INVESTIGATIONS IN COMPUTER-AIDED DESIGN FOR NUMERICALLY CONTROLLED PRODUCTION

Final Technical Report

1 December 1959 - 3 May 1967

D. T. Ross

and

J. E. Ward

Electronic Systems Laboratory
Electrical Engineering Department
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge, Massachusetts 02139

Contracts

AF-33(600)-40604
AF-33(600)-42859
AF-33(657)-10954

FOREWORD

This Final Report (ESL-FR-351) submitted in May, 1968) summarizes the work performed from 1 December 1959 through 3 May 1967 under three successive United States Air Force Contracts: AF-33(600)-40604 from 1 December 1959 to 31 January 1961; AF-33(600)-42859 from 1 February 1961 to 31 May 1963; and AF-33(657)-10954 from 1 February 1963 to 3 May 1967. Details of the work have been previously recorded in a series of twelve Interim Engineering Progress Reports, plus a number of Technical Reports on specific topics.

Contract AF-33(657)-10954 with the Electronic Systems Laboratory of Massachusetts Institute of Technology, Cambridge, Massachusetts was initiated under Manufacturing Technology Division, Project 8-236, "Integration of Design Data into Numerical Control". It was accomplished under the technical direction of Mr. W. M. Webster of the Advanced Fabrication Techniques Branch, MATF, Manufacturing Technology Division, Wright-Patterson Air Force Base, Ohio.

This project has been accomplished as a part of the Air Force Manufacturing Methods Program, the primary objective of which is to develop, on a timely basis, manufacturing processes, techniques and equipment for use in economical production of USAF materials and components.

This technical report has been reviewed and is approved.

JACK R. MARSH, Chief
Advanced Fabrication Techniques Branch
Manufacturing Technology Division
Air Force Materials Laboratory

ABSTRACT

This report summarizes the activities of the M.I.T. Computer-Aided Design Project from 1 December 1959 through 3 May 1967 in the development of a generalized "system of software systems" for generating specialized problem-oriented man-machine problem-solving systems using high-level language techniques and advanced computer graphics. Known as the AED Approach (for Automated Engineering Design) the Project results are applicable not only to mechanical design, as an extension of earlier development of the APT System for numerical control, but to arbitrary scientific, engineering, management, and production system problems as well. All results have been programmed using machine-independent techniques in the Project's AED-0 Language, based on Algol-60, and are operational on several widely available computers.

Advanced techniques for verbal and graphical language and generalized problem-modeling are based on the concept of a plex which combines data, structure, and algorithmic aspects to provide complete and elegant representation of arbitrary problems. Program developments are supported by hardware and software innovations in computer graphics. Various design applications and a general technique for three-dimensional shape description complement and illustrate the general approach. The unique AED Cooperative Program allows visiting staff from industry to learn Project results while contributing to their further development.

A complete bibliography of 274 references to Project documents, talks, and thesis reports is included.

PERSONNEL

The developments covered in this report are the result of the efforts of many people over an extended period -- December 1, 1959 to May 31, 1967. Listed below are the 129 technical personnel (faculty, staff members, visiting staff, graduate and undergraduate students) who have participated directly in the work during this period, organized according to the subgroupings within the Computer-Aided Design Project.

Mr. Douglas T. Ross has served as Project Engineer for the Computer-Aided Design Project for this entire period. The support and counsel of Professor J. Francis Reintjes, Director of the Electronic Systems Laboratory, is gratefully acknowledged.

