented.

The summary pre-FORTRAN count is that of Table II.

TABLE II—Paper distribution by conference location

JCC	Hard-ware	Appli-cations	Soft-ware	H/A	H/S
Eastern	104	22	4	4.7	26.0
Western	68	55	10	1.2	5.5
Total	172	77	14	2.2	12.3
%	65	29	5		

COMPUTER SCIENCE EDUCATION

This was just starting, and in just a few schools. When you hired a programmer then, you didn't ask about a degree in computer science; there weren't any. IBM used its Programmer's Aptitude Test as one screening method, and it worked somewhat, but people had a tendency to read more into it than was warranted.

A lot of us had our own pet questions, for we were taking them off the street. Magazine writers were curious about how one became a programmer. Dave Sayre had been a crystallographer, and Sid Noble and Art Bisguier were hired when I, an ex-movie set designer, advertised for chess players.

Although there may not have been enough collected theories to support specific degrees, the university people were all busy creating courses. The summer sessions at MIT and Michigan brought many practitioners together. Language processors were being built there and at Purdue, Pennsylvania, Carnegie Tech, Case, UCLA, and many others.

STORED PROGRAMMING

Programs have always been "stored programs." The only difference is in where they were stored. In desk calculator days—in our heads. To program the IBM 601, one had to file notches in a phenolic strip, and they were stored in a box or hung on the machine. The IBM 604 was programmed by wires placed in plugboards, and often we stored them for reuse, if they were general enough. More often they were unwired for a new program (I wired about 700-800 60-step boards for the 604).

For the CPC the program was obviously in the cards. Bob Bosak and I devised a card system with 4 different tracks of 3-operand instructions, and so could feed a deck of cards continuously in a loop.

STRUCTURED PROGRAMMING

Structure in programs is generally ascribed to Wilkes, Wheeler, and Gill,⁵ in their book on programming for the EDSAC. The subroutine was the first element of structure, and was generally accepted by programmers, particularly those writing interpretive systems.

We had no DO UNTILs or semaphores at our disposal, but many programs had a structure that's all but forgotten now. It was called "optimum programming," a method of placing sequential instructions just right on a magnetic drum, so they would be ready to read just after the previous instruction was completed.

PROGRAM PORTABILITY

The first way used to reconcile the differences between two types of computer was to recode the problem. The second way was to write a programmed interpretive emulator for one machine in the code of the other. When this resulted in performance degradation of 100:1 up to 1000:1 it lost a certain amount of favor.^{43,44}

The third way was to use the source language of the interpreter and write another interpreter for the second machine. This had some success, because the degradation was often not very high (except for extremely dissimilar machines), and it could even run faster! Several of these were made.⁴⁴ If machines of today's speeds had suddenly been introduced then, this may have become commonplace; compilers might have a different role. Even now, after thousands of compilers, interpreters still enjoy a considerable vogue. The fourth way, with different compilers, did not to my knowledge receive substantial usage until FORTRANSIT, and even there the portability path from a 704 to a 650 was difficult because the 650 supported fewer index registers.

PERFORMANCE MEASUREMENT

Although no hardware instrumentation was available for probes, much performance measurement did occur. It was vital because the computers were too slow for the amount of calculation waiting to be performed. While working at Marquardt, I was chastised one day by my boss, for not shaving. It was caused by being up since the previous morning running a trajectory simulation on the CPC. Under such circumstances, everyone wanted programs to run as fast as they could. That was why the program optimizers for drum machines (like SOAP) were so heavily used.

When the 701 superseded the CPC, the balance between user and machine changed. One man at the RAND Corporation took two years to program a problem that ran in two minutes. He experienced considerable culture shock.

There was competition everywhere to have the fastest program for a given task, quite often a mathematical subroutine. When published, those subroutines always had timing associated so the user could plan wisely. The situation was much the same as in the early days of microcomputers. Jewel work was needed, and the domain was small enough to see and measure something. There was even competition between software and

hardware people. The 705 engineers were shocked when a programmed divide ran faster than the hardware instruction—without firmware, they could not program a Newtonian iteration.

I suspect that FORTRAN itself had much to do with the temporary hibernation of performance evaluation. After programming in the other languages, it gave so much power because of the ease of use (and the efficiencies were incorporated for you in the compiler), that the number of user of computers could expand much more rapidly. It wasn't until operating systems came into heavy use that we rediscovered the need to prevent waste.

COMMUNICATIONS AND TIMESHARING

It wasn't Ethernet, but George Stibitz had tied into a relay computer by way of a Teletype—in 1940. SAGE was one of the first major projects to use direct inputs from communications lines. FORTRAN wasn't available when it began, and couldn't have been used for much of the job if it had, for it wasn't just a scientific problem.

Timesharing was just talk. The first time I find the word appearing is in a J. W. Forgie paper on the input-output system for the Lincoln TX-2 computer, concurrent with the 1957 FORTRAN paper. I proposed such usage in an article the next month; it was suggested that IBM should fire me, because that wasn't in line with their policy.

COMPILERS

Compilers existed before FORTRAN, but they were all rudimentary in comparison. Grace Hopper, chief pioneer of the concept, might have gone faster further if she had had the type of support given to Backus and his group. IT, A2 and A3 were true compilers, but they avoided interactions and optimization.

DATA INDEPENDENCE

This concept arose with the commercial compiler languages. Grace Hopper and company wrought the Data Division concept. Scientific languages all stuck to floating point, with integers for loop control.

Data structure was usually built into the program, and it didn't seem important, because hardly any interchange of programs took place between different computers. Even if that were possible one could not necessarily get the same answers due to different hardware characteristics.

SOFTWARE PIECE PARTS

Piece parts for software first came to attention at the first Software Engineering conference in 1968, proposed by Doug McIlroy. However, Bob Glass makes a convincing case⁴⁵ that they were in existence before FORTRAN, certainly via the SHARE organization. Indeed they were necessary to counteract the inefficiencies of working without such compilers.

SOFTWARE PACKAGES

In the modern sense the software package did not exist, for today they cost money. Before FORTRAN it was unthinkable to sell software, although the packages did exist. They were traded or given away. Examples are several general CPC boards, plus the many 650 packages published in the IBM Technical Newsletter No. 10.²⁷

There is no doubt that packages existed. They were source programs for interpretation, not compiled source as today. A buzzword of the times was "abstraction." Douglas Aircraft had a "matrix abstraction," for example.²³ It manipulated matrices and performed combinatory functions. Ergo, if your problem could be expressed in matrix form, it could be solved. So it was urged that all problems be expressed this way, a not altogether natural way of use. But many of today's software packages have similar contortional requirements upon the user.

Codes for nuclear computation also fell in the category of software packages, even if they were exchanged in machine language form. Hundreds of these codes were disseminated.

SUMMARY

I'm enjoying the developments of today, but my pleasure is a bit spoiled by the terrible waste in software development, and so much poor software. It's tempting to recall Miniver Cheevy, who loved "the medieval grace of iron clothing." Software before FORTRAN could be considered quite medieval, even primitive, but there were certain graces.

From my starting in the computer field in early 1949, until FORTRAN arrived, I was either working too hard to see the Peter Principle in effect, or else it didn't exist in such a virulent form. It was exciting to build software then. We had management support and trust for whatever we thought was possible. The number of levels of management was low, and the control tenuous. I reported to John Backus in FORTRAN days, but never felt the slightest pressure. I looked upon him as a friend, not a menace. So today we have better tools and knowledge, and theories of program correctness and such. I don't think that they have added to the fun and excitement of Computing Prior To FORTRAN!

