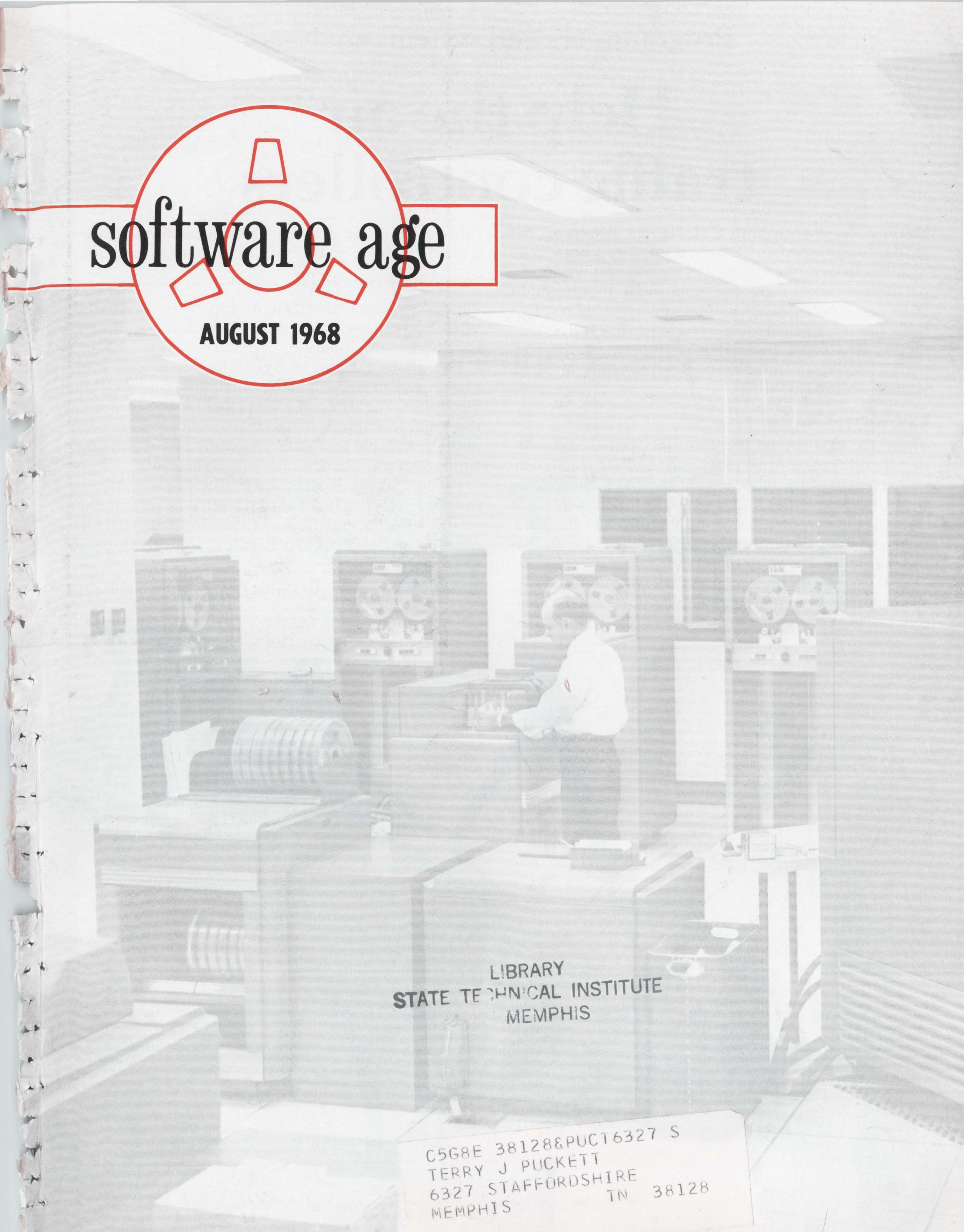
software age

AUGUST 1968



Programmers and systems analysts

Help the air traffic controllers

You can do it with both feet planted firmly on the ground—right next to IBM/Atlantic City's battery of System/360 modified units. Among this system's many peripheral devices are real-time radar inputs, flight strip printers, video and alphanumeric displays. But more important than glamorous hardware is the satisfaction of working on an urgent and important problem—assisting air traffic controllers in keeping airplane traffic running smoothly.

Crowded skies

There are 40,000 more planes in the United States today than there were 10 years ago. And air traffic control specialists are busy around the clock controlling them. Last year, the FAA's 18,000 specialists in flight control guided more than 48 million take-offs and landings, and by sometime in the 1970's it is predicted that number will have more than tripled.

IBM is helping these controllers by computerizing their information input and display systems. The computer also correlates aircraft tracks with flight plan information. (At present, the controllers position markers by hand on the basis of spoken information.)

Immediate openings

This is vital work and IBM/Atlantic City, N.J., needs additional experienced programmers and systems analysts to help us with it.

These jobs take an ability to interface with other people and to draw together insights from far-flung disciplines. You'd be using sophisticated programming tools and techniques to create and test hardware and multi-processing computer programs. SAGE and other similar real-time experience will be helpful.

If you'd like to participate in this work at Atlantic City, N.J., call Jim Dunn at (301) 921-7724 or write him at the IBM Federal Systems Center, Dept. BH1011, 18100 Frederick Pike, Gaithersburg, Maryland 20760.

IBM®

An Equal Opportunity Employer

software age

AUGUST, 1968 Vol. 2–No. 7

CONTENTS **PAGE**

Publisher — H. L. Rothra
Associate Publisher — David W. French
Advertising Manager — Norman Jacobs
Production Manager — Betsy Pavkovich
Director of Circulation — Howard Rogers
Circulation Manager — Judith Arnopolin

Circulation of this issue more than 91,000

SOFTWARE AGE is published monthly
by PRESS–TECH, Inc.
(312) 869–1244

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

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

CONTROLLED CIRCULATION POSTAGE PAID
AT MADISON, WISCONSIN

For more information, circle No. 6 on the Reader Service Card

A User Oriented Approach To Antenna System Programming

S. M. Chamberlain
Spectrodata, Inc.

and

R. L. Mitchell
Technology Service Corp.

INTRODUCTION

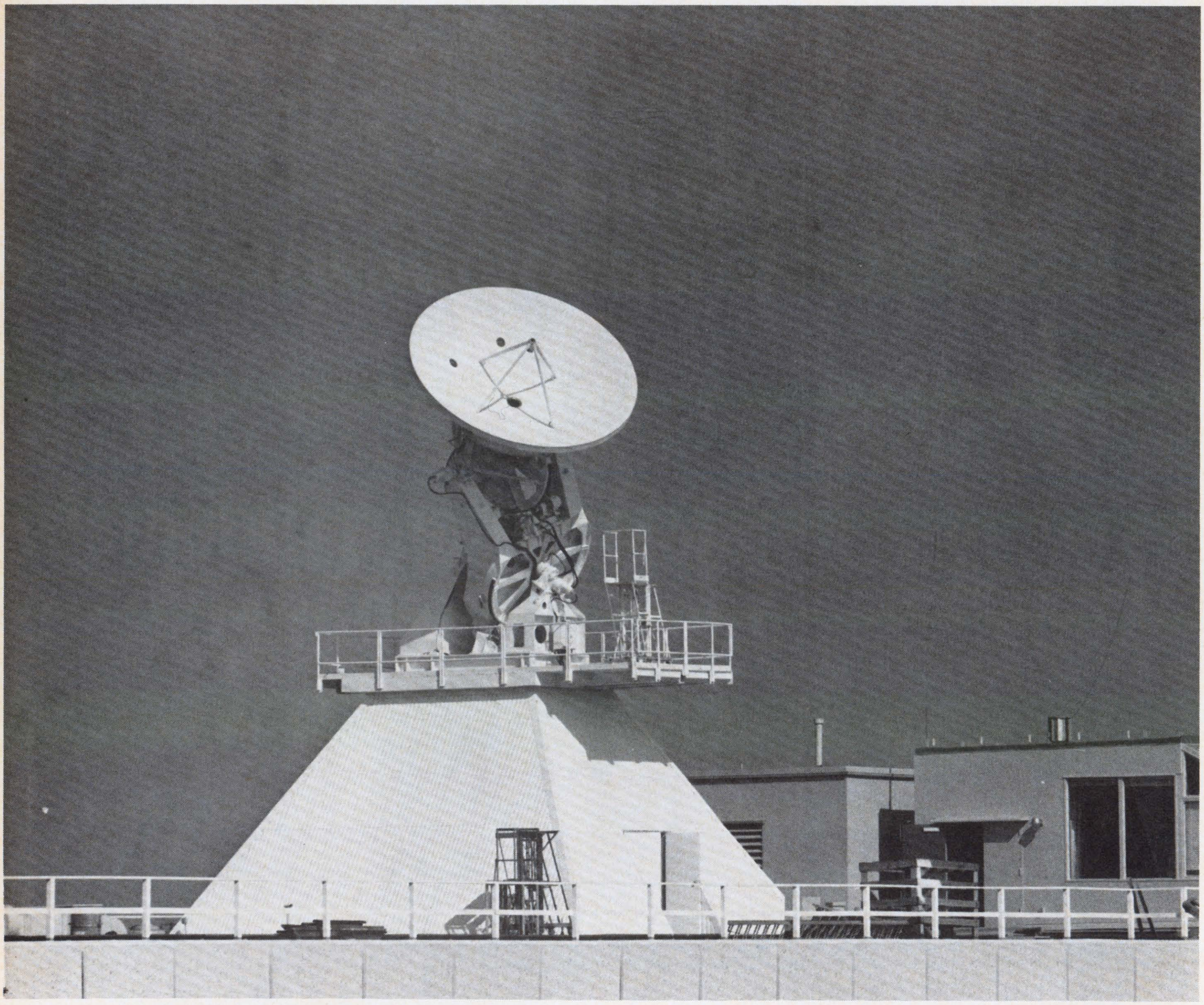
This paper describes a programming philosophy for a radio antenna system that is based on FORTRAN with specially written subroutines to perform each antenna function. The program is capable of operating in a real-time environment.