From the Computer Applications Group, Electronic Systems Laboratory
(December, 1, 1959 to May 31, 1967)

Douglas T. Ross, Group Leader and Project Engineer
Clarence G. Feldmann, Associate Leader
Dr. Jorge E. Rodriguez, Assistant Leader

Reuben J. Bigelow
Nathan S. Bromberg
Robert Bobrow
Leon M. Bousquet
Leo O. Craft
Dr. Ronald W. Cornew
David S. Evans
N. Dudley Fulton
Marcus C. Goodall
J. Martin Graetz
Irene Greif
Norman F. Hirst
Peter Johansen
Dr. Jacob Katzenelson
John C. Kotelly
Peter T. Ladd
Charles A. Lang
Robert B. Lapin
Harold D. Levin
Panos Z. Marmarelis
David F. McAvinn
Charles S. Meyer
James D. Mills
Harrison R. Morse, III
Robert C. Nelson
Barbara R. Petree
Robert B. Polansky
Eugenia Rohrberg
John Reed
Albert F. Smith
Daniel E. Thornhill
John F. Walsh
Jerry D. Welch
Thomas S. Weston
Barry L. Wolman
Jackson S. Wright
Robert B. Zara
Joel S. Zucker

Visiting Staff of the AED Cooperative Program
(March 1, 1964 to May 31, 1967. See Chapter X for additional details.)

Stephanie I. Ackley	-	System Development
Donald Barovich	-	IBM Corporation
Anton J. Berger	-	The Boeing Company
Charles W. Bower	-	The Boeing Company
Howard J. Cilke	-	Sandia Corporation
Robert K. Coe	-	United Aircraft Corporation
B. Thomas Fox	-	Sandia Corporation
Leonard H. Haines	-	IBM Corporation
Walter L. Johnson	-	Ford Motor Company
Jack H. Jones	-	McDonnell Aircraft Corporation
James R. Kennedy	-	Lockheed-Georgia Company
Richard O. Ladson	-	Univac Div. of Sperry Rand Corp.
Fabrizio Luccio	-	Olivetti
Richard S. Lynn	-	North American Aviation
James J. Martyniak	-	North American Aviation
Richard A. Meyer	-	IBM Corporation
Arthur K. Mills	-	The Dow Chemical Company
Arthur T. Nagai	-	The Boeing Company
John V. Oldfield	-	University of Edinburgh
James H. Porter	-	Chevron Research Company
Henry W. Spencer	-	Grumman Aircraft Corporation
June L. Walker	-	IBM Corporation
Irwin Wenger	-	Raytheon Company
Richard B. Wise	-	IIT Research Institute
Stephan Zurnaciyan	-	Northrup Corporation

From the Design Division of the Mechanical Engineering Department
(December 1, 1959 to May 31, 1965)

Prof. Steven A. Coons
Prof. Robert W. Mann
W. A. Albertson
David M. Auslander
Prof. Dwight M. Baumann
Richard U. Bayles
Peter T. Bennett
Prof. Charles Berg
Prof. Ernesto B. Blanco
Lawrence L. Clarke, Jr.
Norman R. Cohler
Larry M. Delfs
Charles A. Garman
Grover C. Gregory
Mark R. Haber
Wayne Hamann
MacKenzie L. Hamilton
Norman B. Heubeck
Timothy E. Johnson
Laird E. Johnston
David Korenstein
Joseph P. Ku
Harry E. Ladd
Prof. Deane Lent
Prof. Frank A. McClintock
Richard M. Merrill
Philip F. Meyfarth
Harit M. Nanavati
Prof. Henry M. Paynter
Richard P. Parmelee
George Piotrowski
Joseph D. Purvis, Jr.
Romolo E. Raffo
Ronald C. Rosenberg
Edgar H. Sibley
Robert Starzec
Ivan E. Sutherland
Coyette C. Tillman
Joseph A. Berderber
Marc R. Weinberger
Jerome I. Weiner
Abbott D. Weiss
Paul A. Wieselmann

From the Display Group, Electronic Systems Laboratory
(March 1, 1961 to May 31, 1967)

John E. Ward, Group Leader and Deputy Director, Electronic Systems Laboratory
Robert H. Stotz, Assistant Group Leader

Linda J. Bernhardt
Arthur R. Best
Abhay Bhushan
Frederick T. Blount
Michael F. Brescia
Thomas B. Cheek
Jan W. Grondstra
Dr. Uri F. Gronemann
Ednre G. Guttmann
Dr. Donald R. Haring
Richard C. Larson
Barry K. Levitt
George C. Ling
Yngvar G. Lundh
Glenn C. Randa
William D. Stratton
Eurique J. Tejara-R
Edward R. Vassar
Albert Vezza
Christopher R. Wylie

CONTENTS

CONTENTS (Contd.)