REFERENCES

1. Youden, W. W. "Computer Literature Bibliography 1946 to 1963." US Natl. Bur. Standards Misc. Publ. 266, 1965 Mar 31.
2. Wilkinson, J. H. "Coding on automatic digital computing machines." Report Conf. on High Speed Automatic Calculating-Machines, Univ. Math. Lab., Cambridge, England, 1949 Jun 22-25, 28-35.
3. Huskey, H. D. "Semiautomatic instruction on the Zephyr." Proc. 2nd Symp. on Large-scale Digital Calculating Machinery, Cambridge, MA, 1949 Sep 13-16, 83-90. Harvard U. Press, 1951, Annals Vol. 26.
4. Stowe, L. "Programming." Office of Naval Research Seminar on Data Handling and Automatic Computing, Washington, DC, 1951 Feb 26-Mar 6, 79-84.
5. Wilkes, M. V., D. J. Wheeler, S. Gill. "The Preparation of Programs for a Digital Computer." Addison-Wesley Press, Cambridge, MA, 1951.
6. Proc. ACM Conf., 1952 May, Pittsburgh, PA.
- 6.1 Adams, C. W. "Small problems on large computers." 99-102.
- 6.2 Lipkis, R. "The use of subroutines on SWAC." 231-234.
- 6.3 Wheeler, D. J. "The use of subroutines in programmes." 235-236.
- 6.4 Carr, J. W. III. "Progress of the Whirlwind computer towards an automatic programming procedure." 237-242.
- 6.5 Hopper, G. M. "The education of a computer." 243-250.
7. Proc. ACM Conf., 1952, Toronto.
- 7.1 Ridgway, R. K. "Compiling routines." 1-5.
- 7.2 Isaac, E. J. "Machine aids to coding." 17-28.
- 7.3 Strachey, C. S. "Logical or non-mathematical programmes." 46-49.
- 7.4 Bennett, J. M., D. G. Prinz, M. L. Woods. "Interpretative subroutines." 81-87.
8. Rutishauser, H. "Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen." Mitteilung aus dem Institut für angewandte Mathematik, Basel, 1952, 1-45.
9. Rochester, N. "Symbolic programming." PGEC (IRE Transactions on Electronic Computers), Vol. EC-2, No. 1, New York, 1953 Mar, 10-15. LC Card 57-39723.
10. Hopper, G. M. "Compiling routines." Computers and Automation 2, No. 4, 1953 May, 1-5.
11. Hopper, G. M., J. W. Mauchly. "Influence of programming techniques on the design of computers." Proc. IRE 41, No. 10, 1953 Oct, 1250-54.
12. Bouricius, W. G. "Operating experience with the Los Alamos 701." Proc. Eastern Joint Comput. Conf., 1953 Dec 8-10, 45-47.
13. Bennett, J. M., A. E. Glennie. "Programming for high-speed digital calculating machines." In "Faster than Thought." B. V. Bowden, London, Pitman, 1953, 101-116. LC Card 54-15305.
14. Wilkes, M. V. "The use of a 'Floating Address' system for orders in an automatic digital computer." Proc. Camb. Phil. Soc., 49, Part I, 1953, 84.
15. Laning, J. H., N. Zierler. "A program for translation of mathematical equations for Whirlwind I." Engg. Memo. E-364, M.I.T. Instr. Lab., 1954 Jan.
16. Backus, J. W. "The IBM 701 speed-coding system." J. ACM 1, No. 1, 1954 Jan, 4-6.
17. Symposium on Automatic Programming for Digital Computers, Office of Naval Research, Washington, DC, 1954 May 13-14. LC Card 56-60789 rev.
- 17.1 G. Hopper. "G. M. Automatic programming definitions." 1-5.
- 17.2 Moser, N. B. "Compiler method of automatic programming." 15-21.
- 17.3 Waite, J. "Editing generators." 22-29.
- 17.4 Goldfinger, R. "New York University compiler system." 30-33.
- 17.5 Holberton, F. E. "Application of automatic coding to logical processes." 34-39.
- 17.6 Adams, C. W., J. H. Laning, Jr. "The M.I.T. systems of automatic coding: Comprehensive, Summer Session and Algebraic." 40-68.
- 17.7 Muller, D. E. "Interpretive routines in the Illiac library." 69-73.
- 17.8 Gorn, S. "Planning universal semiautomatic coding." 74-83.
- 17.9 Brown, J. H., J. W. Carr III. "Automatic programming and its development on the MIDAC." 84-98.
- 17.10 Livingston, H. M. "Automatic programming on the Burroughs Laboratory computer." 99-105.
- 17.11 Backus, J. W., H. Herrick. "IBM 701 speedcoding and other automatic programming systems." 106-113.
- 17.12 Elmore, M. "The LMO edit compiler." 114-116.
- 17.13 Keller, A., R. A. Butterworth. "Programming for the IBM 701 electronic data processing machine with repetitively used functions." 117-149.
18. Jones, J. L. "A survey of automatic coding techniques for digital computers." M.S. Thesis, M.I.T., 1954 May.
19. Proc. Eastern Joint Comput. Conf., 1954 Dec 8-10.
- 19.1 Rice, R. Jr. "Why not try a plugboard?" 4-10.
- 19.2 Krider, L. D. "Applications of automatic coding to small calculators." 64-67.
20. Rutishauser, H. "Some programming techniques for the ERMETH." J. ACM 2, No. 1, 1955 Jan, 1-4.
21. Herbst, E., N. Metropolis, M. B. Wells. "Analysis of problem codes on the MANIAC." M.T.A.C. 9, No. 49, 1955 Jan, 14-20.
22. Hopper, G. M. "Automatic programming of digital computers." Proc. High Speed Comput. Conf., Baton Rouge, LA, 1955 Feb 16, 113-118. LC Card 57-63206.
23. Denke, P. H., I. V. Boldt. "A general digital computer program for static stress analysis." Proc. Western Joint Comput. Conf., 1955 Mar 1-3, 72-78.
24. Bradshaw, T. F. "Automatic data processing methods." Proc. Auto.

- Data Proc. Conf., Cambridge, MA, 1955 Sep 8-9, 3-27, Harvard U. Press, 1956.
25. "Automatic programming the A-2 compiler system." Parts 1 and 2, *Computers and Automation*, 4, Nos. 9 and 10, 1955 Sep and Oct.
 26. *Electronic Digital Computers and Information Processing*, Darmstadt, Germany, 1955 Oct 25-27, F Vieweg, Braunschweig, 1956. LC Card 59-18764.
 - 26.1 Rutishauser, H. "Methods to simplify programming, 5 years work with the Z4 computer" (German), 26-30.
 - 26.2 Samelson, K. "Problems of programming techniques" (German), 141-142.
 - 26.3 Lehmann, M. J. "Automatic computer programming" (German), 143.
 - 26.4 Loopstra, B. J. "Processing of formulas by machines." 146-147.
 - 26.5 Thuring, B. "The automatic programming of Univac by the A-2 compiler system." (German), 154-156.
 27. Technical Newsletter No. 10, IBM Applied Science Division, New York 1955 Oct.
 - 27.1 Ruthrauff, R. E. "Symbolic coding and assembly for the IBM Type 650." 5-14.
 - 27.2 Horner, J. T. "Relative programming for the IBM Type 650." 15-27.
 - 27.3 Bosak, R. "Development of a floating decimal abstract coding system (FACS)." 28-30.
 - 27.4 Bemer, R. W. et al. (Lockheed MSD). "A general utility system for the IBM Type 650." 31-48.
 - 27.5 Mandelin, A. R., K. D. Weaver. "A selective automonitoring tracing routine called SAM." 49-62.
 - 27.6 Battin, R. H., R. J. O'Keefe, M. E. Petrick. "The MIT Instrumentation Laboratory automatic coding 650 program." 63-79.
 - 27.7 Titus, C. K. "An integrated computation system for the IBM 650." 80-89.
 28. Hume, J. N. P., B. H. Worsley. "TRANSCODE, a system of automatic coding for FERUT." *J. ACM* 2, No. 4, 1955 Oct, 243-252.
 29. T. Gorman, T. P., G. Kelly, R. B. Reddy. "Automatic coding for the IBM 701." *J. ACM* 2, No. 4, 1955 Oct, 253-261.
 30. C. Adams, C. W. "Developments in programming research." *Proc. Eastern Joint Computer Conf.*, 1955 Nov 7-9, 75-79.
 31. Gordon, B. "An optimizing program for the IBM 650." *J. ACM* 3, No. 1, 1956 Jan, 3-5.
 32. *Proc. High Speed Computer Conference*, Baton Rouge, LA, 1956 Feb. LC Card 57-63206.
 - 32.1 Hopper, G. M. "Automatic coding techniques 1955." 6-12.
 - 32.2 Heller, J. "Mathematical service routines." 151-153.
 - 32.3 Perry, D. P. "Specifications for an automatic matrix program." 210-215.
 33. *Proc. Western Joint Computer Conf.* 1956 Feb 7-9.
 - 33.1 Ross, D. T. "Gestalt programming, a new concept in automatic programming." 5-9.
 - 33.2 Grems, M., R. E. Porter. "A truly automatic programming system." 10-20.
 - 33.3 Moncrieff, B. "An automatic supervisor for the IBM 702." 21-25.
 - 33.4 Bemer, R. W. "PRINT I, a proposed coding system for the IBM Type 705." 45-48.
 - 33.5 Goldfinger, R. "The IBM Type 705 Autocoder." 49-51.
 - 33.6 Meek, H. V. "An experimental monitoring routine for the IBM 705." 68-69.
 34. Brooker, R. A. "The programming strategy used with the Manchester University Mark I computer." *IEEE Conf. on Digital Computer Techniques*, Suppl. Part B, Vol. 103, London, 1956 Apr 9-13.
 35. Perkins, R. "EASIAC, a pseudo-computer." *J. ACM* 3, No. 2, 1956 Apr, 65-72.
 36. Bauer, W. F. "An integrated computation system for the ERA-1103." *J. ACM* 3, No. 3, 1956 Jul, 181-185.
 37. Blum, E. K. "Automatic digital encoding system II (ADES II)." *Proc. ACM Conf.* 1956, 29.
 38. Chippis, J., M. Koschmann, S. Orgel, A. J. Perlis, J. Smith. "A mathematical language compiler." *Proc. ACM Conf.* 1956, 31.
 39. *Symp. Advanced Programming Methods for Digital Computers*, 1956 June 28-29, ONR Report ACR-15, 1956 Oct.
 - 39.1 Hopper, G. M. "The interlude 1954 to 1956." 1-2.
 - 39.2 Wegstein, J. H. "Automatic coding principles." 3-6.
 - 39.3 Thompson, C. E. "Development of common language automatic programming systems." 7-14.
 - 39.4 Benington, H. D. "Production of large computer programs." 15-28.
 - 39.5 Jones, F. "SHARE, a study in the reduction of redundant programming efforts through the promotion of inter-installation communication." 29-34.
 - 39.6 Carr, J. W. III, B. Arden. "Advanced programming techniques with smaller computers." 35-38.
 - 39.7 Goldstein, M. "Computing at Los Alamos, Group T-1." 39-44.
 - 39.8 Wells, M. "Coding for the MANIAC." 45-48.
 - 39.9 Holberton, F. E. "Proposed advanced coding system for the UNIVAC-LARC." 49-56.
 - 39.10 Waite, J. H. Jr. "RCA approach to automatic coding for commercial problems." 57-66.
 - 39.11 Selfridge, R. G. "The PACT compiler for the 701." 67-70.
 - 39.12 Blum, E. K. "Automatic digital encoding system II." 71-76.
 40. *J. ACM* 4, No. 4, 1956 Oct.
 - 40.1 Melahn, W. S. "A description of a cooperative venture in the production of an automatic coding system." 266-271.
 - 40.2 Baker, C. L. "The PACT I coding system for the IBM Type 701." 272-278.
 - 40.3 Mock, O. R. "The logical organization of the PACT I compiler." 279-287.
 - 40.4 Miller, R. C. Jr., B. G. Oldfield. "Producing computer instructions for the PACT I compiler." 288-291.
 - 40.5 Hempstead, G., J. I. Schwartz. "PACT loop expansion." 292-298.
 - 40.6 Derr, J. I., R. C. Luke. "Semiautomatic allocation of data storage for PACT I." 299-308.
 - 40.7 Greenwald, I. D., H. G. Martin. "Conclusions after using the PACT I advanced coding technique." 309-313.
 41. *Symp. on Automatic Coding*, Franklin Institute, Philadelphia, PA, 1957 Jan 24-25, Monograph No. 3.
 - 41.1 Petersen, R. M. "Automatic coding at G.E." 3-16.
 - 41.2 Katz, C. "Systems of debugging automatic coding." 17-28.
 - 41.3 Bemer, R. W. "PRINT I, an automatic coding system for the IBM 705." 29-38.
 - 41.4 Kinzler, H. M., P. M. Moskowitz. "The procedure translator, a system of automatic coding." 39-56.
 - 41.5 McGee, R. C. "Omnicode, a common language programming system." 57-70.
 - 41.6 McGinn, L. C. "A matrix compiler for Univac." 71-86.
 - 41.7 Perlis, A. J., J. W. Smith. "A mathematical language compiler." 87-102.
 - 41.8 Yowell, E. C. "A mechanized approach to automatic coding." 103 ff.
 42. *Proc. Western Joint Computer Conf.* 1957 Feb 26-28.
 - 42.1 Grems, M. D., R. K. Smith, W. Stadler. "Diagnostic techniques improve reliability." 172-178.
 - 42.2 Baskus, J. W., R. J. Beeber, S. Best, R. Goldberg, L. M. Haitb, H. L. Herrick, R. A. Nelson, D. Sayre, P. B. Sheridan, H. Stern, I. Ziller, R. A. Hughes, R. Nutt. "The Fortran automatic coding system." 188-197.
 - 42.3 Newell, A., J. C. Shaw. "Programming the Logic theory machine." 230-240.
 43. *Table, Comm. ACM* 1, No. 2, 4.
 44. *Tables, Comm. ACM* 1, No. 11, 5-6.
 45. Glass, R. L. "Software parts nostalgia." in *Reader's Forum, Datamation* 27, No. 12, 1981 Nov, 245-247.