This programming philosophy is the result of many years of experience in attempting to design a suitable program for an antenna system at Aerospace Corporation that would be convenient and adaptable to the many users of the system. Since the requirements not only differed among the various users but also changed with time for a given user, either a staff of highly trained programmers with real-time programming experience would be necessary to meet this demand if the programs were written in assembly language or else a user oriented language would have to be developed. Since the Aerospace Corporation installation was small, the latter approach was chosen.

BACKGROUND

The Aerospace Corporation Radio Antenna[1,2] (Fig. 1) is a 15 foot paraboloid on an equitorial mount capa-

Figure 1. Aerospace Corp. 15-ft. millimeter wave antenna.



ble of operating at a frequency as high as 300 GHz (3×10^{11} cycles per second). The beamwidth at this frequency is about 1 minute of arc necessitating the use of extremely accurate pointing control. Because of this, it was decided to use a digital computer to control the antenna. This antenna was the first one digitally controlled. A digital computer offers other advantages in data handling and data reduction, but the principal function of the computer in this system is one of controlling the antenna position as a function of time.

The Aerospace Radio Antenna is used for observing selected radio sources, the sun, moon, and planets, earth satellites, and also for propagation experiments. The operating frequency is presently from 50 to 220 GHz, but the surface tolerance of the paraboloid is good up to 300 GHz or higher. The pointing accuracy is presently maintained at 0.001 radian, but about 0.00025 radian is theoretically possible. A functional block diagram of the system is shown in Fig. 2.

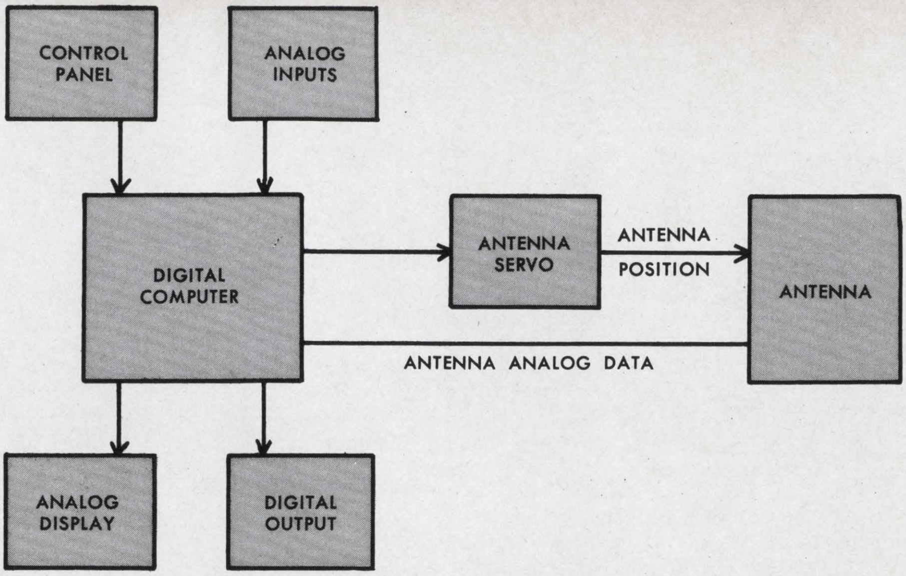


Figure 2. Functional block diagram of system.

The computer selected for this task was an SDS-920 with initially 4096 words of memory and presently 8192 words. The computer must perform in a real–time environment, i.e., the computer operations must be synchronized with time.

For the first several years, programs were written in assembly language (SDS-SYMBOL). It was initially hoped that once a good program had been written, it would not be necessary to make other than slight revisions. However, as is often the case, there was poor communication between user and programmer in addition to the changing requirements of the user. The programs that resulted were generally inconvenient to use since modifications that would make them more convenient were not easily done in assembly language. The low budget of this installation restricted the size of the programming staff.

What was clearly needed than was a programming philosophy that would be aimed at the user—one in which the user could write his own programs if possible. Such an approach is described in the next section.

METHOD

Scientific Data Systems had previously made available to Aerospace Corporation a FORTRAN system that could be interrupted (SDS Real-Time FORTRAN). With this as a starting point the next step was to modify the Real-Time FORTRAN system and to write subroutines that could do each function such as read real time, move antenna to a fixed point, move antenna with constant velocity, sample output, etc. In addition, subroutines were written to handle the data transfer from system control panel to computer. Subroutines to do mathematical type operations such as coordinate transformation and data reduction could be written in conventional FORTRAN by the user.

The subroutine structure uses a modular building block concept. There are basic subroutines that perform a single function such as taking a single reading from an A–D converter. Then there are higher level routines (which may be written in FORTRAN) that perform more complex functions such as moving the antenna at a constant rate while taking a series of readings from an A–D converter and averaging them. The user may also develop his own routines that perform functions unique to his own particular needs.

Normally, the user need only call the higher level routines. For special purposes, however, he may use the basic routines. Thus, the user can perform almost any desired function with the hardware and still retain an all-FORTRAN program.

There are three inherent advantages to this approach: convenience, versatility and reliability. The approach is convenient and versatile because programs are easily written and easily revised. The main program is generally a series of CALL statements. The user does not have to worry about such details as computer interrupt processing, settling times for hardware functions, bit manipulations for I/O with hardware, keeping track of time, closing servo loops, etc. Input/Output changes—a major factor in the Aerospace system—are almost trivial in FORTRAN. Generally, the user can write or specify exactly what he wants. The choice of FORTRAN as the basic programming language was also very convenient for the engineers and scientists since many of them were already familiar with it. This approach is reliable because once each subroutine is checked out—and this is usually easy—only the main program can be a source of errors. Debugging a main program of say 50 statements can usually be done in a straight forward manner. In addition, a hardware change requires a change in only one or two subroutines—not the main program.

There are, however, two limitations: computation time and memory requirements. Computation time is not a serious limitation for the general astronomical observation program on the Aerospace antenna since the response time of the antenna is of the order of a second both electrically and mechanically. But it is somewhat restrictive for propagation experiments where it is desired to scan the antenna rapidly across a reference point source. The present 8192-word memory is a serious limitation for large general purpose programs since the FORTRAN run–time system takes a significant portion of the memory. Future plans at Aerospace include additional memory.

EXAMPLE

An example of call statement that causes the antenna to track a point moving at a constant rate is

CALL TRACK (I1, ..., IN)

where the arguments specify the initial position and velocity of the moving point in antenna coordinates. One argument returns a flag that specifies whether the track is allowable (within the constraints of the mechanical limits of the antenna). If the track is outside the allowable limits of the antenna, the routine stops the antenna motion.

Using the modular concept, the user can build up a library of his own subroutines to perform complex functions. To track the sun and process the readings, a program might consist of

CALL INIT	Initializes program.
1) CALL POSIT(H,D)	Computes present position of sun in hour and declination angle.
CALL TKS(H,D)	Moves antenna to sun (if possible) and reads compressed data into table in COMMON.
CALL PCD	Reduces data in table and outputs results to magnetic tape.
CALL SWITCH (I)	Reads a sense switch on system control panel to determine when user desires to stop taking data.
IF(I) 1,2,1	
2) CALL POST	Does post processing of data on magnetic tape and types out results.
CALL FINISH	Stows antenna and terminates program.
END	

CONCLUSIONS

The programming philosophy described in this paper proved to be extremely advantageous for the particlar application of controlling an antenna system. Similar software systems could be developed for other computer controlled antenna systems—or for even most other computer controlled real–time systems. If a Real–Time FORTRAN system exists then only slight modifications might be necessary in addition to developing a library of subroutines. It is also possible to modify a standard FORTRAN system to perform the real–time functions, but this will often be a laborious task. The requirements based on available hardware for a particular installation must also be taken into consideration.

ACKNOWLEDGMENT

The authors are grateful to John P. Oliver of Aerospace Corporation for many helpful suggestions, most of which were incorporated into the final system. His assistance with the checkout phase of the program is also appreciated.

REFERENCES

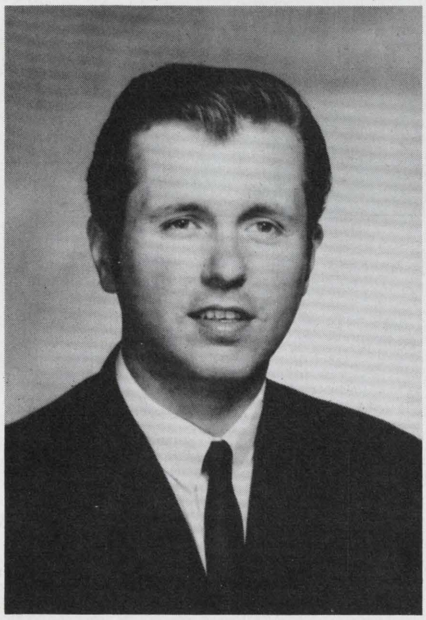
1. King, H. E., E. Jacobs, and J. M. Stacey, "A 2.8 Arc–min Beamwidth Antenna: Lunar Eclipse Observations at 3.2mm," *IEEE Transactions on Antennas and Propagation,* AP-14, No. 1, Jan. 1966, p. 82.
2. Jacobs, E. and H. E. King, "2.8-Minute Beamwidth, Millimeter–Wave Antenna–Measurement and Evaluation," *1965 IEEE International Convention Record,* Part 5, p. 92.