CONTENTS (Contd.)

CONTENTS (Contd.)

LIST OF FIGURES

LIST OF FIGURES (Contd.)

CHAPTER I

INTRODUCTION

A. INTRODUCTION

This report covers the activities of the MIT Computer-Aided Design Project from December 1, 1959 through May 31, 1967, under three successive Air Force contracts: AF-33(600)-40604, AF-33(600)-42859, and AF-33(657)-10954. It is a "final" report only in the most technical sense. The three Air Force contracts covering this seven and one-half year period are indeed finished, but the in-depth probing of the fabulously rich concept of computer-aided design has, as history will show, only progressed through its initial birth pangs. Computer-Aided Design -- the synergetic integration of the creative abilities of a human with the immense capabilities of computer hardware and software into a man-machine problem-solving team -- will probably never be sufficiently understood and mechanized to be considered in a "final" state. A good strong beginning has been made, however, and the MIT Computer-Aided Design Project, which has played a central role in the achievement of this viable status, is continuing its work under a new Air Force contract: F33615-67-C-1530. It is hoped that this report will convey to the reader an adequate understanding of what has been accomplished and how, as well as an appreciation of the vast further potential of continued research into this stimulating and far-reaching new technology.

In planning the organization and content of this report, a balance has had to be struck. A noisome chronology and summary of the well-intentioned but faltering steps of research would be highly inappropriate for a report of this kind. On the other hand, a straightforward listing of accomplishments in outline form would be an almost incomprehensible jumble of jargon. Many times the important topics of computer-aided design and its underlying theoretical foundations are quite foreign and far removed from topics which the nonspecialist considers important. Therefore essentially no meaningful communication can take place in outline form. A listing of statistics would

similarly be uninformative; over 100 man years of highly-skilled research effort cannot be boiled down into any meaningful nugget or distilled into any useful essence.

For the above reasons, this report is primarily a guided anthology with a philosophically-oriented motivational introduction. It is hoped that this organization of the material will serve to convey not only an impression of the activities and accomplishments of the MIT Computer-Aided Design Project over the years, but also that it will permit a deepened understanding of the far-reaching potential and significant remaining problems of the field of computer-aided design as they have been glimpsed during the conduct of this research. It is also hoped that the dynamism and excitement which permeates the collective efforts of all groups active in the field will seep through from the verbiage, for this work has relevance to all of man's activities and must ultimately be reckoned with in some fashion by all segments of society. Although the term "computer-aided design" is restrictive and in many ways inappropriate, the evolutionary activity which it is intended to connote has the broadest possible implications and must not be underestimated with respect to either its potential or the difficulties of achieving that potential.

B. CHAPTER OUTLINE

Since a correct understanding of the accomplishments of the MIT Computer-Aided Design Project depends so heavily upon a prior understanding of what is or what is not computer-aided design as we mean it, we begin in Chapter II by describing the mandate which we set for ourselves early in 1959, specifying the appropriate role which could be played by a government-sponsored research effort carried out in an academic community for the purpose of advancing industrial technology. This section establishes that the role of the Project has been to concentrate on the basic fundamentals and rigorous foundations for a generalized approach to computer-aided design, rather than to mount a frontal ad hoc attack on any particular form of computer-aided design.

We then consider, in Chapter III, what are in fact the fundamentals. What are they and why are they fundamental? How do they relate to the overall problem of generalized man-machine problem solving? The idea of a "plex" model for representing arbitrary problems forms the basis for this discussion.