HL5UA XXXVIII -- CHICAGO - 1984 April 08-11

TM
THE SCREEN ENVIRONMENT
AS AN APPLICATION AND SOFTWARE FACTORY SHELL

R. W. Bemer
Bob Bemer Software, Inc.
Phoenix, AZ, U.S.A.

Abstract: Screen editing is just part of the capability of the Screen Environment. Other aspects allow its use as the front-end for operating systems and application packages, for which there are no current standards. This shell principle, now very well established, is critical to making an optimum human interface to all of the software a computer user utilizes.

This presentation will give information about the features that provide the shell capability, plus a proposal for a standard, uniform time-sharing environment for all L66 and DPS8 users.

The SCREEN ENVIRONMENT™

An Application and Software Factory Shell

Examine any software package. How do you run it? Well, you must provide input data, parameters, flow rules, etc. Because of operating system involvement, much of this information must be provided to the JCL stream, the lowest level of interface. If JCL work did not get complicated, an application user could be taught to construct the needed JCL, or to modify a canned JCL in which the variable parts and options were indicated.

But it is complicated, and this method has been superseded in the interest of ease of use and human interface. The application user is now asked WHAT is wanted done, not HOW to do it. Another program is interposed between the user and the JCL. It must know how to analyze the WHAT responses and turn them into the HOW of a JCL -- the sequencing and looping logic, the input and output formats, etc.

Input is furnished to the intermediate program (the JCL constructor) by:

- o Responses to displayed (text) questions,
- o Selection among the displayed (text) options by specifying their identifiers, or
- o Filling in areas in a form.

In computer usage these have been mutually exclusive, historically. But this is not true in paper forms, which may include all three types, plus marking boxes for the options (which may then be selected in multiple), a more positive visual reinforcement of a correct selection. You may fill in paper forms in any order you choose, change replies later if you wish (erase and rewrite), and make copies when partially complete

(with common information), to be completed with variable information. Sound like different runs of the same application program? And guiding information may be placed on the form.

Computer forms have not yet achieved the flexibility and utility that they can. This is because:

- o Answers and choices are obtained at fill-in time. The controlling program depends upon a built-in time sequencing, for questions, or for moving to the field to be filled in next. The user is not usually granted any choices of sequence or replacement.
- o Relatively small viewing screens demand both vertical and horizontal scrolling to show all of a form of reasonable size, with integral "help" verbiage. The Honeywell expense account, for example, is some 250 character positions wide.
- o It has been thought necessary to load the screen with attributes, including constraints upon the content entered (because of the difficulty in directing change, or handling reentry at a later time, as mentioned previously). These attributes are physical to the screen, and lines may not be inserted or deleted to provide free-form entry, in the way that paper forms may have extra ruled spaces if needed. Attributes may also interfere with vertical scrolling, even for expensive 72-line buffer terminals, and I have not yet seen screens that scroll attributes horizontally!

The Advantages of Paper

It is instructive to consider how answers are extracted from paper forms. A human scans for the appearance of the desired question on the sheet. The answer is read at a known relative physical position to the question. Answers may be abstracted for any question by a repeat of the process, from top of page, or they may be obtained sequentially. In a computer these actions can be reproduced with a perfile (the paper form) and a TEX program (the scanner for questions and answers).

The relative decision times for a JCL may be quite different from those of a person specifying an application run. Utilizing a perfile in this way can free the user from the timing constraints of the operating system, which he presumably cares nothing about. More freedom is thus allowed in forms design. In short, we ought to be constructing our computer front-ends to be as much like paper input as possible!

Reusable Input

All of these front-end procedures need text input by the user. It is desirable for a user to be able to amend existing instructions to the application program without going through the entire front-end process again.

- o The input might be miskeyed and need correction.
- o It may be keyed correctly, and yet found wrong in substance.
- o It may be subject to change for multiple runs.
- o It may be subject to change as the procedure gets nearer to completion.
- o If the input is similar to some previous input that was saved, that could be remodified instead of starting from scratch.

Shells

There is good reason why you see trade papers and magazines full of news and advertisements of UNIXTM and UNIX-like products. It gives not only portability but, equally important, the ability for the operating system to be manipulated by programs. This permits a "shell" to be created that can act as an intelligent surrogate for users. The language TEX has this same capability, which has been used in the Screen EnvironmentTM.

You will find that windowing capability is another hot item in the news. Mostly they mean more than one window on the same screen. That has some uses, but it in the opposite direction to providing a view of larger documents for program control.

The most modern aims and intentions in this direction are to be found in the (in)complete environments for the ADA language. Note what the ADA community means in using the word "environment". It is fully consistent with the avowed purpose of the Screen EnvironmentTM. It is not just a screen editor, as HLSUA was told in a previous presentation.

The Screen EnvironmentTM was designed from the beginning to be extensible. It is possibly better in that regard than IBM's SPF (Systems Programming Facility). It is TEX, of course, that makes it possible. One evaluator said:

"SE allows an escape to TEX (from which other tasks could be done) and then to return to the same place. This might be compared to SPF's split window feature, though SE's escape is easier to use and more flexible. ... A feature I particularly like is the ability to invoke user procedures (TEX or CRUNs). It makes SE extensible, allowing users to write their own commands and store them in a library."

Screen EnvironmentTM as a Shell

The Screen EnvironmentTM has many properties that make it very useful as a shell to use for front-end actions:

- 1) It is an completely general file editor, to create or modify any file(s), and particularly perfiles that represent the running instructions for software application packages.
- 2) In emulation of the capabilities of paper, it can display a file some 1200+ characters wide and as many lines long as GCOS will permit.
- 3) It can run background processes via the call-by-name method. The user may create a file defining the content of several variables that have names starting with a "u", wherein user processes are invoked. Eg., a TEX program call that compiles and runs the current file (just edited), putting the execution report in the fourth window and ringing a bell to advise you of completion -- Just type F4ucobol
- 4) After these background processes are initiated, via a DRUM/MAIL scheme, other work may be begun or continued.
- 5) Upon completion of a background process, the results may be displayed within the Screen EnvironmentTM, ready for editing and further processing (e.g., being put into the front-end procedure for other software packages).
- 6) Inherent in the operating method of the Screen EnvironmentTM is a constant awareness of the cursor position, known and used at all times. One can thus modify, by editing, text representing properties (as of a file) to cause, using some subservient process, changes to the properties themselves!

Imagine an "ACCE" command having its output displayed, as any file, by SE (the Screen EnvironmentTM, for short, from now on). One could then edit this display to indicate properties to be altered. The subservient process, knowing the cursor positions of both the attributes and the text change, makes this connection to create an actual ACCE command to effect the change!

7) Escape back to system level is possible, while still retaining the ability to edit system commands. One stays at system level unless a null response is given to the "=" prompt, at which time the previous screen returns.

The Proposal

We note that there are no Honeywell or ANSI standards for the construction of front-end processes, and therefore the necessary actions and human interfaces now vary substantially in form, method, ease, and flexibility of operation. We submit that this chaos puts a needless and harmful burden upon the users, and that a method exists to eliminate it.

Proposition:

The HLSUA should recommend that wherever possible, in new design or modification, all application packages should be operated via a perfile controller. This does not imply any restrictions on design and layout of such perfiles. It will ensure interchangeability and consistency among all packages because no necessary information is buried in the front-end program.

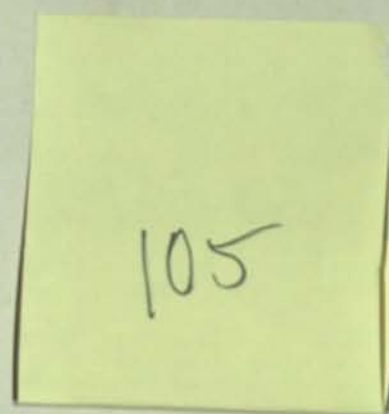
This is a principle equivalent to the separation of Data and Program Division in programming languages, and will be found to be of equal importance.

How Do We Get There?