About the Authors

R. L. Mitchell, 30, received the B.S., M.S., and Ph.D. degrees in electrical engineering from Purdue University, Lafayette, Indiana, in 1960, 1961, and 1964, respectively.

From 1964 to 1968, he was a Research Engineer at Aerospace Corporation, Los Angeles, California. He is currently a Senior Scientist at Technology Service Corporation, Santa Monica, California. His primary fields of specialization are radar theory, communication theory, and numerical analysis, with emphasis on detection theory for fluctuating targets, development of computer programs for data display and analysis, and millimeter-wave radar.

Dr. Mitchell is a member of Tau Beta Pi, Institute of Electrical and Electronics Engineers, and the American Association for the Advancement of Science.

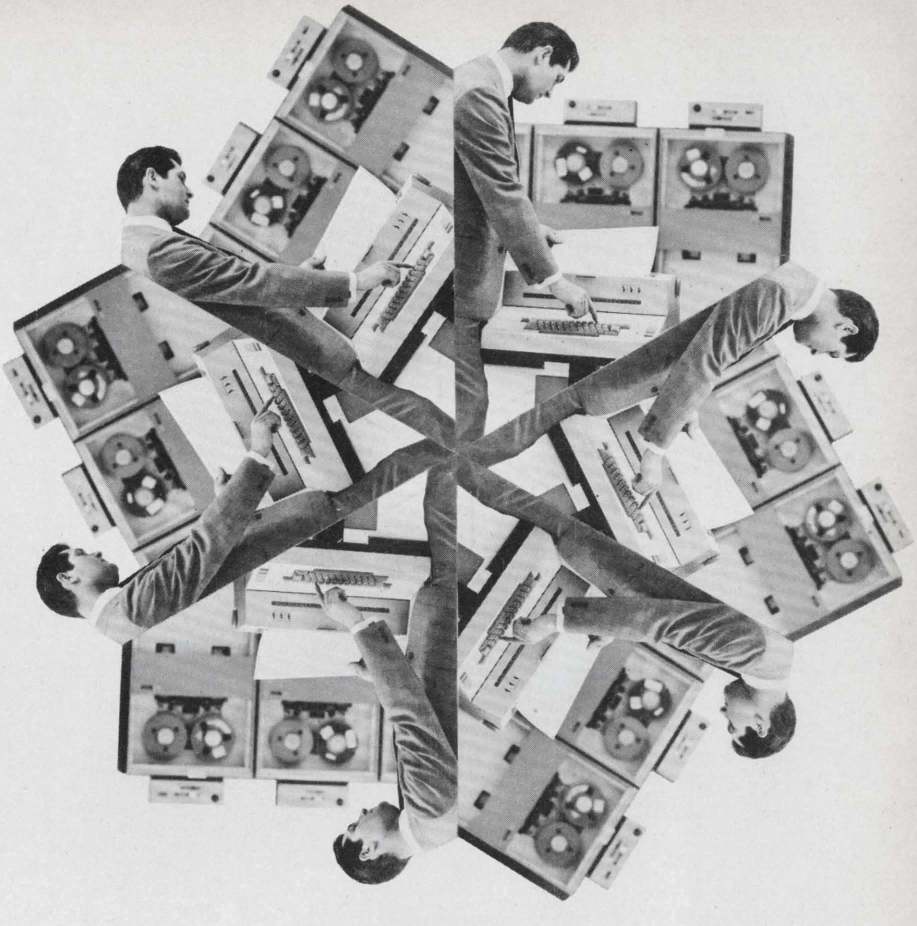


Steven M. Chamberlain, 30, graduated in 1961 from the University of Connecticut with a B.A. degree in mathematics and statistics.

From 1961 to 1964 he worked for the Autonetics Division of North American Rockwell Corporation as a computer programmer/analyst.

From 1964 to 1966 he worked with real-time computer controlled systems for Scientific Data Systems. Subsequently he became involved with real-time programming/analysis for Spectrodata, Inc., where he currently is a Member of the Technical Staff.

Mr. Chamberlain is a member of the Association for Computing Machinery.



software applications feature

The Application of Man-Machine Computing Systems to Problems in REMOTE SENSING

Philip H. Swain and Don A. Germann *

Laboratory for Agricultural Remote Sensing
Purdue University
West Lafayette, Indiana

This article describes a data processing system which coordinates a broad range of pattern recognition related techniques to aid in the design of pattern classifiers. It is shown how development of the user-system interface enhances the value of the system as a tool for research in agricultural remote sensing.

Introduction

■ Research at the Laboratory for Agricultural Remote Sensing (LARS) concerns the analysis by pattern recognition techniques of multispectral remote sensing data collected by ground-based, airborne, and (ultimately) satellite-borne instruments. A rather broad class of applications of this research is possible (not all of which have even been clearly defined as yet), and it is generally the case that each application requires a pattern recognition system with significantly distinctive characteristics. In such a situation it is necessary to have at hand an efficient and flexible method for designing pattern classifiers; in particular, for performing statistical analysis, feature selection, and calculation of other parameters needed for the realization of the classifiers. This article describes such a method and its implementation in the form of a system of computer programs. An important feature of this system is the considerable degree of user-system interaction through which is achieved the flexibility required by the research environment.

Data Handling and Data Analysis in Remote Sensing: General

Figure 1 shows a block diagram of the overall data flow for the Laboratory for Agricultural Remote Sensing Data Processing System (LARSYS). The principal data input is multispectral data collected on analog tape by a multichannel optical-mechanical scanner and tape recorder mounted aboard an aircraft.** The LARS Aircraft Data Handling Processor (LARSYSAH) prepares the data for use by the researcher: The data are edited, digitized, and calibrated and recorded on digital tape in a packed format (to reduce the physical volume). To make the data readily accessible to the user, line-sample coordinates (much like x-y coordinates) are added during the digitization process. A special computer subroutine is available which will read any desired area of data (specified by a set of line-sample coordinates) into core memory and pass it to the user's program in unpacked form.

Also available as part of LARSYSAH is a program which prints grey-level displays of selected data on a computer line printer. These displays, which are similar to black and white photographs of the ground areas over which the data were collected, are useful in coordinating the ground truth (see below) with the multispectral scanner data.[1,2]

* Under the direction of Dr. King-Sun Fu, Assistant Head of Research, School of Electrical Engineering.

** The airborne optical/infrared scanning equipments used in this research were made available by the U. S. Army Electronics Command (USAECOM), Fort Monmouth, N. J. on a no-cost basis to the University of Michigan (who collected the data) for use on contracts administered by USAECOM.

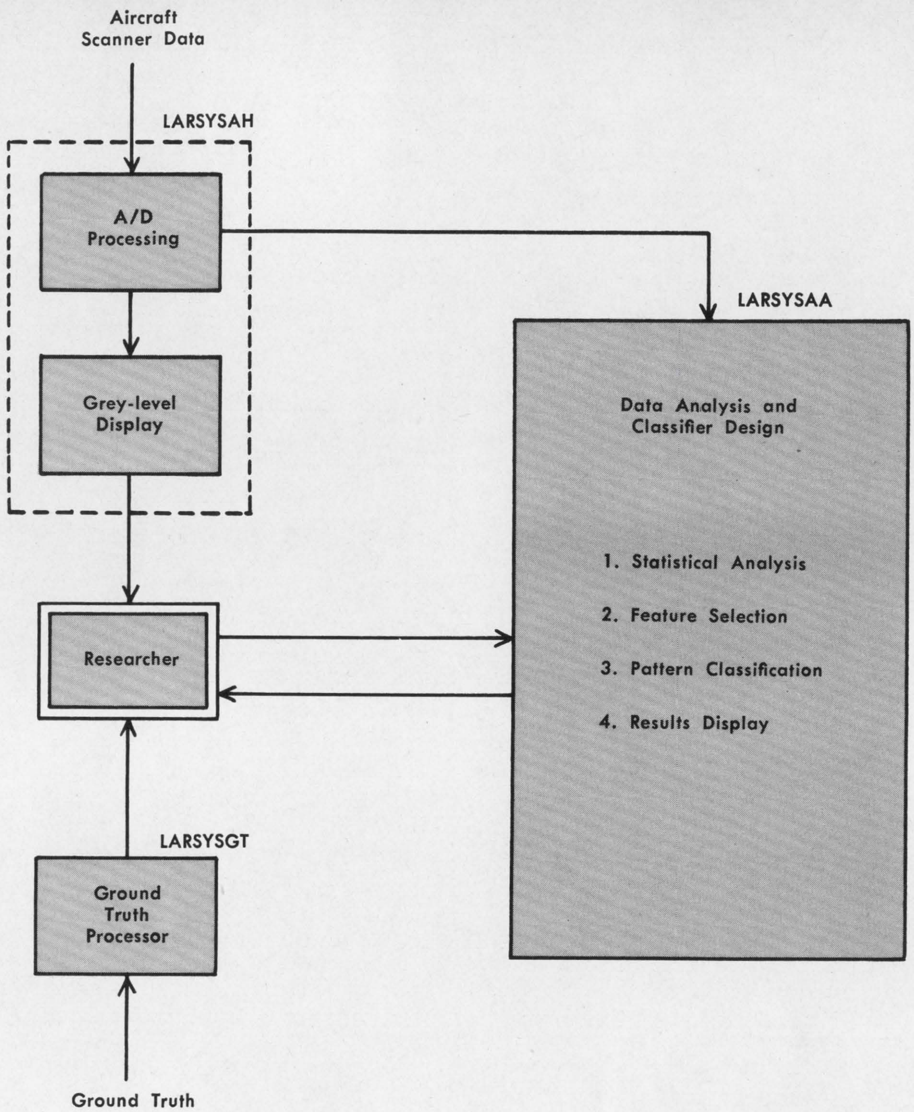


Figure 1. Laboratory for Agricultural Remote Sensing Data Processing System (LARSYS)

The other form of data utilized is "ground truth," which is collected on film and in the form of detailed written field reports. Ground truth, including such information as crop species, crop varieties, soil types, percent ground cover, etc., is cataloged and made available to the researcher in convenient form by the LARS Ground Truth Processor (LARSYSGT).