The next step is to bring these fundamentals to life to create representatives of the important concepts in working form, so that the potential for accomplishing a given task is given substance. In this development, set forth in Chapter IV, it becomes clear that the primary prerequisite concerns not only the construction of working tools, but also arranging the handles and controls for those tools in such a way that a human can grasp and manipulate them. The AED language and compiler allow plex concepts to be applied to real problems.

The tremendous complexity of creating problem-solving systems is discussed in Chapter V, and it is shown that the tools themselves must be imbedded in larger automated systems in order to enable the vast amounts of detailed precision work to be accomplished with reasonable facility. It is at this point that the "system of systems for making systems", which is the primary current result of the Project, evolves.

To complete the AED story, the relationship of this generalized approach to real computer-aided design problems is outlined in Chapter VI, "Application of the AED Approach". Questions of coupling the man to the machine and of organizing the use of the system-building systems are covered to complete the picture of the current status of the practice of the art of computer-aided design as envisioned by the Project.

The lack of abundant illustrative examples of completed computer-aided design systems is cogent testimony to the fact that the field of computer-aided design has only begun. Examples are cited, but their degree of specialization and carefully-controlled conditions show that the true "reduction-to-practice" of computer-aided design concepts still lies ahead.

Chapters VII and VIII outline the activities of the Project in the design and construction of suitable display consoles for on-line man-machine graphical communication, and of the system software necessary for flexible and efficient use of these consoles in a time-sharing environment. Close cooperation with Project MAC at M. I. T. (and in many cases joint support) has played a key role in the success of these activities, which have already had a substantial influence on the characteristics of commercially available equipment.

Chapter IX summarizes results obtained from support extended to the Design Division of the MIT Department of Mechanical Engineering in the early years of the Project. In addition to several thesis investigations of possible applications of computer-aided design, members of the Design Division were active in popularization of the general concept. The most significant development of this support was the generalized parametric surface techniques for three-dimensional shape description by Professor S. A. Coons, which is briefly summarized.

The presentation finally comes full circle in Chapter X by outlining the activities of the Project in cooperative efforts and extensive communication and liaison activities with industry. The opening rounds of computer-aided design research are now over. The research will continue, but in order for computer-aided design to enter a healthy childhood, (which must precede adolescence and ultimate maturity), industry itself must become deeply involved, take some risks, reap some benefits, and contribute directly to the further advance of the field. The challenge is being accepted by industry. The combination of government sponsorship of academic research for use in the industrial society is working. The future cannot be predicted with certainty, but we may leave this "final" phase of the beginnings of computer-aided design with optimism and some considerable confidence, based upon our accomplishments to date.

Appendix I lists reported applications of AED software and the ESL Display Console by users at M. I. T. and elsewhere, other than by the Computer-Aided Design Project itself and industry participants in the Cooperative AED Program.

Appendix II is a complete listing of Project documentation (reports, technical memoranda, theses, AED literature, and technical journal papers), plus the more significant of the many lectures, talks, and technical presentations given over the reporting period. This list contains 274 citations, and abstracts are included for all formal documentation and theses. In addition to printed reports and technical papers, motion pictures have been used to convey the flavor of certain aspects of the Project work. A number of these movies are available for loan or purchase, as listed in Appendix III.