Here is a scenario possible even if the HLSUA does not adopt such a proposal:

- 1) More and more users obtain the Screen Environment™, or other products, because of the obvious capability for increasing productivity. They learn of these from existing users, from sales efforts, and from trade press reports on this and similar products.
- 2) As experience is gained in using SE, more and more people at the sites find that they can write procedures that are executed automatically when SE is called, and/or when SE is relinquished permanently to system level.
- 3) Package vendors, including HIS, will hear users talk about the greater flexibility of SE in the front-ending of processes. Many users of multiple packages will discover those inevitable conflicts and variations in style and method among them.
- 4) Some of these will then discover how SE can front-end the package front-ends themselves. They will then see inefficiencies in such 2-step processes, and thus ask the package vendors if they cannot supply an alternate form for going directly from SE into their package(s).
- 5) Many vendors, particularly HIS, which already makes heavy use of SE in creating their packages, will find it useful to provide packages in two forms -- one for installations having SE, the old one for those that do not.
- 6) Vendors will notice that a substantial portion of their fabrication and maintenance costs did lie in the front-end processes, the removal of which could make their packages more competitive in price if need be.
- 7) From somewhere, perhaps a birds-of-a-feather group, will arise desires and pressures for some kinds of standards in the front-ending of applications. Because using SE is the simplest way to do this, it will become widely known, soon being as indispensable as COBOL, for example.
- 8) Then it will be discovered that SE is capable of supporting most of the features necessary to the proposed COSCL operating system, toward which GCOS must adapt. It will be discovered further that operating systems are also susceptible to front-ending, not only application packages. AND AWAY WE GO! It is feasible to build a forms-driven GCOS, run via screens displayed under the Screen Environment™. When will we see it? Who will grasp the market opportunity?

READERS' FORUM



WORKING ON MOON MOUNTAIN

It was shortly after I took early retirement from Honeywell, so it must have been September 1982 when Paolo Soleri, the noted architect, brought a group of his disciples and workers to Phoenix for discussions with several HIS people. The topic was the proper integration of computers into the house and office architecture of the future. For Soleri is a futurist in action, in the process of building a city called Arcosanti, in the desert north of Phoenix. I was invited to the discussion because I had been involved in using a computer from home since 1972.

I had considered the matter before that, however. At the first Software Engineering Conference in Garmisch, Germany, in October 1968, I had met Dr. Edward David, then in charge of the Picturephone project at Bell Laboratories. I was planning to build a house atop an old volcano in Phoenix, and wanted to plan it as much as possible for future computer work. Even then we sensed the tremendous potential of integrating computers and communications. The house would be sited clear of interference for microwave, if that option should open. A wall would be reserved for a holographic screen, if that science fiction should come true.

Ed said he would try to get me the first Picturephone in Arizona as a test bed, and recommended bringing in 25 telephone lines to the house as a precaution for high bandwidth requirements. I had to go quite high in the hierarchy of Mountain Bell to get that done. Everyone thought I was out of my mind to want 25 telephones! Finally they agreed, and put in quite a fancy switchbox, at no particular cost to me at that time, with AT&T still integral.

The Picturephone project did not fare well in the original two cities chosen for test marketing. In addition, Dr. David left the project to become President Nixon's Science Advisor. I did have a need for the lines, however. In 1972, I became editor of the *Honeywell Computer Journal*, a now-defunct magazine that we nevertheless published well enough at that time to win over *Scientific American* in a contest sponsored by the Printing Industries of America.

Writing and creative work is not called up by opening the tap between 8 a.m. and 5 p.m. Inspirations really do come at odd hours. I convinced Honeywell to tie one of my lines to its telephone switching network, and brought home one of their Terminate 300s—a hardcopy terminal, since there were few video

screen terminals in existence at that time, and they were not fitted out for word processing. That term, you may recall, was unknown then. I was setting out to do it, however, because we were going to put on magnetic tape the files we created on a Honeywell computer, to take to Datagraphics, a Phoenix firm, for photocomposition directly from our own copy. But that is a different story, and has nothing to do with working at home.

Although the journal ceased publication in 1974, whereupon I returned to more of a programmer status, I discovered (rediscovered?) that programming, too, is creative work. From then until this day, I apparently program during my sleep, and cannot wait to get it into a permfile on the computer. What with showering, dressing, breakfasting, and driving to work—I could possibly lose the most important parts of my new code.

So I have worked at home for over a decade. This qualifies me, I believe, to assert some of the advantages and disadvantages of this mode of operation, perhaps in a manner that will amuse as well as instruct.

First, are you married? If so, your spouse is obviously the key element in the way you work. Mine, a woman, knew little of the technical details of the computer profession despite having been the receptionist for IBM World Headquarters. Her most pressing concern was why didn't I go to the office more? Wouldn't Honeywell fire me for not being there so someone could check on whether I was actually producing anything useful? Eventually she began to see articles in periodicals and newspapers about this mode of working and became more comfortable with it. I got no pioneering credits, however.

Here are some of the advantages of this mode of work:

- You're on hand for emergencies like a broken water line. A call to the office and a (perhaps long) trip home are saved.
- You have more flexibility in planning the work of the day, to interleave it with other activities such as shopping. Or if a movie is less crowded in the afternoon, why not see it then and do your computer work in the evening?
- If your spouse also works with computers, then you need not worry about the distance between your two offices.

There are, however, some disadvantages:

- You're on hand for more than emergencies, like walking the dog if it whines. There may come a time when you will have to bang the door shut and yell, "@!\$%—I am at the office!"
- Speaking of those dogs, when you get up at 3 a.m. so as to not lose your latest gem, they are likely to wake up with you, concurrently (a nice computer word) waking the spouse, who may not wish to be awakened.
- If you and your spouse disagree on other matters, and if he is male, then he may have a job away from home, which avoids friction. If a female, then long absences from home may be indicated, and your shopping will include a great many tv dinners.

Now we discuss the relative physical comforts of the home office vs. the office office. You may think I am going to

Get Ahead of the Gang



Face it. We could

all use a little concrete advice now and then.

Whether you're making a monumental decision like a career change, or a more routine one like an oil change, the free Consumer Information Catalog can make it easier.

The Catalog is published quarterly by the Consumer Information Center of the U.S. General Services Administration to bring you the latest on government programs and a mountain of other information that you can use.

The Catalog lists more than 200 government booklets to help you . . . start a business, find a job, plan your retirement, repair your home, or carve out a nutrition and exercise program to improve your profile. And many of these booklets are free.

So order your Catalog today. Any way you look at it, you'll be head and shoulders above the crowd.

Just send your name and address on a postcard to:

**Consumer Information Center
Dept. MR
Pueblo, Colorado 81009**

A public service of this publication and
the Consumer Information Center of the
U.S. General Services Administration



READERS' FORUM

discuss the square footage and the desk available. I am not.

The advantages:

- The attire required may be informal. A bathrobe can do until the computer goes down about 10 a.m. Then a shower while they reboot; you haven't really lost any time.
- The attire may be *very* informal in climes comparable to that of Phoenix. I like to put my terminal out by the pool in the summertime. That way I can get a good suntan (all over, which is an advantage of a mountaintop where nobody can see you) while simultaneously doing useful work. When it gets unbearably hot, a quick back flip into the pool does nicely. (Always remember to dry your hands before returning to the keyboard.) When I tell people of this they often ask, "Why don't you get a light pen that writes under water, so you can just stay in the pool?"
- The size of your office, and whether it has a door that closes, are of no consequence. Privacy and pecking order morale are assured.

The disadvantages:

- It is difficult to read a video terminal in bright sunlight. I do not think the designers have considered this problem yet. One must build a sort of enclosure and turn the brightness up as far as it will go. And there will always be somebody who writes programs using the faint intensity for emphasis.
- Services are distant. You may use up your last red marking pen. There is no secretary to send to a supply cabinet. If your terminal has problems you will have to haul it to work and back for repair. Unless, of course, you know something about electronics yourself, and the company has provided you with spare parts. For hardcopy printers you must keep a good supply of paper handy, always remembering to stock up at the office when supplies are low.
- Services hardly exist on Saturday and Sunday, and you will surely be working then, so make up a Friday checklist in preparation for the exodus.
- If you are the type that goes to many meetings, in contrast to getting out the work, working at home will be a problem unless your office is *very* close to your home. Fortunately the electronic meeting software is advancing nicely, and this problem will disappear.

The last comparison is on the matter of intellectual stimulation and ongoing education. In the meeting with Paolo Soleri, I was introduced as an ardent advocate of working *alone* (they didn't say *at home*, which is different). I protested vigorously, saying that there were two types of information transfer—directed and broadcast—and they must not be confused.

The advantages:

- For directed information transfer, working at home is ideal. You are free from disruptions, such as the casual walk-in to your cubicle. Such contacts may be deferred until a better time to read and answer your electronic mail, which is much more efficient and less time-consuming. Of course, if your motivation is to have fun at the office, rather than producing, then you should read this memoir no further!
- They really have to want you at a meeting before an invitation is sent, and they know it must be planned, not just spur-of-the-moment. Having the participants come in specially for personal meetings superimposes a needed structure on work.
- At the office, some offices are closer than others. At home, by electronic mail, all are equidistant. Zero, that is. I recall a particular example of what Sperry Rand used to advertise as "synergy," where the output was greater than the total of the inputs. I was working, from my home, on System X, about two miles away. Gerry Despain was at his office on Camelback Road, some 10 miles away, working on the same mainframe. He modified a program of mine and asked me, via electronic mail, to try the improved version. I did, saw how it could be even better, remodified it, sent him mail, and so forth. Back and forth, until we shortly obtained a joint product better than either of us