The principal concern of this article is the LARS Aircraft Data Analysis and Classifier Design System (LARSYSAA) which performs the function of pattern classifier design based on the data from the aircraft mounted multispectral scanner.

The LARS Aircraft Data Analysis and Classifier Design System (LARSYSAA)

The User–System Interface: Introduction. There are at least three important reasons why the user-system interface has received considerable attention in the development of the analysis and classifier design system.

1. Optimal classifier design requires a substantial amount of interaction between the various phases of

See the market place—page 29

the design system.* At the present state-of-the-art, this interaction is best coordinated by the researcher.

2. Remote sensing applications invariably involve huge masses of data. As a result, the quantity of data input and results output required for a classifier design task consumes a considerable amount of computer time. It is essential, therefore, that the analysis and classifier design system be largely immune from user errors (e.g., control card errors), so that errors in the later stages of processing will not result in loss of all the work which has gone before.

3. In the face of the two requirements already noted, the experimental status of the remote sensing problem makes it desirable that most or all of the processing system be written in a high level compiler language so that modifications to the system may be made quickly and easily by the researcher.

The third of these requirements has been satisfied through use of FORTRAN IV (except for a few minor utility functions which can be accomplished most efficiently through use of assembly language). The way in which the other two requirements (user control, user-error recovery) have been met is discussed in the following sections.

LARSYSAA System Monitor; Free-form Card Format. Figure 2 shows the control structure of the LARSYSAA system. The figure indicates that the system is composed of a Monitor and four distinct processing phases, each processing phase directed by its own supervisor. The multiphase structure results largely from the need to minimize the amount of core memory occupied at any one time by program instructions, in order to maximize the amount of memory available for data. In fact, for the same reason the individual processors are also decomposed into multiple phases

* In this article three phases of classifier design are distinguished: statistical analysis, in which general statistical properties of the data are measured; feature selection, in which an optimal set of features are selected for use in the recognition process; and classifier synthesis, in which the classifier is designed (or "trained") and tested using the results of the preceding phases. The precise nature of the computations carried out in each phase depends on the particular feature selection and pattern recognition algorithms used by the researcher.

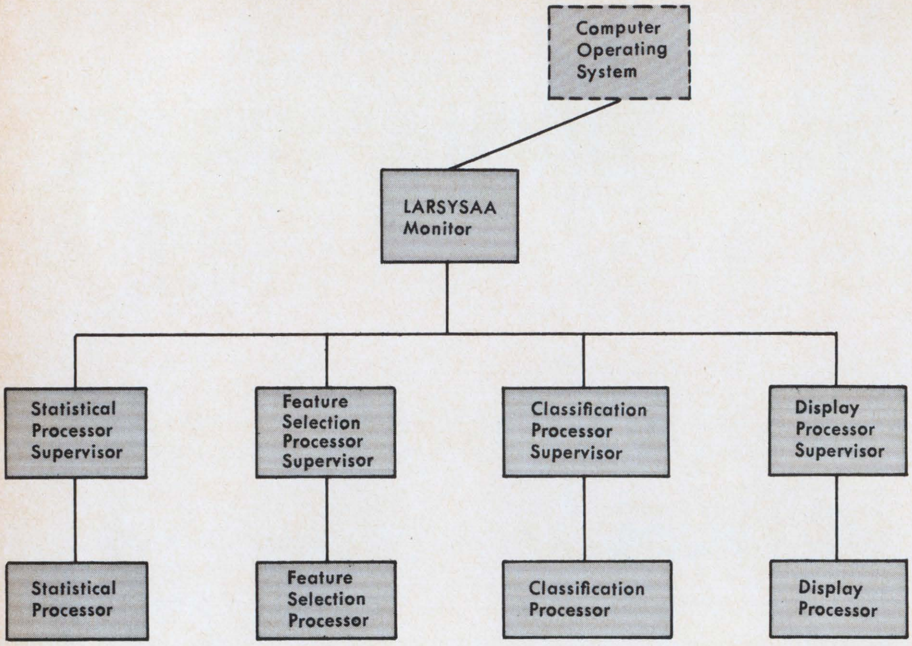


Figure 2. LARSYSAA Control Structure

which are only called into core memory by the respective supervisors as needed.

Processing under LARSYSAA commences when the computer operating system recognizes a job control card calling for the LARSYSAA system and loads the LARSYSAA Monitor into core memory from the program library.

The principal responsibility of the LARSYSAA Monitor is to recognize and interpret Monitor Control Cards which request loading of the processor phases from the program library. These control cards and all other control cards and data cards read by LARSYSAA may have an almost arbitrary format. On control cards, a special key word must be punched first, followed by any other key words associated with the control card, separated by commas. On data cards, parameters are punched in a specific order, separated by at least one blank. Should the user inadvertently punch an unrecognizable keyword or inconsistent parameter, the card in error is printed on the console typewriter along with instructions as to the action necessary to resume processing. Diagnosis and correction of errors in this manner before they can produce abnormal termination of processing prevents the loss of intermediate results stored in core memory by preceding stages of processing. The researcher, who will generally operate the computer and punch control cards as they are needed (he may elect to type them in at the console if desired), is thus freed from the burden of remembering and adhering to rigid input format requirements and may concentrate more fully on the analysis and design problem at hand. The experienced user finds himself operating in a conversational mode with the computer program; the novice finds that the key-word, free-form card input and attendant error checking speeds the process of learning to use the system effectively.

The Processor Supervisors. The responsibilities of the Processor Supervisors are threefold: interpretation of processor control cards, dynamic memory allocation, and processor control. Once all of the processor control cards have been read (and any necessary error recovery performed), the processor scans the list of operations (processing options, see Table 1) that have been requested and calculates the amount of core memory required to perform those operations. If the core memory needed exceeds that available, the supervisor reports this fact to the user who may then reenter the processor control cards with appropriate changes. In pattern recognition terms, these changes generally involve trading off the number of processing options requested against the number of pattern features and/or pattern classes.

After a suitable set of processor control cards has been read, the supervisor calculates base addresses and sizes of all variable arrays so that core memory will be efficiently utilized. The size and address information is then available to the processor subroutines as needed.

The flexibility gained through the use of dynamic memory allocation broadens significantly the range of problems that can be handled, even in the face of fairly severe core memory limitations.

Once the memory allocation procedure is complete, the analysis and design operations requested are carried out under the direction of the supervisor.

The Processors. To discuss the processors in general terms, a possible analysis and design procedure will be described.

Given a new set of digitized scanner data and the associated ground truth, the researcher's first task is to select a set of training patterns. To accomplish this, he obtains a set of grey-level printouts which aids in locating the boundaries of the agricultural fields, roads, bodies of water, etc. Data from fields of known classification are then used by the Statistical Processor to calculate various statistical quantities for the pattern classes and produce the graphical data displays listed in the first part of Table 1.

Because of uncertainties as to

Statistical Analysis Facilities

Compute mean vector and covariance matrix for each class.
Compute mean vector and covariance matrix for each field.
Punch data deck containing statistics and other pertinent information for future use with Classification Processor.
Histogram selected features for each class.
Histogram selected features for each field.
Print spectral plots for each class.
Print spectral plots for each field.
Print as many spectral plots as desired, each displaying results for up to four different classes.

Feature Selection Facilities

Determine optimal sets of 1, 2, 3, . . . features.

Classification Facilities

Perform pattern recognition using any subset of classes and features made available by the Statistical Processor.

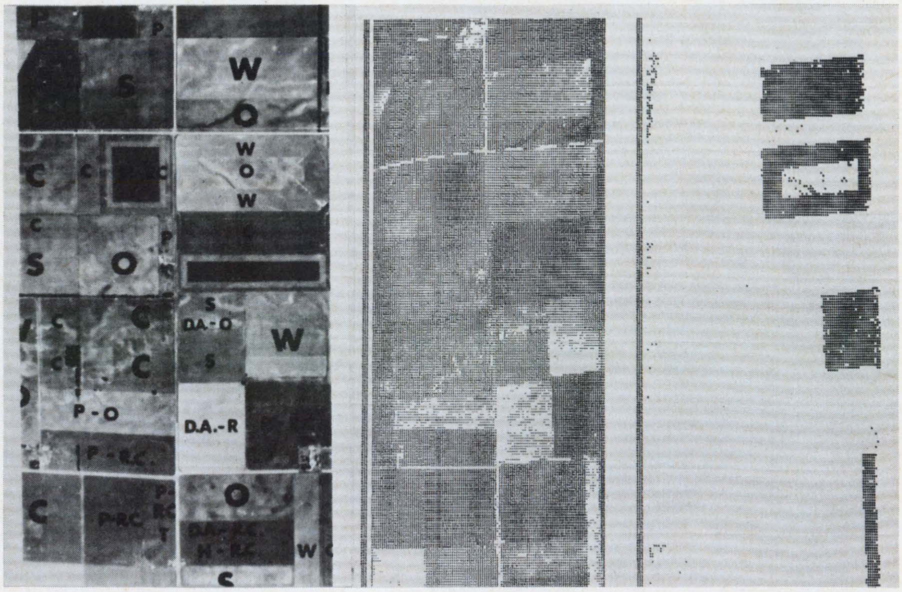
Display Facilities

Print information as to source of training data.
Outline training sets if they appear in results display map.
Print results of training operations.
Use a specified symbol set for results display map.
Compute and print classifier performance evaluation for training set
a) On per class basis.
b) On per field basis.