CHAPTER II

ACTIVITIES OF THE PROJECT

A. THE PROJECT MANDATE

Work on computer-aided design within the Computer Applications Group of the Electronic Systems Laboratory at M.I.T. predates the contractual periods covered by this report. The real beginning was in June of 1956 when the group inherited the MIT activity in numerical control from the preceding project, which had been primarily engineering oriented, and focused on automatic programming for numerical control through the APT Project. In the very first interim report of the APT Project, brief reference is made to the ultimate extension of the preparation of numerical control information into the design area. Throughout the APT System development, this ill-formed concept always was in the background. As the work on the development of the APT System was drawing to its logical conclusion with the cooperative effort with companies in the Aerospace Industries Association, the possibilities of extending the sophisticated use of computers into the design preparation stages which precede part programming came more sharply into focus. Since the early fifties we had made extensive use of on-line displays and manual intervention (the first graphical input through a display was done in 1954) and had developed a backlog of sophisticated computer usage techniques in various areas. Some of this experience had been used profitably in the APT System development, but other portions needed a broader context in order to be brought to fruition. These experiences, coupled with an informal acquaintanceship with the problems of aerospace design and manufacturing gained from the intimate contact with aerospace companies during the APT effort, formed the background for the initial proposal that useful results could be expected from a research program in computer-aided design.

Even our incomplete knowledge of the aerospace design-to-manufacturing cycle was sufficient to demonstrate clearly that a small academic project could not hope to create a system which

would solve everybody's problem directly. Furthermore, since public funds were involved we felt that it would not be appropriate to concentrate on a limited and specialized application which would be of direct utility to only one or a few of the potential companies we hoped to benefit. Instead, we felt that a program of research, aimed at fundamentals and coupled with close liaison with industry on a broad scale (once preliminary research had yielded usable results), would be much more appropriate. Industry, especially in aerospace matters, had historically been project oriented, with insufficient time for truly long-range technological research such as we felt was needed. On the other hand, the ability of the aerospace industry to adapt rapidly to sophisticated technological innovations is well known, the rapid incorporation of numerical control being but one of many excellent examples. We hoped, therefore, to be able to develop a systematized solution to some of the basic problems of generalized computer-aided design, and then to serve a catalytic role in stimulating industry itself in the application of the research results to diverse application areas.

Our deliberations on the role which we could play began even before we had coined the term "computer-aided design", and, in fact, a part of the early discussions centered around the distinction between "automatic" and "computer-aided" design and "computer aids to" design. (It was only later that we found occasional misunderstanding in oral presentations, when listeners thought we were talking about "computerated" design!) It seemed quite clear to us that any work in automatic design, in which a parameterized design is optimized in some fashion by parameter variation, would of necessity be too closely related to a particular application area to permit a general approach. Although various aspects of the APT System analyses and portions of earlier work in which we had used various linearlization and adaptive programming techniques were related to automatic design, we felt that such questions definitely should occupy a secondary role. Similarly, we were not interested in advancing the use of computers as aids to existing design practice. Instead, we wanted to couple a man and a machine into a problem-solving team suitable for fresh design problems requiring creative solutions, and

performing better than either man or machine alone. Our ultimate goal was to have a system capable of going from the conception of the need for a part through to the finished product by means of numerical control.

We sought to develop a system whose language and characteristics could be adapted to meet the needs of the user, whatever the domain of application. Our view was that it was impossible to put bounds upon the problems with which the system would have to cope, because design itself knows no bounds. Geometry, materials, aerodynamics, thermodynamics, and even aesthetics, all may play determining roles in a given design. If the system was truly to be applicable to creative design, it would have to be adaptable to these and many other areas. Thus in our view, there was no distinction between computer-aided design and generalized man-machine problem-solving.

These grandiose plans were not entirely wishful thinking, for the previous activities of the Computer Applications Group had touched upon many of the requisite areas and it seemed clear that even modest success would have handsome payoff if properly applied. Many of the basic ideas on which we hoped to base the new research program emerged and went through their earliest refinements in the context of our final contributions to the APT System. Still others were exercised in several small research investigations which we carried out at the close of the APT Project in preparation for the new computer-aided design focus.

Later sections of this report show that as our viewpoint has sharpened and as our techniques and general approach have matured over the years, we have changed terminology slightly, so that instead of speaking of adapting a general system to a special purpose, we now prefer to think of employing a family of generalized systems to create a specialized new member of the family. We call the family of systems by the general name AED (an acronym for Automated Engineering Design) and tout "the AED Approach" as the entire sweep of concepts, techniques, and working tools for creating specialized computer-aided design systems. The terminology and techniques are refined, but the overall concept is the same.