ADVERTISERS' INDEX

AGS Management Systems	158
AT&T Information Systems	90-91
Able Computer	25
Advanced Programming Resources	88
Alpha Micro Systems	144-8/9
Amcor Computer	46
Ansal	46
Ashton-Tate	101
Bank of America	167
*Bell Telephone Mfg.	144-18
Boeing Computer Service	CV2
Boise Cascade	113
*CTI Data	144-30
CXI, Inc.	99
Calcomp	1
Comdex Europe	175
Compugraphic Corp.	37
Computer Corp. of America	48-49
Computer Power Products	128
Computer Power Systems	51
Computer Security Institute	112A-X
Control Data Corp.	142-143
Control Data Corp.	CV3
Convergent Technologies, Inc.	6-7
Cullinet	20-21
D & B Computing	47
Data General Corp.	121
Datagraphix	153
Dataproducts Corp.	19
Decision Data Computer Corp.	12
Digital Communications Assoc.	16-17
Digital Equipment Corp.	146-147
DuPont Co.-Tyvek	50
Dunspus	45, 61
*Faraday Electronics	144-15
Genicom	109
Gold Hill Computers	81
Gould Software Div.	162-163
Henco Software	92
*Hermes Precisa Int'l.	144-29
Hewlett-Packard	62-63
Hewlett-Packard	89
Honeywell	38-39
*IBC-Integrated Bus. Computers	144-24
IBM	28
IBM	73-75
IBM	122-123
*IBM	144-31
*ICC-Europe	144-28
*ICC-U.S.	144-21
ITT Courier	52
Infotron Systems	2
Innovative Software	10-11
Integrated Technologies, Inc.	151
Intel	82-83
Intel	161
Lear Siegler, Inc.	107
*Liberty Electronics	144-7
Arthur D. Little, Inc.	160
Lotus Development Corp.	154-155
M/A-COM Linkabit	69
MAI/Basic Four	58-59
MSA	26-27
*MTI	144-5
*MTR	144-13
McCormack & Dodge	35
Micom Systems, Inc.	CV4
Mitchell Management Systems	152
Motorola	136-137
NCR Corp.	172
NEC	70-71
NEC	95
NEC	145
Network Systems Corp.	141
Nixdorf	5
*Nova Graphics	144-10
On-Line Software	127, 129
Pacific Bell	169
Paradyne	87
*Plexus Computers	144-2
*Portex '85	144-19
Precision Visuals	111
Prime Computer	114-115
*RCA Service Co.	144-23
R.T. Katak	119
Rair Microcomputer Corp.	43
Rollm Corp.	130-131
SPSS, Inc.	15
Scan Optics	56
Software AG	9
Software Corp. of America	57
Sperry Corp.	139
Systems Center	54-55
Tab Products	159
Technology Insight	152
Tektronix, Inc.	97
Televideo Systems, Inc.	102-103
Telex Computer Products, Inc.	148
3M Co-Corporate	79
3M Co-Interactive Systems	125
*Unity Corp.	144-26
VM Software	157
Van-Tel Security	65
Visual Technology Inc.	66-67
Wang Institute	4
Wyse Technology	30-31
*Xebec Corp.	144-16/17
Zilog Systems	135
Ziyad, Inc.	22
*OEM Edition	

READERS' FORUM

would have done individually. Yes, people can approach this at the office, but the odds are smaller that they will do so. If you don't believe me, try to get two people to make the experiment. It is an exciting and stimulating method of working.

The disadvantages:

- One may read journals to keep up with developments in one's field, but the time lag to publication remains long. It will probably stay long, even though the obsolescence cycle in the computer field grows ever shorter. To really keep up you must not only read the trade papers, but mingle—in the cafeteria and in conferences—anywhere there is broadcast information. Don't depend upon electronic publishing as it is now. It is a joke. My eye takes in a printed sheet as a single frame. I may or may not opt to read anything there. The mechanics of how I scan this way need not be explained. The fact is that I do it, and there is no way today that I can scan an electronic page similarly, or as cheaply as a printed one.

- Electronic mail is also directed. There is no easier way to pick up information than gossip, or talking to friends. Face-to-face conversation can wander in free-form, as electronic mail can never do (ever try interrupting electronic mail?). One thing sparks another; then someone mentions having heard from someone else that Joe Xyz has something that may interest you.

Working at home is advantageous in more ways than just cutting transportation costs, but I think it has been successful only because I have not acted the hermit as a result of it. Having my own business would be less effective without my ties to Honeywell—using new software as it is tested, reading and following those good mail suggestions, and talking to people when I go to pick listings from the page printer. Those are my links to broadcast information. With them I can function effectively at home, via directed information, without becoming out-of-date.

Before sending in this article for publication, I naturally composed it into a readable draft, and notified a friend or two by electronic mail, to get their usual good ideas and have them catch mistakes.

One was the Gerry Despain just mentioned. I thought his comments could add a certain flavor, and validate and augment my own experience:

"Being at Camelback, I sometimes feel left out of 'broadcast' information myself. There is not a large group of us, and some of the people I would like to be in closer communication with are at the Deer Valley Plant.

"So even working in an office sometimes requires overt action to involve oneself in broadcast information. It is partly to satisfy these needs that I spend time every day in reading the Multics forums. Even for people at Deer Valley, the forums give access to discussions carried on by people all over the country (in fact, the world) that is more like broadcast than directed information.

"One of the things I miss at home is easy access to a printer—or in your case a high-speed printer. I have only one telephone line. My wife wishes I had 25.

"Working at home you are more susceptible to problems. For example, parity errors and "RETRANSMIT LAST LINE" for 24 hours every time we have a heavy rain (or our neighbor waters his garden—in the middle of which is our telephone post), or lightning hits on mountaintops.

"Is Moon Mountain really of volcanic origin?"

My kudos to the electronic meeting software are endorsed, and the answer is yes, Gerry.

—Bob Bemer
Moon Mountain, Arizona

If you'd like to share your opinions, gripes, or experiences with other readers, send them to the Forum Editor, DATAMATION, 875 Third Ave., New York, NY 10022. We welcome essays, poems, humorous pieces, or short stories.

Annals

of the History of Computing

SPECIAL ISSUE

IBM 650

IBM Expands the Computing World

Cuthbert C. Hurd, Editor

Prolog • Cuthbert C. Hurd

Announcement

IBM Announcement Letter, July 2, 1953 • T. V. Learson and C. C. Hurd

Engineering

The SSEC and the IBM Type 650 • Ernest S. Hughes

IBM Magnetic Drum Calculator Type 650 • F. E. Hamilton and E. C. Kubie

The IBM 650 Magnetic Drum Calculator • George R. Trimble

The IBM 650 and the Woodenwheel • E. L. Glaser

Universities

The IBM 650 and the Universities • Bernard A. Galler

A Computer for Carnegie • G. L. Bach

Two Thousand Words and Two Thousand Ideas—The 650 at Carnegie • Alan J. Perlis

Information Processing Language V on the IBM 650 • Herbert A. Simon and Allen Newell

The IBM 650: An Appreciation from the Field • Donald E. Knuth

GAT: An Early Compiler and Operating System • Bruce Arden

Educational Experience with the IBM 650 • John G. Herriot

Programming Aids and Applications

Computation Seminar, August 1955

Nearly 650 Memories of the 650 • R. W. Berner

FORTTRANSIT Recollections • David Hemmes

Wolontis-Bell Interpreter

The IBM MDDPM—Some Recollections of a Great Machine • Barry Gordon

The IBM 650 at Savannah River • R. R. Haefner

The Solution of Simultaneous Equations • G. Truman Hunter

Epilog • Cuthbert C. Hurd

Index to Volume 7, *Annals of the History of Computing*

8
/
1

About this Issue

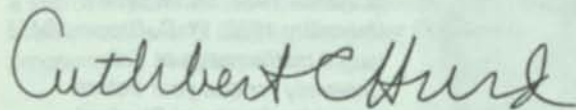
This special issue on the IBM 650 is a companion to the April 1983 issue of the *Annals*, "Special Issue, IBM 701, IBM Enters the Computing Field" (Volume 5, Number 2). We are publishing it on about the thirty-second anniversary of the delivery to a customer of the first machine.

The IBM 650 revolutionized the infant computer industry. For example, an order of magnitude more 650s were sold than the total number of computers that existed worldwide at the time of its development. The 650 turned a powerful sales and service organization away from control panels (plugboards) and toward stored programs. It proved that computers were for Everyman, and it initiated the momentum in technology that is known today as "high tech."

Readers who are interested in more detail concerning the 650—from a different point of view—may refer to *IBM's Early Computers*, by C. J. Bashe, L. R. Johnson, J. H. Palmer, and E. W. Pugh, MIT Press, 1985. We thank IBM for its kind permission to publish excerpts from a variety of IBM publications.

I would like to acknowledge my great indebtedness to the authors in this issue and to other contributors, mentioned or not, as well as to the referees who were led by Robert F. Rosin.

The assistance and support of Bernard A. Galler, Arthur L. Norberg, Henry S. Tropp, and especially Rosamond W. Dana are deeply appreciated.



Cuthbert C. Hurd
Editor, Special Issue

Contributors



Bruce W. Arden (B.S.E.E. Purdue 1949; M.A. University of Michigan 1955; Ph.D. University of Michigan 1965) designed control panels for an IBM cpc at the Allison Division of General Motors in 1950. In 1951 he joined the Willow Run Research Laboratories at the University of Michigan where he worked on the design of the BOMARC missile and the MIDAC computer. He was a research associate at the Statistical Research Laboratory during the IBM 650 era. He later served on all professional levels in Michigan's Communication and Computer Sciences Department and ultimately became its chairman. In 1973 he chaired the Electrical Engineering and Computer Science Department of Princeton University, where he is currently the Arthur Le Grand Doty Professor of Engineering.



George Leland Bach (A.B. Grinnell College 1936; Ph.D. University of Chicago 1940; LL.D. Grinnell College 1956; LL.D. Carnegie Institute of Technology 1967) is currently Frank F. Buck Professor Emeritus of Economics and Public Policy in the Graduate School of Business and the Department of Economics at Stanford University. He came to Stanford in 1966 from CIT where he served as dean of the Graduate School of Industrial Administration, chairman of the Department of Economics, and Maurice Falk Professor of Economics and Social Science. From 1941–1946 Bach served as economist and special assistant to the board of governors of the Federal Reserve System; he has served as a special consultant to that board since 1963.



Robert W. Bemer (A.B. Albion College 1940), starting in 1949, worked at Rand, Lockheed, IBM, Univac, Bull GE, General Electric, and Honeywell. At IBM he worked on PRINT I, FORTRANSIT, Commercial Translator, and XTRAN (pre-ALGOL). He says the COBOL ID, Environment Divisions, and the Picture Clause are due to him, as are the terms "COBOL," "CODASYL," and "software factory." He contributed six characters to ASCII and

Nearly 650 Memories of the 650

R. W. BEMER

Editor's Note

Of the many interesting aspects of Bemmer's career, one that has impressed me most is his ability to move back and forth between what we at one time perceived as the two classes of computers: "scientific" and "commercial." Note his contributions to the IBM 650, 704, and 705, as well as to compilers (perhaps still called automatic coding devices) and interpreters of quite different sorts.