List areas used as test samples for performance evaluation.
Outline on results map the areas used as test samples.
Compute and print classifier performance evaluation for test set
a) On per class basis.
b) On per field basis.

Apply likelihood thresholding to establish a rejection class.
Recompute and print performance evaluations on the basis of any specified grouping of classes.

Table 1—LARSYSAA Processing Facilities



(a) (b) (c)

Figure 3. (a) Aerial photo; (b) Grey-level display; (c) Classification results.

ground truth details (such as the precise locations of field boundaries), the researcher may at this point have limited confidence in the training set he has selected. However, by using the graphical output of the Statistical Processor, he may select a set of pattern features which appear to be useful for differentiating between the classes and, temporarily bypassing the Feature Selection Processor, use the Classification Processor and Display Processor to produce a classification based on the tentative training set. It has been found that even such "crude" classification can yield a printout or map of the data which is considerably more detailed than a grey-level printout (probably because the information contained in several spectral channels is condensed into a single display). The researcher can use this result to refine the training set, perhaps then performing one or more reiterations of the same procedure to achieve additional refinement.

Once a reasonable amount of confidence in the training set has been attained, the Feature Selection Processor is brought into the processing loop. By means of a suitable algorithm (see, for example, Min, Landgrebe, and Fu[3]), the Feature Selection Processor provides information as to the best one, two, three, etc., features to be used to obtain optimal classification for the specific problem at hand. Using this information and additional passes through the Classification and Display Processors, the researcher's principal remaining task is to decide, on the basis of reasonable computation time and the desired level of classification reliability, which optimal feature set (i.e. how many features) should be used.

Figure 3 shows a grey-level printout and an aerial photograph of the agricultural terrain over which the data were taken. The classification task was to discriminate wheat (printed with W's) from various other classes (all printed as blanks).

Program Modification Facilities

As noted above, the research environment requires that the analysis and classifier design program be easily modified—for example, to test

new classification algorithms. This flexibility is achieved by a) the dynamic storage allocation approach discussed above; b) inter- and intra-program communication via common storage areas; c) residence of the source language program on a tape which is easily modified by an editing program; and d) a self-directed System Construction Program which, once initiated, performs all of the steps necessary to go from source language to operational program. Item b increases considerably the importance of the System Construction Program which has the responsibility of inserting all COMMON cards into the source language "deck" during system construction. This relieves the user of the chore of modifying the COMMON cards in every program and subroutine each time a change is made involving the common variables.

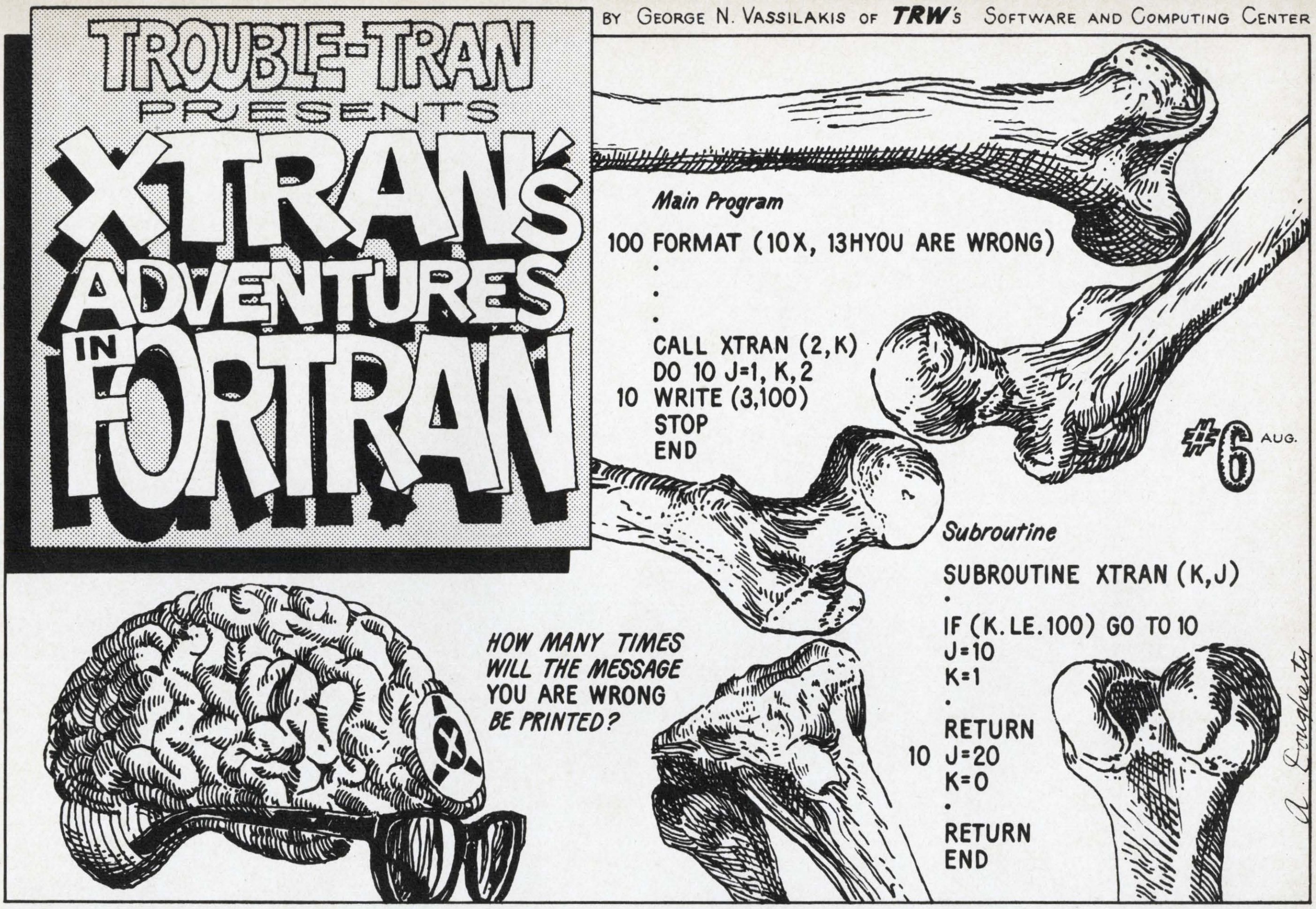
Concluding Remarks

This article has presented some aspects of a system of computer programs which, in spite of a fairly complex processing situation and demanding core memory requirements, allows the user to solve a wide range of problems without actual modification of the program. When modifications are unavoidable, the system structure is such as to allow the changes to be implemented easily with the aid of a System Construction Program. A "conversational mode" of operation which is of particular value in the research environment has been achieved through the development of techniques which opimize man-machine communication and minimize the inefficiencies which usually result from a high level of on-line user-system interaction.

REFERENCES

1. Laboratory for Agricultural Remote Sensing, *Remote Sensing in Agriculture, Volume 2,* Research Bulletin No. 832 (Annual Report), Agricultural Experiment Station, Purdue University, Lafayette, Indiana, 1967.
2. Laboratory for Agricultural Remote Sensing, *Remote Sensing in Agriculture, Volume 3,* to appear.
3. Min, P. J., Landgrebe, D. A., and Fu, K. S., "On Feature Selection in Multiclass Pattern Recognition," *Proc. Second Annual Princeton Conference on Information Sciences and Systems,* March 25–26, 1968, Princeton University, Princeton, N. J.

BY GEORGE N. VASSILAKIS OF TRW'S SOFTWARE AND COMPUTING CENTER



XTRAN'S ANSWER TO LAST MONTH'S PROBLEM

The correct answer bearing the earliest postmark will net the reader submitting it $25.00
The second correct answer with the earliest postmark wins $15.00

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

CONTEST RULES: 1. USA standard FORTRAN IV is assumed.
2. Version 12 of the IBM 7094 FORTRAN IV Compiler is used in verifying answers.

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

TROUBLE-TRAN EDITOR,
SOFTWARE AGE, 1020 CHURCH ST.,
EVANSTON, ILLINOIS 60201

WINNERS OF TROUBLE-TRAN PROBLEM 3

FIRST PLACE: R. E. Waychoff, Burroughs Corporation, 460 Sierra Villa, Pasadena, California 91109

SECOND PLACE: S. J. Metz, Applied Data Research, Inc., 2425 Wilson Blvd., Arlington, Virginia 22201

PROBLEM #5

BOTH THE MAIN PROGRAM AND THE SUBROUTINE WILL COMPILE WITHOUT ANY DIAGNOSTICS. X(J) APPEARS ON THE RIGHT SIDE OF THE EQUAL SIGN AND, SINCE THERE ARE NO DIMENSIONS SPECIFIED FOR X, IT WILL BE TREATED AS A FUNCTION. A CALL TO THE FUNCTION X WILL BE GENERATED WITH J AS THE PARAMETER.

SINCE THE SUBROUTINE IS USING X AS THE NAME OF A COMMON BLOCK, X WILL BE INSERTED IN THE LOADER DICTIONARY; AND SINCE THE LOADER DOES NOT DISTINGUISH BETWEEN THE NAME OF THE COMMON BLOCK AND THE NAME OF A ROUTINE, THE CALL TO X IN THE MAIN PROGRAM WILL TRY TO EXECUTE BLOCK X AS A SUBPROGRAM.