B. ASSOCIATIONS WITH OTHER GROUPS

Although the computer-aided design concept was not intended to be restricted to any particular type of design, the members of the Computer Applications Group had essentially no engineering design experience and would need design help from other sources. We felt that the analysis and synthesis techniques for electronics and automatic control systems were well enough developed so that when the time was ripe, design activities in these areas could be arranged with other appropriate Electronic Systems Laboratory personnel. Mechanical design, however, with its more complex domain of three-dimensional geometry and nonlinear problems of the continuum would be a much richer and more difficult area in which to perform investigations. Since a definite goal for the Project was ultimately to couple results of the design process to numerically control production techniques, it was highly appropriate that the problems of mechanical design should receive special attention. Therefore, the MIT Computer-Aided Design Project, housed in the Electronic Systems Laboratory of the Electrical Engineering Department, arranged to support a computer-aided design effort in the Design Division of the Mechanical Engineering Department at MIT.

From December 1959 through January 1966 the Design Division, first under the direction of Professor Robert W. Mann and later led by Professor Steven A. Coons, approached the computer-aided design problem from the designer's point of view while the Computer Applications Group worked from the programming and generalized problem-solving point of view. Much useful cross fertilization resulted from this joint venture, and a number of significant thesis investigations (listed in the appendix) contributed to the goals of the Project. Several of these efforts, notably the Sketchpad System, the doctoral thesis of Ivan E. Sutherland, and the Sketchpad III System, the master's thesis of Timothy E. Johnson, for two- and three-dimensional graphical manipulations, both using the TX-2 Computer at Lincoln Laboratory, the master's thesis of Mackenzie L. Hamilton and Abbott D. Weiss in preliminary ship design, the doctoral thesis of Richard I. Parmelee on three-dimensional stress analysis, and

the doctoral thesis of Coyt Tillman on a generalized differential equation-solving system, represented examples of computer-aided design systems for particular areas which were illustrative of the overall goals of the Project. Professors Coons and Mann played a significant role in disseminating and popularizing the general concept of computer-aided design, and Professor Coons has also made a significant contribution in the form of generalized parameterized techniques for representation of general three-dimensional shapes.

Another significant application system supported directly by the Project (jointly with another NASA-sponsored project in the Electronic Systems Laboratory) was the AEDNET System for the simulation of nonlinear electronic circuits, developed under the leadership of Dr. Jacob Katzenelson from 1964 through 1966. In addition to these computer-aided design application systems which received direct support, a large number of other applications of both the hardware and the software developed by the Project have been made by numerous other departments and projects at MIT, as listed in an appendix to this report.

In addition to its activities in generalized software and specific applications, the Project also has been from the beginning active in the development of hardware for the man-machine interface. Initial investigations were made using the display system of the TX-0 Computer, transferred to the Electrical Engineering Department from MIT's Lincoln Laboratory in 1958. Later, manual intervention equipment and display experiments were made with the IBM 709 Computer of the MIT Cooperative Laboratory. In 1963, with the formation of Project MAC, an interdepartmental project sponsored by the Advanced Research Projects Agency for research and development in multiple access computer or "time-sharing" techniques, the display activity was reorganized to form the Display Group of the Electronic Systems Laboratory, and hardware developments since that time have been jointly sponsored by the Computer-Aided Design Project and Project MAC. The initial activity of the Display Group concerned the completion of the ESL Display Console, the master's thesis of Robert H. Stotz, for operation in the Project MAC time-shared environment using the IBM 7094 Computer rather than the

IBM 709 Computer of the Cooperative Computer Laboratory. Later work of the Display Group is described in a subsequent chapter.