I do not know B. C. Borden or where he/she may be, but Bemmer's quotation about cooperation between

Carnegie and IBM to produce FORTRANSIT is welcome and helps to complete the record.

I especially appreciate Bemmer's generous sketches of his colleagues at Lockheed. Here he closes one loop: he discusses Fletcher Jones, a founder and former president of Computer Sciences Corporation, whose foundation endowed the professorship Donald Knuth holds at Stanford.

I started at Lockheed Aircraft Corporation on January 3, 1954, as Group Engineer-Research in the Missile Systems Division. I was brought in from Marquardt Aircraft to form the new computer group, of which I was employee No. 1. Robert Birdsall had interviewed me, and he was very important to the success of our activity.

My task was to install two machines: first an IBM CPC, and afterward the IBM 650. Both were ordered before I got there, mostly by Art Hubbard, my new boss, manager of the Mathematical Analysis Department. They rarely make bosses as nice as this one. In actuality, I was to build a group for *both* digital and analog computers (Jack Sherman and Bob Prince were

my prizes in the analog area). I had persuaded Birdsall that there was such a short supply of (and a high demand for) programmers that we would have to offer them salaries about 25 percent above that of the typical aircraft engineer!

Armed with such authority, I intended to build an exceptional group. Unfortunately, before my authority to hire anyone was real, the first person to come along was Olney Robert Perry (brother of Malcolm Perry of SABRE development). He noted that both of our fathers were school superintendents. I was so sorry to lose him (he couldn't wait for that job when it was so easy for him to get another) that I vowed to hire the next school superintendent's son without question (he was Ralph Rea, who did very well at NSA).

We got the CPC working with programs that Bob Bosak and I had developed. Bosak was in charge of a similar group at Lockheed Marietta (Georgia), which was also to get a 650. As friends, and former sharers of an apartment in Hollywood (where my rental piano was used by Stan Freberg), we had a mutual respect and decided to pool our software system efforts.

We got the third IBM 650, which was the first one used for scientific work. It was Serial No. 10, so we assumed that IBM had kept the first seven. The February 16, 1955, issue of the *Lockheed MSD Star* shows a picture of me at the console, surrounded by Hubbard and Willis Hawkins, chief engineer. The

© 1986 by the American Federation of Information Processing Societies, Inc. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the AFIPS copyright notice and the title of the publication and its date appear, and notice is given that the copying is by permission of the American Federation of Information Processing Societies, Inc. To copy otherwise, or to republish, requires specific permission.

Author's Address: Bob Bemmer Software, Inc., 2 Moon Mountain Trail, Phoenix, AZ 85023.

Categories and Subject Descriptors: C.1.1 [Processor Architectures], Single Data Stream Architectures; K.2 [History of Computing]—hardware, IBM 650, people, software. *General Terms:* Design. *Additional Key Words and Phrases:* Lockheed, programming aids

© 1986 AFIPS 0164-1239/86/010066-06\$01.00/0



Left: Bob Bemer, Art Hubbard, and Willis Hawkins inspecting the new 650 at Lockheed's Missile Systems Division in February 1955. *Above:* Hawkins, Bemer, and Hubbard at the console.

newspaper featured the news that "the IBM computer is the first of its kind west of the Mississippi River."

The joint nature of our work is well described in Bosak's paper in that classic Technical Newsletter No. 10 (TNL10), published by the Applied Science division of IBM. He said, "Our planning began in April of 1954 with the formation of a steering committee. . . . The system was complete when we came to Endicott the last of November."

I remember testing at Endicott for two reasons. In my paper "Computing Prior to FORTRAN" (1982), I mentioned that long running times and odd hours demanded Coca-Cola, which, to be palatable, required a little rum. The cola was easily obtainable; I had to smuggle in the rum. It may have been the first to be consumed on IBM premises. The second memory is of Lucy Siegel of IBM, who wanted us to use IBM's one-per-card instruction loader. I said, "What for?"; we had a five-per-card loader ourselves thanks to some tricky board wiring. Some archivist may wish to study the wiring diagram and code in the Lockheed paper in TNL10 to see just what made it possible. I was proud of those diagrams.

Bosak's group did the arithmetic routines, while mine did the functions and utilities. All of the papers in TNL10 had specific authors except ours. At my insistence, we appeared jointly as the Mathematical Analysis Section of Lockheed Missiles. Brief descriptions of some of those authors and their 650 work are in an appendix here.

The second distinguishing feature of our presentation in TNL10 was the fact that it was printed entirely on the IBM 407 printer—not by typewriter. Our version of word processing consisted of keypunching, with partial correction effected by selectively reproducing the old card. That was if the columns did not change. If they did, we usually had to repunch the card. This

permitted us to publish in TNL10 not only the text but the entire coding for the FLAIR system (FLoating Abstract Interpretive Routines). The text did not look very pretty, but keypunches produced only uppercase. There may have been earlier uses of computing equipment for publication, but we were unaware of any. It reflects my early and abiding interest in computerized text processing.

650 Usage at Lockheed MSD

Our work in producing FLAIR (called FACS at Lockheed, Georgia) paid off because we got applications running quickly. Many started in February 1955 on that first machine. A second 650 system (Serial No. 37) was accepted in May, and a third in July. The local paper reported that it was up and running in 96 minutes, where the first two had taken three and a half hours each.

The 650 was reliable, beyond the expectations of IBM, as we viewed it. (I had a self-checking board on the very first CPC, at the Rand Corporation; I could see a bug growing by the increasing number of times the same computation was performed to get a correct answer.) We often ran 650s unattended (had IBM known it they might have taken them away, or at least raised a fuss). We had to. Some trajectory calculations ran for very long times, and we did not have a three-shift operation. We also ran them over weekends, after preparing the guards to inspect them on their rounds, with instructions on how to stop them if certain lights turned on.

IBM used the Lockheed MSD installation as a showcase for prospects. We received thanks from IBM Applied Science, under Don Penderly; Collins Radio; General Electric in Idaho; Fred Brown, IBM branch manager; and many others.

The Endicott Seminar

The Computation Seminar in Endicott (August 1-4, 1955) was very impressive. I had been to meetings of diverse computer people before, but they were not as uniformly high-powered as these 67 persons were, nor were they familiar with and focused on one specific type of computer. IBM, with profit ever in mind, nevertheless did a great service in all areas by creating this meeting. Oddly, the seminar was just prior to SHARE I, the first meeting of the user group of IBM scientific computers.

Almost all scientific usage of the 650 was done with floating-point routines, which we had to fabricate ourselves. The good news at Endicott was that floating point had been wired in. The bad news was that they had not asked any of us users about it. Dura Sweeney and George Trimble described their method in TNL10 (p. 108). They also carried the exponent 50 higher than actual, to avoid its own sign, but they put it at the right-hand end of the word, whereas software types put it at the left for direct magnitude comparison. I asked why. We were told that it saved hardware. Later, when I joined IBM myself, one of the two confessed that it wasn't really so.

Intermission

In December 1955, I left the 650 world by joining IBM to write a processor for doing scientific work on the IBM 705, which was a decimal and alphanumeric machine designed originally for commercial work. PRINT I was a considerable step up from the system we had done for the 650. Although PRINT I was interpretive in execution, an initial process removed as much redundancy as possible from the executing program. PRINT was also the first load-and-go system.

PRINT I was designed and built in the same office complex as FORTRAN, and I watched that work with interest. Design started in December 1955. I gave a talk to the Western JCC in February 1956. PRINT I was field operational by August. Obviously it was time for new work!

650 FORTRANSIT

I do not know who B. C. Borden is, but this is my opportunity to thank him or her for providing the only known published paper on FORTRANSIT (1958). Recent private polling seems to indicate that far more computer people cut their teeth on FORTRANSIT than on FORTRAN—due, of course, to the greater number of 650s in the field and the scarcity of 704s and 709s. This was particularly true in the universities, owing

to IBM's 60 percent educational grant plan. Borden said:

IBM's applied programming people worked closely with the "Computation Center" at Carnegie Institute of Technology with a view to developing a "processor" for these source statements. Priority was placed on time. We urgently wanted to have the 650 in a position to be able to accept FORTRAN statements and write object programs. As there already existed a compiler and an associated assembler for the 650, the decision was made to adapt and extend this system of existing programs and get into production. . . . It was immediately apparent that the language as it stood was in no way tied to the 704. All that was necessary was to write new processors for other machines and we would have a compatible language. Work started directly on both 650 and 705 processors.

The summary of the motivations is quite correct. The implication that both projects started simultaneously is not.

In 1955 I met Alan Perlis, who had worked on his compiler for the Datatron machine at Purdue, and was to continue for the 650 at Carnegie Tech. My air-travel records show that I went from LaGuardia to Pittsburgh on December 6, 1956. There must have been some other purpose, but my annotation says "FORTRANSIT." I do remember walking on the diagonal of a snowy quadrangle, asking Perlis to permit me to incorporate the IT compiler behind a FORTRAN source front. He gave no argument against the idea, although I heard later that he did not think much of such a jury-rig business. No mention of the FORTRANSIT project was made in his paper about the IT compiler (Perlis and Smith 1957). He just agreed to send me the source program, which he did shortly after.

A 1957 January ditto sheet headed "PRO RES" gave a capsule description of each person in IBM Programming Research, with their projects. Dave Hemmes and Otto Alexander were listed as having come to IBM the previous November, and were working on the FORTRAN language to 650 compiler.

It is true that some efficiency losses were sustained by the cascading process in those days, but I think the fact that FORTRANSIT came out in August 1957, just two months after 704 FORTRAN, justified the method. A true FORTRAN compiler for the 650 did not come out until 22 months later.

The early welcome of FORTRANSIT led to the development of 705 FORTRAN. We started that project at GUIDE, the IBM commercial users group, in September 1958, with Bill Selden in charge.