DEBUGGING PAST AND PRESENT

Kenneth P. Seidel

Member of the Senior Technical Staff
Informatics Inc.
Sherman Oaks, California

■ Computer literature has long shown an awareness of the need to provide programmers with debugging tools to simplify and accelerate the checkout of computer programs.

In this article, we shall examine the early software of the IBM 704 and 709, and then that of Operating System/360. The contrasts will reflect the growing sophistication of the software industry, and show the wealth and variety of languages and debugging facilities available to today's programmers.

In the early 704 days, the symbolic assembly program (SAP) was the chief software—originating at United Aircraft. This, together with a small monitor program to expedite batch processing, and stand-alone post mortem dump programs, was about all that existed in the way of software—there were no compilers, relocatable loaders, IOCS, or dynamic snapshot facilities. The monitor usually provided a post mortem dump which could be entered by the program or by an operator key-in procedure (in case of an unexpected halt, loop, or other malfunction).

During this exclusively assembly language era, then, the typical approach to debugging (perhaps after some amount of "desk-checking") could be characterized as serial—the program was loaded, and execution began. If something went wrong, the operator (or programmer, in too many instances) sooner or later realized it. Then the operator would key-in the transfer to the monitor-dump program. If the malfunctioning program had not destroyed the monitor area, the dump was taken, and the next job began. Otherwise, the operator loaded the dump deck through the card reader, dumped core in the range of memory known to contain the user's program, and then started afresh for the next "stacked" job.

The programmer, after inspecting the post mortem dump, drew conclusions as to the cause of the problem, devised corrections, prepared absolute patch cards, and resubmitted the job; probably he had eliminated one error, and was ready to encounter the next one. As time (days) passed, errors were isolated, corrected, and the program converged toward the debugged state —with additional time out for correcting the symbolic version of the source deck in parallel, and reassembly.

Then, in 1958, along came FORTRAN. Even though it was initially a stand-alone system, it possessed the great virtue of freeing programmers from the error-prone instruction-by-instruction coding in assembly language, by providing a higher level language for problem statements. The compiler generated the executable machine language, making programmers more productive.

As experience with FORTRAN became widespread, it was realized that debugging still was a major consumer of programming time. The malfunctioning FORTRAN program was still debugged by means of dumps, although more powerful dump programs (PDUMP) were becoming available. These provided dynamic capability, and permitted one to dump memory areas in various formats.

But much assembly language coding was still being done. Until the advent of the 709, there was little improvement in debugging aids for such programs.

With the advent of the 709, operating systems were seen as the place in which more powerful services should reside. The Share Operating System (SOS) represented a tremendous advance over previous monitors. The object program structure was a compactly encoded (SQUOZE) facsimile of the symbolic assembly source deck. The SOS loader was, therefore, capable of incorporating symbolic modifications (insertions, deletions, changes) into the program being prepared for execution, by means of either Alter (source line) numbers or symbolic relative addressing. Also available for insertion were macro-instructions capable of causing dynamic formatted dump printouts to be issued at selected points under specified conditions (e.g., every fourth time).

The above is only a synopsis of the early historical development of debugging software, for a particular line of computers. Certainly, other specific histories would exhibit additional techniques, while still possessing basic similarities.

Now, let us turn our attention to the languages within IBM Operating System/360. We

shall discuss the following debugging languages:

- TESTRAN, for assembly language programs
- COBOL debugging language
- FORTRAN IV debug facility
- Debugging-oriented facilities in PL/I

TESTRAN

This Operating System/360 facility is described in Control Program Services. It consists of a set of macro-instructions which are placed in the front of an assembly language program source deck. The macros assemble as separate control sections. At load (linkage editing) time, the test requests are converted to "supervisor calls" which replace the normal instructions at points where requests are specified.

This provides the facility of executing debugging statements "remotely" as though they were inserted in the normal flow of the program.

There are twenty-four basic statement forms, in five major classes.

The DUMP class enables the user to record the contents of (a) data regions, (b) data regions when a change of value occurs, (c) addresses of dynamically allocated storage, (d) registers and program status word, and (e) textual commentary.

The TRACE class enables the user to (f) record program transfers, (g) record execution of CALL statements, (h) record references to specified regions, and (i) activate and terminate trace modes.

The TEST class enables the user to designate (j) the beginning or ending of dynamic testing, (k) the points at which testing is to occur, (l) internal flags and counters, and (m) conditions for executing test requests.

The GO class permits flow within a test request; the SET class permits assigning values to internal flags or counters, or to user registers or data areas.

The TESTRAN statements, when activated, produce debugging information in a special file (data set), which is then processed in a separate "editing" run.

TESTRAN contains most of the well known debugging facilities. The disadvantages are that reassembly is usually necessary to incorporate additional test requests, or to incorporate program changes devised as the result of errors detected by the use of TESTRAN. (Also, the number and complexity of control cards and test cards needed to accomplish the total process might be considered a disadvantage.) Overall, the system satisfies the criterion that the debugging language be of the same type as that employed in the program being debugged, but the ease of use is not equal to that of the SOS modify-and-load feature.

COBOL Debugging Language

The System 360 COBOL compilers contain language extensions designed to facilitate debugging programs at the "COBOL level," without disturbing the integrity of the source deck in question. The design concepts evolved from the "debugging packet" additions made by IBM's Bob Johnsen to the 7090/7094 COBOL compiler.

Four statement forms augment the normal COBOL vocabulary, in System 360:

- READY TRACE
- RESET TRACE
- EXHIBIT
- ON

These statements may be placed anywhere. In debugging operations, however, the debug-

ging packet usually contains these (and perhaps other) statements.

The trace mode is activated by execution of a READY TRACE statement, and suspended by execution of a RESET TRACE statement. When the trace mode is in effect, arrival at each paragraph or section is indicated by a printout of its name on the standard system output device.

The EXHIBIT statement provides the option of printing specifically designated data values, identified by their names, on the standard system output device. The printout may be restricted to only such times as a change in value is detected, by using the form EXHIBIT CHANGED or EXHIBIT CHANGED NAMED.

The ON statement provides a convenient means of controlling selective debugging on a regularized basis. For example, the sentence:

```
ON 1 AND EVERY 4 READY TRACE,
   EXHIBIT NAMED
EMPLOYEE-NAME, DEPARTMENT,
   ELSE RESET TRACE.
```

restricts the program procedural trace printout to the first, fifth, ninth, etc., times through the debugging packet, and also causes the data items EMPLOYEE-NAME and DEPARTMENT to be printed on SYSOUT in self-identified form.

A debugging packet is preceded by a card of the form

*DEBUG pname [, TRY]

where *pname* is a paragraph or section name in the subject source program, qualified (if necessary), and "TRY" tells the compiler to attempt execution even if a major error appears in the packet itself. (If TRY is omitted, any significant packet error prevents execution.)

By use of the disk-resident source program library, the user need only specify the source deck name on a BASIS card, followed by the debugging packet; the combined source statements are recompiled as a unit, with the procedural body of the packet being incorporated into the object program at "pname."

Modifications may be applied to the source stream called out by the BASIS card, in the form of INSERT and/or DELETE cards placed between the BASIS card and the debugging packet. Of course, the source deck itself may be placed in the input stream, followed by one or more debugging packets.

FORTRAN Debug Facility

In October 1966, IBM issued a Technical Newsletter which defined a debug facility for the OS/360 FORTRAN. This additional capability adopts the concept of a debug packet. A FORTRAN packet begins with an AT statement, which simply designates a statement number in the program or subprogram to be debugged. The statements within the packet are performed prior to execution of the statement designated by the packet-identifying AT statement.

A program specification statement has been added to the language in order to specify various options applicable to debugging. This DEBUG statement, which must precede all packets, permits the programmer to select:

a) The unit on which debugging output is to appear.
b) Which arrays are to be tested for subscript validity, when referred to.
c) TRACE, which displays program flow by statement numbers.
d) Variables (or arrays) to be displayed whenever new values are assigned; for arrays, only changed elements are displayed.

e) SUBTRACE, so that job flow through subprograms is displayed symbolically.

Within packets, additional executable statements are provided, to turn the TRACE mode ON and OFF, and to DISPLAY variables. This latter statement is a short form of data-directed (NAMELIST) output.

In the debugging capabilities of FORTRAN and COBOL for System/360, we see a marked influence of the latter on the former. Both facilities offer convenient, simple, yet flexible tools designed to speed the program checkout process. They seem assured of ready acceptance by the user community. Also, it is evident that these same debugging language features could be adapted easily to conversational mode FORTRAN and COBOL processors.

Debugging Capabilities in PL/I

It is natural that the newest general purpose language—IBM's PL/I—contain considerable provision for programmer control of conditions affecting the proper running of a program.

In PL/I, the user has optional control over a host of conditions, which he may disable or enable dynamically at will. He may also specify the actions to be taken at the time an associated interrupt condition is "raised." Some of these controllable conditions are pure hardware-oriented (e.g., FIXEDOVERFLOW, ZERODIVIDE); others are, in today's technology, handled only by software (e.g., SUBSCRIPTRANGE, SIZE).

The conditions which are controllable by the programmer have the keyword designations:

- CONVERSION
- FIXEDOVERFLOW
- OVERFLOW
- SIZE
- UNDERFLOW
- ZERODIVIDE
- CHECK
- SUBSCRIPTRANGE

These words may be placed at the start of a program block (or statement, in some cases), preceding the block label, and delimited by a colon. This enables the related "interrupt" condition. To disable any of these conditions, the same keywords may be used, preceded by NO without intervening blanks.

The programmer may specify his own processing steps to handle any condition which arises (while enabled), by writing an ON statement. For example, the statement

```
ON SUBSCRIPTRANGE
CALL CORRECTION (I, J, K);
```

will cause the procedure "CORRECTION" to be invoked when any subscript violates its permissible range, with the variables I, J, and K being passed as arguments.