Since 1963 the MIT Computer-Aided Design Project has received very substantial support from Project MAC in the form of generous access to its powerful time-sharing facilities, and allocation of convenient laboratory and office space. In turn, all results of the Project have been made available to all users of the Project MAC facilities, (and later the duplicate facilities of the MIT Computation Center), and it is through this mechanism that the large number of other departments and projects have been able to make use of the Project results.

A final and very significant association of the Project with outside groups is the "AED Cooperative Program" which has been in operation since March, 1964. This program, which is described in more detail in a later chapter, is a unique cooperative venture with industry whose primary purpose is to promote the dissemination and appreciation of the results of the Project while at the same time contributing toward their further advance. Experienced system programmers from industry join our regular staff on a visiting basis for one year to learn about and contribute to the work of the Project. The contributions of the visiting staff members neatly balance the educational load on our permanent staff so that not only is technical progress maintained, but ideas and skills which can only be transmitted by active participation are seeded in the most direct possible way into the vital activities of industry. Perhaps more than any other aspect of the Project, the AED Cooperative Program symbolizes and makes real the unique benefits which derive from a free and open intermingling of the talents and backgrounds of the academic and industrial community with the stimulation and support of government sponsorship.

C. EFFICACY OF THE MANDATE

Over the years, people who make initial contact with the MIT Computer-Aided Design Project (and occasionally old friends of long acquaintance as well) frequently say in effect, "What on earth does what you're doing have to do with computer-aided design?" The

above outline of the multitudinous joint associations with many different groups and activities gives a partial answer. Computer-aided design as we mean it really translates into generalized man-machine problem-solving, and the above listing has only that thread in common among the various groups.

The MIT Computer-Aided Design Project would be much less effective, and its activities would be much less significant, were we to play any other role than that which we have pursued. There are many other groups interested in the same general areas, but in more of a hurry to obtain immediate results. By concentrating upon the fundamentals of the man-machine problem-solving process, we serve a catalytic role which links together in a positive fashion many disparate activities. It is significant to note that with all the groups with which the Project is associated, no coersion other than an expounding of the benefits of the AED Approach has been used to establish these associations. The efforts of the Project are a success solely on technical performance grounds.

The proper answer to the question,"What does our work have to do with design?" lies in a broadened understanding of the meaning of the word design. The remaining portions of this report elaborate upon this meaning and serve to put the final results of this opening phase of computer-aided design research into focus. There still is a great deal of additional work to be done, both in an extension and elaboration of these beginnings and in the refinement and reduction-to-practice of the total set of concepts, and these activities will be the substance of subsequent phases.

CHAPTER III

PROBLEM MODELING

There is some merit in the question concerning the relevance of our work to computer-aided design, for our consideration of fundamentals begins not with design or problem-solving or programming or even mathematics, but with philosophy (in the old-fashioned meaning of the word)--we begin by establishing a "world-view". We have repeatedly emphasized that there is no way to bound or delimit the potential areas of application of our system, and that we must be prepared to cope with any conceivable problem. Whether the system will assist in any way in the solution of a given problem is quite another matter, about which more will be said later, but in order to have a firm and uniform foundation, we must have a uniform philosophical basis upon which to approach any given problem. This "world-view" must provide a working framework and methodology in terms of which any aspect of our awareness of the world may be viewed. It must be capable of expressing the utmost in reality, giving expression to unending layers of ever-finer and more concrete detail, but at the same time abstract chimerical visions bordering on unreality must fall within the same scheme. Above all, the world-view itself must be concrete and workable, for it will form the basis for all involvement of the computer in the problem-solving process, as well as establishing a viewpoint for approaching the unknown human component of the problem-solving team.