Following this I gave little attention to the 650, becoming engrossed in the Commercial Translator

effort and ASCII. I did maintain an interest in language processors for all machines, however. I started to collect five each of manuals for various automatic coding systems. Eventually this collection turned into the basis for both the ACM repository and the library of the British Computer Society. At the time of collection I served as editor of the Techniques section of the *Communications of the ACM* (the first four years, 1958–1961). As filler, I published a list of various systems available for several machines. The May 1959 issue had these systems as known for the 650:

ADES II	ESCAPE	OMNICODE
APT	FAST	RELATIVE
BACAIC	FLAIR	RUNCIBLE
BALITAC	FORTTRANSIT	SIR
BELL	FORTTRUNCIBLE	SOAP I
BELL L2 L3	IT	SOAP II
CASE SOAP III	IT 3	SPEEDCODING
DRUCO I	KISS	SPUR
EASE II	MITILAC	GAT-2 (RAMAC)
ELI	MYSTIC	

Operational data on some of these are contained in the survey I did for Subcommittee Five of ISO TC97 (Computer Standards), as published in *CACM* (March 1963). Of course, the real 650 FORTRAN done by Flo Pessin should be added.

REFERENCES

- Bemer R. W. 1982. "Computing Prior to FORTRAN." *Proc. Natl. Comput. Conf.*, Reston, Va., AFIPS Press, pp. 811–816.
- Borden, B. C. June 9–10, 1958. "FORTRANSIT, a Universal Automatic Coding System for the IBM 650." *Proc. Canadian Conf. for Computing and Data Processing*, Univ. of Toronto, pp. 349–359.
- CACM*. May 1959. "Automatic Programming Systems." *Comm. ACM* 2, No. 5, p. 16.
- CACM*. March 1963. "Survey of Programming Languages and Processors." *Comm. ACM* 6, No. 3, pp. 93–99.
- Perlis, A. J., and J. W. Smith. April 1957. "A Mathematical Language Compiler." In *Automatic Coding*, Franklin Institute Monograph No. 3, pp. 87–102.

APPENDIX

Lockheed MSD 650 Programmers

Richard Middleton (407 and 533 utility panels) had worked on the graveyard shift at the Rand Corporation with Bob Nash and me. Our group once worked 70 straight hours without sleep (weekend) on a tiebreaker for the *Los Angeles Times* "Tangletowns" puzzle. We had thought to do it by punched-card equipment (such was the early fascination that we thought little of the ethics), but soon found that this was a problem for minds, not computers. Our prizes included a Hawaii trip, swimming pool, fur coat, and \$1000 cash. Middleton was then studying optometry, but I persuaded him to give up the profession to work at MSD (Missile Systems Division).

Irene Brown worked on the logarithm routines. She came from a university in the Boston area, and upon arrival asked why I was paying her \$110 per week when she was making only \$60 at the university. I told her it was because that was what the men doing the same job as hers were making. (I am proud of my pioneering efforts at women's liberation.)

Charles Wimberly did the arctangent routine, but was more preoccupied with coding for the UCLA SWAC.

Ben Handy came to us later, but I met him while doing the SWAC work. He then went from UCLA to the startup Computer Research Corporation (CRC) before I turned him to Lockheed.

Don Jackson was lured from the California Division of Lockheed. It took several months to get him, because he hadn't as much confidence in his own competence as I did. Even then I actually had to (appear to) demand that he buy a topcoat and briefcase to accompany me to Endicott for the November checkout.

Elaine Gatten is listed for the floating-to-fixed decimal routine; she was our first keypunch operator at Marquardt.

Albert Podvin (sine-cosine routine) is a classic example of how unorthodox my hiring methods could be. Lockheed then would not permit me to hire programmers without undergraduate degrees, so I decided to use a subterfuge. I said that I would hire him as a keypunch operator (and with such a salary). If he panned out, we would try to use some stratagem to name him as a programmer. Meanwhile he learned to program so fast that it seemed unbelievable. I thought I had a pretty good method for computing sines, but Podvin did one, using the $\sin 3\theta$ formula, that ran a lot faster! Later Al came with me to Univac, and was a major designer of EXECs. After that he went to IBM, eventually becoming a principal consultant to B. O. Evans, who ran the IBM System/360 project.

Richard Talmadge succeeded me as manager of the computation section at Marquardt. We later persuaded him to come to Lockheed MSD. The rest of us coded with pencils that had erasers on the end. Talmadge used a pen. If he ever made a mistake I did not hear of it. Later he joined IBM in Los Angeles and was in charge of the Commercial Translator project for the 709. He and his people produced a processor so good that it inhibited acceptance of COBOL by 709 users, even after IBM had switched formally to supporting COBOL.

David Hemmes started in computing at the Rand Corporation, came to Marquardt, then Lockheed MSD, and later to IBM in New York. The New York move was for the benefit of the 650—I needed him for the FORTRANSIT project.

Fletcher Jones was another later arrival, coming to MSD from North American Aviation. He saw the 650, did a little coding for it, and seemed to disappear for most of the two weeks he worked for me. He then announced that larger machines, like the 701, entranced him more, and he went back to North American again, where he had been dickering most of those two weeks.

FORTRANSIT Recollections

DAVID A. HEMMES

Editor's Note

When I spoke with Hemmes about doing a paper on FORTRANSIT, he was almost as eager to have the attached picture reproduced as he was to write the article.

Hemmes quotes from the FORTRANSIT Programmer's Reference Manual. A further reading of the manual reveals the extent to which the support of programming aids had matured. For example, the manual states that the system will be maintained by the IBM 650 Applied Programming Department. It

names the following "continuing contributors to FORTRANSIT": I. C. Liggett (manager), O. E. Alexander, A. M. Paster, H. M. Starr, P. I. Strauss, B. D. White, and L. S. Woo. Also, there was a 650 program librarian, and the numbers of each of eight different card decks are listed along with the names of the system deck packages. Thus, whereas Elmer Kubie and George Trimble wrote the original 650 programming aids, IBM now had a full-fledged Programming Department.

Someone else had to tell me that FORTRANSIT was the first compiler to make it possible for a source code to compile on more than one computer. Actually, the FORTRANSIT language was a subset of the original FORTRAN language. Thus FORTRANSIT made it possible to compile a subset of the FORTRAN language on both an IBM 704 and an IBM 650.

I once wisecracked that if I had known FORTRANSIT would enjoy such widespread use, I would have worked harder on all parts of it. Now I can honestly say that if I had known I would be writing about it 27 years later, I would have kept notes. Since I did not, I must rely entirely on my memory, and I cannot guarantee complete accuracy of this report.

© 1986 by the American Federation of Information Processing Societies, Inc. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the AFIPS copyright notice and the title of the publication and its date appear, and notice is given that the copying is by permission of the American Federation of Information Processing Societies, Inc. To copy otherwise, or to republish, requires specific permission.

Author's Address: 1534 Klamath Drive, Sunnyvale, CA 94037.

Categories and Subject Descriptors: C.1.1. [Processor Architectures], Single Data Stream Architectures; K.2 [History of Computing]—FORTRANSIT, hardware, IBM 650, people, software. General Terms: Design. Additional Key Words and Phrases: programming aids.

© 1986 AFIPS 0164-1239/86/010070-073\$01.00/00

Early in 1957 the original FORTRAN, "a compiler for the IBM 704," was about to be released to a community of users. John Backus and his group were breathing a big sigh of relief and looking forward to new things. At the same time, Alan Perlis and some graduate students (J. W. Smith and H. R. Van Zoeren) at Carnegie Institute of Technology were touting a compiler for the IBM 650 called Internal Translator AKA Interpretive Translator (IT). IBM was negotiating an agreement with Perlis for use of IT, but I never knew the details. All I knew was that Bob Bemer, who was running some sort of rival camp to Backus, came to me and said that we wanted to perpetuate FORTRAN as a language. He said FORTRAN should be a source language for more machines than the 704. He went on to say that the 650 was the top candidate for the next machine to accept FORTRAN source code. What I suspect is that he was trying to ace out Perlis before the IT language became more widely used.*

My initial reaction was disappointment when he assigned me to this project. I was new to IBM and had

* The Programmer's Reference Manual states on page 3 that FORTRANSIT I requires an alphabetic device and then brags that there is no requirement for a special character device even though the input cards for the system contained special characters. This is only partially true. FORTRANSIT I and II required neither device. FORTRANSIT I (S) and II (S) required both.

had some 650 experience prior to joining IBM. I had expected to work on bigger things than the 650 when I went to work at World Headquarters. Bemer flattered me into believing that I was the only person in the world who could write the input/output program. Although I was not, he was actually not far off target: the input/output section was about 65 percent control-panel wiring and 35 percent programming, and I had a strong background in control-panel wiring.

I teamed up with Otto Alexander, Florence Pessin, and Leroy May, and we went to work. It was a fortunate combination of talents. I wrote the input/output, Florence wrote the arithmetic section, and Otto and Leroy did the rest. We all helped one another to debug.

What we produced is best described by the introduction to the Programmer's Reference Manual, "FOR TRANSIT: Automatic Coding System for the IBM 650," IBM Applied Programming Department, IBM, 1957. (Note that FORTRANSIT was originally two words.)

The FOR TRANSIT system makes available to 650 users the language of FORTRAN, and thus there now exists a language system which is common to two IBM computers, the 704 and the 650, making possible the compatibility of programs. FORTRAN, a language closely resembling the language of mathematics, is designed primarily for scientific and engineering computation. One of its main purposes is to provide the scientist with an efficient means of writing 704 programs requiring no knowledge of the computer and a relatively short period of training. FOR TRANSIT provides similar advantages in the 650 area by superimposing a translator on the compiler* developed at the Carnegie Institute of Technology by Dr. A. J. Perlis and his associates.

The FOR TRANSIT system consists of three major parts:

1. The translator, FOR TRANSIT, which accepts FORTRAN statements and produces corresponding IT statements.
2. The compiler, a modification of IT, which accepts IT statements and compiles 650 instructions in symbolic (SOAP II) language.
3. The assembler, a modified version of SOAP II,** which produces an optimized machine language program from the symbolic instructions.

* Internal Translator (IT) A Compiler for the 650, by A. J. Perlis, J. W. Smith, and H. R. Van Zoeren, Computation Center, Carnegie Institute of Technology. (See IBM 650 Library Program Abstract 2.1.001.)

** SOAP II, Symbolic Optimal Assembly Program for the IBM 650 Data Processing System, by S. Poley, Service Bureau Corp. (See SOAP II Programmer's Reference Manual, Form 32-7646.)