For purposes of testing programs, it is possible to force conditions to arise (and thus cause the execution of the related ON-condition) by execution of the SIGNAL statement.

Of special note is the CHECK condition, in which a list of variables is designated for detection of change within the scope of the CHECK prefix. The standard system action in this case is the output (to a printable debugging file) of the identifiers and their new values.

Conclusion

Debugging tools are now recognized to be an integral requirement of each software subsystem. Depending on the nature of the subsystem, different approaches are taken, each aimed at the special background of a hypothetical "typical user." The result, as exemplified by System/360, is that a means of improving the program checkout process is supplied to the user, on a level appropriate to his particular chosen method of coding.

PL/I in particiular is excellent in this regard, and FORTRAN and COBOL have also succeeded in adding to the formerly static standards.

In the FORTRAN, COBOL, and PL/I higher level language areas, the compiler diagnostic messages must also be considered a part of the debugging facilities. Good compilers not only detect violations of syntax, but are capable of requiring semantic meaningfulness as well, since they must deal to some extent with overall program flow and structure. Consequently, the user of these languages has a built-in advantage when he starts program debugging.

On the other hand, the assembly language user must rely on his own wits to a much greater extent. This is not unnatural, to be sure. In fact, the industry is approaching the point where the assembly language user is a relatively rare specialist. However, the debugging aids available to such a specialist are almost exclusively execution-oriented. That is, traditional assembly program philosophy is not concerned about overall program structure, or how independent lines of code are interrelated. Therefore, the error-prone and tedious characteristics of assembly languages persist to this day, and their related debugging aids do little to free the programmer from that mode of thought. It would seem that the development of a more sophisticated assembly language processor, capable of "inquiring" into the intent of sections of code, might produce a breakthrough to improve assembly language programming efficiency. ■

REFERENCES

1. IBM Data Processing Division, Form C28–6515–4, *IBM System/360 FORTRAN IV Language.*
2. IBM Data Processing Division, Technical Newsletter No. N28–2147, Amendment to *IBM System/360 FORTRAN IV Language,* Form C28–6515–4.
3. IBM Data Processing Division, Form C28–6516–5, *IBM System/360 Operating System COBOL Language.*
4. IBM Data Processing Division, Form C28–6541, *OS/360 Control Program Services.*
5. IBM Data Processing Division, *SOS-SHARE System for the IBM 709:* Distributions 1–6.

market place

The Classified Ad Department
of
SOFTWARE AGE

FREE SAMPLE

SEND FOR FREE SAMPLE KIT for marginal punched addressing label paper for EDP printout. Immediatae delivery from stock. Computer Label Co., 401 Broadway, New York, N. Y. 10013. Phone 925–3345.

HELP WANTED

PART-TIME PROGRAMMERS/ANALYSTS with scientific or mathematics degree. All work will be corresponded through the mail, on high contract rates. Prefer thorough knowledge of FORTRAN IV as applied to large scale computers. Send brief resume to Box SA-801, SOFTWARE AGE, 1020 Church St., Evanston, Ill. 60201.

SOFTWARE OFFERED

CIRCULATION & MAILING LISTS: complete tested (working) software for card or magtape conversion, maintenance and label printing on 360/20. Card version require 20/20, 8K with 2560 and 2203 with 120-c line. Tape version require 20/20, 12K with 2203 or 1403, 2560 and two tape drives. Produces 4-up E-W Cheshire labels @ 15,000/hr with 2203; up to four lines of address, with or w/o additional matchcode line, using unusual 1-cd. format providing 4 cols. (up to 8 items by using zone and digit pchs separately) of classification, demographic and/or expire data. Programs identify "suspect" duplications (i.e. near but not total duplication) for operator decision. Automatic zip markings to speed mailing operations. State selection, every Nth name and other selective label print programs also available. Write or phone H. L. Rothra, PRESS–TECH, Inc., 1020 Church St., Evanston, Ill. 60201. (312) 869–1244.

MOONLIGHT SERVICE

NEW YORK CITY: We have "moonlight" help available to do systems analysis, programming and documentation. Commercial applications on IBM 1401/60, 7070/74, 360, and Spectra 70 using Autocoder, Cobol or BAL. Call Mr. Chapman (212) YU 8–6503; or write to 321 Associates, P. O. Box 995, Wall Street Station, New York, N. Y. 10005.

market place

CLASSIFIED ADS in software age

where you can reach **91,000** programmers, mathematicians, analysts and EDP managers.

CLASSIFIED ADVERTISING

Non-Display Classified: For firms or individuals offering commercial products or services, 75¢ per word (including name and address). Minimum order $7.50. For Blind Ad Service, an additional $10.00. For "Positions Wanted" Ads, 55¢ per word (including name and address). No minimum. **Payment must accompany copy** except when ads are placed by accredited advertising agencies. Frequency discounts: 5% for 6 months; 10% for 12 months paid in advance.

Display Classified: One inch by one column, $70.00. Column width 2¼". Photographs accepted for an additional $20.00. Advertiser to supply all photo, art, cuts, or camera ready copy.

General Information: One inch display Help Wanted and Employment Services ads **will be** accepted in the classified section. Employment ads ⅞ page or more will appear run of book, will be keyed to the resume form in back of publication, and will qualify to free daily resume service.

Closing Date: 1st of preceding month (for example, May issue closes April 1st).

Send order and remittance to: Classified Dept., Software Age, 1020 Church Street, Evanston, Illinois 60201.

software age
1020 CHURCH STREET • EVANSTON, ILLINOIS • 60201

CLASSIFIED ADVERTISING ORDER FORM

Please refer to the above information for complete data concerning terms, frequency discounts, closing dates, etc. Cash with order.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35

_______ Words { @ 55¢ (Reader Rate) / @ 75¢ (Commercial Rate) } = $_______

_______ Include Photograph @ $20.00 additional (**Display Ads Only**)

☐ Run Ad Blind $10.00 Insert_______time(s) Total Enclosed $_______

NAME_______

ADDRESS_______

CITY_______ STATE_______ ZIP_______

Signature_______

WORD COUNT: Include name and address. Name of city (Des Moines) or of state (New York) counts as one word each. Zone or Zip Code numbers not counted. (Publisher reserves right to omit Zip Code if space does not permit.) Count each abbreviation, initial, single figure or group of figures or letters as a word. Symbols such as 35mm, COD, PO, AC, etc., count as one word. Hyphenated words count as two words.

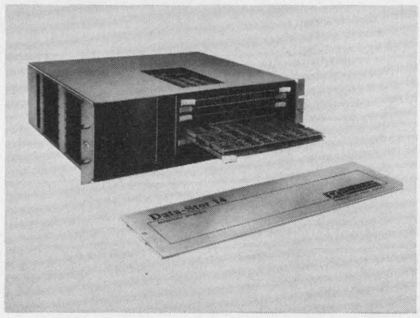
SA-8

new products

A **tape truck** 16″ wide by 24″ long (turning radius, 14¾″) will move effortlessly and transport up to 90 reels of tape in the revolutionary Data Seal, according to Data Products, Inc. This 50% increase in capacity may also be gained when using regular or slim-line canisters. File rack design affords a flat shelf, which may be utilized for other operating supplies.

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

* * *



Data–Ram Corporation's new Data-Stor 14™ Series of **general purpose core memory systems** is designed to provide high speed, economy and maximum flexibility in an extremely compact package. All units feature an internal, regulated power supply which contributes to overall system space savings.

The Data-Stor 14 Series provides capacities from 1,024 x 8 to 8,192 x 18, 1.4 usec full cycle time, 800 nsec half-cycle time and modular construction. A 2½D, 3-wire memory configuration is employed along with 100% integrated circuit logic which includes data retain circuits, optional sequential, sequential interlace and random sequential interlace.

Logic signal levels are: Logic "0"–0 volts nominal $<$ 0.45 volts; Logic "1"–5 volts nominal, $>$ 2.5 volts.

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

* * *

Numeridex Tape Systems, Inc., has added several new products to its line of **tape and peripheral tape products** for the computer and office field. They are: 1) a rigid, clear plastic container, for storing 6″, 7½″, and 8″ reels or long programs on tape to 1200 ft long; 2) a .0027″ thick Mylar/Aluminum Foil/Mylar tape in Green both sides to eliminate misreads on refractive type photo-electric readers; 3) a two color tape—one side Copper, one side Blue to prevent accidental tape reversal.

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

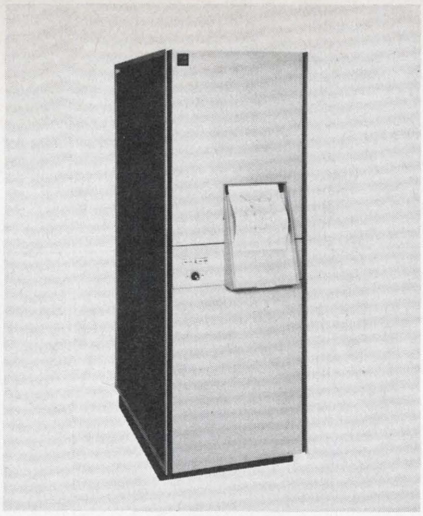
* * *

A new **decollator attachment** that is said to add more flexibility to system flow, in removing carbon from a continuous marginally punched one-side fastened form, leaving this form with two or three fastened opaques, has been released by Moore Business Forms, Inc. The new attachment called Moore Model 231 Deleaver removes one or two carbons from continuous forms, at right angles to the paper flow through the decollator, operating in conjunction with regular decollating operations.

Deleaved sets are refolded for subsequent handling, while deleaved carbons are spindle rewound for fast and clean disposal. Balance of the opaques and carbons in the continuous set strip are then decollated in the normal way.

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

* * *



Betagraphics **hard-copy CRT printer** produces 8½ x 11 inch prints of computer-generated graphics and alphanumerics in seconds, according to Beta Instrument Corp. The paper is exposed by images on the high resolution CRT and developed completely dry, no liquids being used. The Model HC610 is designed to be interfaced to a computer, coupled to a display controller or slaved to any graphical computer display.

The Betagraphics Printer is a random-access X-Y device featuring high-speed magnetic deflection, geometry correction and dynamic focus. Full Format settling time is less than 12 microseconds, point plotting rate is 500,000 points per second, line-writing rate is 2 microseconds per inch and character-writing bandwidth is greater than 1 megahertz.

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

* * *

Franklin Ribbon & Carbon Company announces its new, special treated, long wearing **teletype and computer ribbons.**

While wear is said to be extended almost 50%, the fabric is not loaded with ink in order to cause smear or smudge.

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

* * *

A complete **computer documentation** guide called DOCU-PAK has been developed by SYNERGETICS Corporation. The 200 page manual is available for a variety of configurations, including system 360 and Honeywell Series 200 computers. The manual includes specifications for documenting the implementation effort—from feasibility study through program listing development and is designed to provide the information required to effectively operate the system once installed. Documentation review and maintenance procedures are incorporated in the manual. Designed to aid in the development of business applications and the personnel who implement them, DOCU-PAK specifies when documentation should be created, what types of information should be captured and how that information should be organized. Detailed standards for system flow charts, program diagrams, decision tables, layouts and COBOL and assembly language coding are specified.

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

* * *

By using Computer Model Co. **mini-models** for 3-dimensional data processing area planning, computer layout problems can be scaled down to size. Human engineering questions are answered before the computer room is built.

Each CMC computer or tabulator model is hand-crafted from high impact, thick-wall Lucite. Individual details are shown by realistic color and by careful modeling. All models are scaled ¼" to one foot, the most common architectural standard for buliding plans. A floor plan layout sheet is also available.

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

* * *

The Indel Model HSR **computer tape rewinder** is said to have achieved precise tensioning during rewinding. It maintains tension of from 7 to 8 ounces on the tape. It rewinds a 2,400-foot reel of tape in one minute and thirty seconds.

One reel starts and tension is gained before triggering the high speed sequence to give proper tension. If a reel has broken tape, when the break is sensed, the machine will automatically apply both reel brakes, and tape will NOT spool off onto the floor, according to the manufacturer.

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

* * *

SUBSCRIBER SERVICE

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

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

My Specialty ____________

prime experience in what industry? ____________

☐ Technical Degree

☐ Non-Technical Degree

☐ No Degree

year born ________

☐ I have analog/hybrid exp.

☐ Please change my address ☐ Please enter my subscription

ATTACH LABEL HERE

FOR CHANGE OF ADDRESS

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

Old home address ____________

City, State ____________ (Zip)

Name ____________

New home address ____________

City, State ____________ (Zip code)

A dual-process, automatic **magnetic tape cleaner** that provides a "low-pressure" cleaning cycle for newer tapes and a high performance process for older magnetic tapes with higher concentrations of imbedded particles is available from Cybertronics, Inc.

The E-24, third generation tape cleaner, is designed to meet all rehabilitation requirements for today's 9 channel and full width tapes—at the highest densities. Besides having two separate and different cleaning systems for new and old tapes, the E-24 features the new "single capston" designed to eliminate tape cinching and creasing and deformation from excess tension by means of its positive, self-adjusting wind control which provides continuous tension throughout the entire reel rewind cycle.

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

* * *

Silvertex IBM #1403 **High Speed Printer Ribbon,** manufactured by Precision Computer Ribbon Co., has a unique ink formulation which renders a strong grey–black write which is said to be more uniform during the life of the ribbon than is usually found with standard black record ribbons. The special inking process also helps eliminate shadowing. This sliver-textured ribbon is available *only* from the manufacturer, who offers off-the-shelf delivery.

It is woven of Nylon 66 filament; fabric is 5 mil caliper; 281 per inch average thread count; 245 lbs. per sq. inch average Mullen burst strength; substrate meets IBM fabric specifications. It comes 14-1/16" wide; available in lengths of 20 and 25 yards.

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

* * *

Unique for the convenience of its rapid disposal system, which jets shreds into a plastic throw-away bag, the DESTROYIT Super-Speed by Michael Lith Sales Corp., is a powerful new **office paper shredder** featuring high speed destruction of large volumes of confidential material. Driven by a ⅝ H.P. electric motor the new table model uses one on-off-reverse button. Staples and clips do not harm the mechanism.

To satisfy varying needs for secrecy, speed and productive capacity, the Super-Speed is made in three models, differing only in the width of the shreds. The smallest shred width is 1/32"; the largest shred width is ¼".

All three models shred paper that is wider than the machine itself, because each is equipped with a broad, slanted feed table 14" wide at its start; this table is tapered to gather the bigger sheets.

Super-Speed has been designed with a peak mechanical operating capacity of 500 lbs. of shreds per hour. This means that in actual operation it will shred as fast as it is fed. Operation is further enhanced by disposable plastic bags which serve as a bin.

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

* * *

The **Magnetic Tape Certifier** introduced by Virginia Panel Corp. can be plugged into any tape transport and can be interchanged between transports by changing the connector. The unit will certify tape at any packing density up to 1,600 bits per in., and handle up to 9 channels on ½" tape. The certifier will accommodate any tape transports which are compatible with IBM 790, 7330 or 2400 units. It will record the number of errors with counter, and a permanent printout gives a record of dropout location. The threshold is adjustable.

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

* * *

A **tape reel identification system** using self-sticking labels that combine a color code and space for printed information on each label is available from W. H. Brady Co. This system eliminates the need for a printed label and a separate color-coded label on the same reel.

Brady self-sticking reel labels can be printed by computer–printer, typewriter, or even hand written. Printout area is rectangular with a curved, color-bar top. The 4⅝" wide pinfeed carrier liner is perfed between each label for individual use. Labels are made from a latex-impregnated paper stock. Pressure-sensitive adhesive backing lets labels be removed from reel without residue. Available in 9 colors.

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

* * *

software age

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

CONFIDENTIAL INQUIRY

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

TYPE OR PRINT CLEARLY FOR PHOTO REPRODUCTION

JOB DESIRED: ______

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

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

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

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

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

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

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

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

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

Degrees			
Years			
Schools			

EMPLOYMENT: Indicate present employment and previous jobs below.

Employer			
City			
Years	to	to	to
Title or Function			

Name ______ Age ______

Home Address ______ Home Phone ______

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

Security Clearance ______ Location Preference ______

Marital Status ______

Military Status ______

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

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

AFTER OCTOBER 10, THIS FORM WILL NOT BE PROCESSED
WRITE ADVERTISERS DIRECTLY OR WRITE US FOR CURRENT S/A ISSUE

check your interests here

Fill in the confidential inquiry form on the other side of this sheet. This form provides all the information advertisers require to screen applicants. If further information is desired, you will hear from the advertiser direct. Then, check below the boxes of those companies to which you want copies of your form sent. Mail to Software Age, 1020 Church Street, Evanston, Illinois 60201. **(Please do not send us your own resume. We will only process this form. A new form must be filled out for each issue in which you are answering ads.)**

	Page
☐ 1. Aerospace Corp.	10
☐ 2. Blind Ad SA-802	24
☐ 3. IBM Corp.	2nd Cover
☐ 4. ITT Data Services	18
☐ 5. Jet Propulsion Laboratory	17
☐ 6. Lawrence Radiation Laboratory	4
☐ 7. Leasco Systems & Research Corp.	27
8. McDonnell Douglas Corp.	15
☐ Santa Monica, Calif.	
☐ St. Louis, Mo.	
☐ 9. Montgomery Ward Data Center	11
☐ 10. National Cash Register Co.	16
☐ 11. Pratt & Whitney Aircraft	20
☐ 12. Rann & Paley, CPA's	28
☐ 13. RCA Information Systems Div.	3rd Cover
☐ 14. Sikorsky Aircraft Co.	23
☐ 15. Univac Data Processing Div.	9
☐ 16. Univac Federal Systems Div.	14
☐ 17. Vickers, Inc., Div. of Sperry Rand Corp.	8
☐ 18. Vitro Laboratories	19
☐ 19. Xerox Corp.	4th Cover

EMPLOYMENT AND SEARCH AGENCIES

	Page
☐ 20. Callahan Center for Computer Personnel	32
☐ 21. Computer Careers, Inc.	32
☐ 22. Computer Personnel, Inc.	24
☐ 23. Computer Professions, Inc.	24
☐ 24. Input, Inc.	30
☐ 25. Lawrence Personnel	30
☐ 26. Management Scientists, Inc.	4
☐ 27. MIS Inc. Management Information Services	31
☐ 28. Professional Career Centers, Inc.	28
☐ 29. RSVP Services	30
☐ 30. Source EDP	26
☐ 31. W. W. & G. Associates, Inc.	26

PRODUCTS AND SERVICES
(Use Reader Service Card)

Business Supplies Corp. of America	5
CC Systems, Inc.	27
Library of Computer & Information Sciences	3
Mathematical Sciences Group	7
Science Systems	25 & 31
Wassell Organization, Inc.	8

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

Name .... Street Address ....

City .... State .... Zip Code

Prime Experience in What Industry .... / My Specialty ....

☐ Technical Degree ☐ Non-Technical Degree ☐ No Degree

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

software age

1020 Church Street
Evanston, Illinois 60201