At this point in my narrative, I would like to insert excerpts from "Twenty-Five Years of FORTRAN," a National ACM Lectureship Series Presentation (1981-1983), by J. A. N. Lee. I have no quarrel with



Dave Hemmes in his 1928 Model A racing car with a Chevrolet Corvette engine at the Westhampton drag strip in 1957, the year FORTRANSIT was developed.

this account of the development of FORTRANSIT, so I see no need to rewrite or parallel it. What I will do is amplify it (in italics).

The 704 team led by Backus was still very busy in early 1957 completing the final stages of debugging and trying to get the system ready for distribution. Thus no one could be spared from that group to start a new project, but there were people who had been close enough to the activity to parallel their work. One was Bob Bemer. At the Langdon Hotel on 56th Street in New York City, he assembled a 650 team consisting of Otto Alexander and David Hemmes. Neither of them had any previous experience with the kind of work to be undertaken; later they were joined by Flo Pessin, who was equally unprepared for the task (and Leroy May—D.A.H.).

Cascading Implementation

Bemer noted that there was another significant activity in progress that was to be implemented on the 650, though it was not originally intended for that machine; it was the IT (Interpretive Translator) system being developed by Alan Perlis at Carnegie Institute. IT compiled a much simpler language into the assembly language of the IBM 650 (SOAP), which in turn was assembled into the object code for the machine. Thus the concept was developed to "cascade" the implementation from FORTRAN to IT to SOAP and hence to object code in four passes (*counting the two passes of the assembler*).

FORTRAN > IT > SOAP > object

Quick Implementation

In spite of the inexperience of the crew, a version of FORTRAN was available for the 650 only a few months after the delivery of the 704 version and the expenditure of only 4-5 man-years of work [*instead of the 30 estimated man-years invested in the 704 FORTRAN*]. To

accomplish this the language was a subset of that implemented for the 704, but this was consistent with the fact that the 650 was an even smaller machine than the 704. Part of the motivation for this effort was the fact that IBM expected many more 650s to be installed in universities than 704s, especially since IBM was now willing to offer a 60% educational grant to those institutions that used the systems for administration, scientific computation, and business data processing classes. FORTRAN was to be the "hooker" of this new generation of students. To accelerate the implementation, . . . optimization was omitted.

Bemer and Pessin

Bemer joined IBM Programming Research in 1955 after a career in the aircraft industry and was appointed manager of Programming Systems in 1960. In the "FORTRAN years" Bemer was active in many other ways that furthered the development of programming languages. . . .

Flo Pessin was given the task, by Bemer, of writing the arithmetic scanning routines for this new version of FORTRAN, but first she invented the name of the system—FORTRANSIT. Based on the cascading approach that Bemer had suggested, recognizing the contribution of IT, and being a double-croctic addict, she coined the name as a three-way pun. (*I always thought I had thought up the name, but I won't quarrel—D.A.H.*) One of the difficulties facing Pessin at this time was her lack of experience in preparation for this task, plus the fact that the 704 team had created no documentation (though they were no different from most other implementers) and there was no help offered by the 704 group. Thus she was forced to invent new techniques of compilation, and like the others did not document them because she really did not know that what she was doing was so innovative.

The first challenge for the input/output section of FORTRANSIT was that the FORTRAN source language used alphabetic and special characters, and I assumed that the 650 accepted only numeric characters. The 650 read cards and wrote (punched) cards; that was its input/output. The 650 accepted only one hole per card column which allowed a choice of a 0,1,2,3,4,5,6,7,8, or 9 punch in each column. Alphabetic characters required two holes per card column, and most special characters required three.

The FORTRANSIT requirement was for the 650 to accept a statement such as

$$\text{ROOT} = (-B + \text{SQRT}(B^2 - 4.0 * A * C)) / (2.0 * A)$$

on a punched card. This was accomplished by an intricate linking of coselectors, pilot selectors, column splits, read-code selectors, punch-code selectors, and a half-time emitter to break up the "zone punches" and the numeric punches and get them on the drum memory from which they could later be retrieved.

Actually, it was more complex than the above description sounds because some idiosyncrasies of the 650 had to be discovered and taken advantage of in order to get it to do something it was specifically designed not to do. Once the card was read, the information had to be reassembled in the drum memory to represent the original character that was punched on the card. Once on the drum, alphanumeric and special characters were no problem. The problem was getting them there.

A lot of logic was associated with this reassembly program, and more time was spent on the flowchart than was spent writing instructions.

While all this was going on I was either ignoring or unaware of the fact that an alphabetic device option and a special character device option were both available to 650 users at extra cost. I became aware when someone, probably from Marketing, learned that I was developing a system to read FORTRAN statements without the alphabetic and special character options and tried to get me turned off. I think it was Bemer who told him to go away.* At any rate, I continued to ignore him, as my design goal was to produce a control panel and associated program that would work on all 650s with no restrictions and no need for options.

This goal of no restrictions went further than just no requirement for the options. Not all the 650s in the field were alike. Engineering changes had been made, each starting at a certain serial number. While this had no effect on using the machine in the normal way, it did affect some of the idiosyncrasies the control panel needed.

With the help of Engineering, I was able to locate one of each kind of change right in New York. Then came a period of traveling to each installation, sometimes riding a subway with the control panel on my lap, and testing the panel on each of the machines. With minor modifications to the wiring, the board was successful on all of them.

I wrote the program; it worked smoothly. Then Otto Alexander rewrote the program to take up less space on the drum. Although I had a small case of hurt pride over the rewrite, I said nothing because Otto was right. Space was at a premium on a 2000-word drum memory, although I did secretly enjoy it when I discovered some bugs in his rewrite and corrected them.

A word about Otto Alexander might be appropriate here. Otto did a lot of things right, besides rewriting

* *Editor's Note:* Bemer comments: "Hemmes is correct that FORTRANSIT was the first major compiler of source code on more than one computer type. That was my goal. It was the start of [my] devotion to standards. . . . I suggest that this was my motive in acting, not to 'ace out' Perlis. I was not a 'rival camp' to Backus, particularly because at that time he was my superior."

my program. He did them quietly, with no fanfare, and he did them well. Pessin has received a lot of praise (and rightly so) for her arithmetic section, my Last Great Board Wiring Job was noteworthy (although not needed if one had the options), and Leroy May was where he was needed when he was needed, but it was Otto who pulled the thing together.

The marketing group began to prevail in their efforts to sell options. As a result, in a short time, in addition to the original FORTRANSIT I, there was a FORTRANSIT I (S) that required a machine with both alphabetic and special character devices, as well as a FORTRANSIT II and a FORTRANSIT II (S) that produced programs designed to execute on a 650 equipped with index registers and automatic floating-point arithmetic.*

The original control panel was used for FORTRANSIT I and FORTRANSIT II, while a simpler version was designed for FORTRANSIT I (S) and FORTRANSIT II (S). The SOAP (assembly) pass had its own control panel.

While not up to the complexity of the Card Programmed Calculator (CPC) control panels, the FORTRANSIT control panels might qualify as the Last Great Board Wiring Job. They were used by a lot of people.

The FORTRANSIT manual was a direct copy of the original FORTRAN manual. It had the same cover and as closely as possible followed the same style. It even used the same format and some of the same examples. It was fun to write. I used a Leroy set for the control-panel illustrations; they were not redrawn. I can't say the same for the text. I had the manual about 80 percent complete, working almost completely unnoticed, when I began to get a lot of help. That was fine—I needed help to wrap it up, get it printed and released—but I was visibly annoyed when the technical writers edited out all my jokes.

One time during the development of FORTRANSIT a conference of some sort was held at Ohio State, and IBM was asked to be on the program and discuss FORTRANSIT. I agreed to go even though I had broken my ankle trying to learn to ski and was walking around in a cast. Before I left New York I bought a pair of rubber overshoes big enough to fit over the cast. It was a good thing I did because there was snow on the ground in Ohio. I was given accommodations in a

student dormitory, and on the day I was to speak I pulled on the overshoes, buckled them, and clumped across the campus to the auditorium. When I walked out on the stage I must have been a comical sight. I was conservatively dressed, as would be expected of someone from IBM, but I had on these huge black shoes.

I was not very well prepared and had planned to discuss the three-pass approach we were taking with FORTRANSIT and wing it from there. Perhaps I would get a few questions I could answer and then escape. I need not have worried about lack of preparation. I was only a few minutes into my talk and was describing the subset of the FORTRAN language that would be source code for FORTRANSIT when there was a howl from the back of the auditorium. The howler began yelling that I had it wrong. He was saying things like, "What do you mean, you are using FORTRAN as the source language? That's not the way it is supposed to be!" He began moving down the aisle toward the stage, and I recognized Alan Perlis. I could hardly believe what was happening and what he was saying. There I stood like an idiot in those ridiculous rubber boots while Perlis, having gained the front of the room, took over the presentation and stood with his back to me giving his own presentation on the merits of IT and appealing to the crowd to rally to his side because he had been duped. He told them he had no idea that we were using a three-pass approach—FORTRAN to IT to SOAP to object code.*

Perlis finally subsided, and I made a few closing remarks and beat a hasty retreat. Walking alone through the snow back to the dormitory, I wondered if it was all an act. How could he not know our approach? Was he really fooled? I still do not know the answers to those questions.**

* *Editor's Note:* Bemer comments: "Yes, I ran the salesman off because I wanted the broadest possible audience for FORTRANSIT."

* *Editor's Note:* Bemer comments: "I confess that I don't remember any report from Hemmes on the Ohio State incident. I do not claim that he did not tell me, but the premise is thin, anyway. It had to be FORTRAN to IT to SOAP! It could not be otherwise. To have IT to FORTRAN we would have had to already have a FORTRAN compiler, and of course we didn't. And if we did, why would we use IT?"

** *Editor's Note:* Perlis comments: "Hemmes's comment about my behavior at the meeting is probably accurate, though I do not remember it. I was much younger then and probably felt very combative concerning FORTRANSIT vs. IT as an input language to the 650. His remarks show some of the passion which language adherents reveal when the importance of their language is called into question."