A. THE FUNDAMENTAL PLEX CONCEPT

Although initial probings in these directions were much earlier, in the summer of 1960 we coined the word plex to serve as a generic term for these philosophical ruminations. "Plex" derives from the word plexus, "An interwoven combination of parts in a structure", (Webster dictionary). In the following paragraphs we attempt to convey the basic principles of the plex concept as it is understood at this time. As later sections will show, the full elaboration of the concept also involves almost every other aspect of the Project's activities and it

is clear that only a beginning has been made. Our understanding of the concept of plex has, however, been greatly enriched over the years as we have used it actively in our work. We hope that further use and study will enable these ideas to be understood deeply enough to permit a rigorous formulation, since we are frequently hampered by the informal nature of the current formulation.

We use the word plex both as a noun and an adjective, and in many contexts and with many nuances of meaning. All uses of the word ultimately tie back, however, to the most fundamental usage -- the modeling plex. The purpose of a modeling plex is to represent completely and in its entirety a "thing", whether it is concrete or abstract, physical or conceptual.

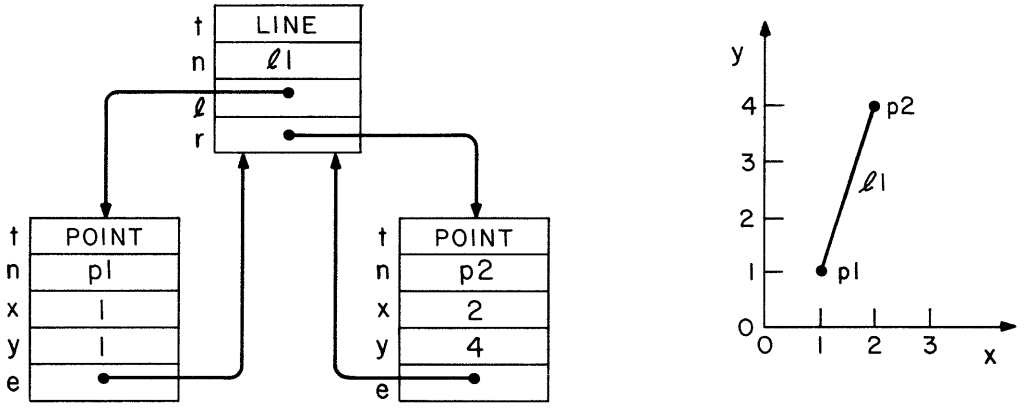
A modeling plex is a trinity with three primary aspects, all of which must be present. If any one is missing a complete representation or modeling is impossible. The three aspects of plex are data, structure, and algorithm.

In order to have a complete model or representation of a thing, we must have data. There must be some finest level of indivisible units or entities in terms of which the properties of the thing being modeled are described or measured. A datum is a value, and may be a thing itself, a token for a thing, a measurement from a measuring scale which is applicable to a thing, etc.. Data by themselves, however, are meaningless and do not constitute an adequate model.

The second aspect of plex is structure. By structure we refer to relationships among the data of a plex. No matter how simple or complex a thing which is being modeled may be, there always must be some sort of relationship between the data from which it is composed. Some of these relationships are firm and indivisible. Others are transient and easily changed or altered. But at all times some structural relationships must apply even if they degenerate to the mere association of all of the data into a set which is applicable to the model of a single thing. Even with data interrelated in a data structure, however, a model is not complete, for the meaning of the data and the interpretation of the relationships are ambiguous and unspecified.

The third aspect of plex, which in combination with data and structure makes the representation complete and workable, is algorithm. The algorithmic aspect is the capstone which allows the data in the structure to be interpreted, manipulated, and made meaningful. It is concerned with the behavioral characteristics of the plex model -- the interpretive rules for making meaningful the data and structural aspects of the plex, for assembling specific instances of the plex, and for interrelating the plex with other plexes and operators on plexes. Specification of the algorithmic aspect removes the ambiguity of meaning and interpretation of the data structure and provides a complete representation of the thing being modeled.

Figure 1 shows a trivial example of a modeling plex. In this case, we are modeling a geometric straight line in a cartesian



$$\text{left.end.point(line)} \equiv \ell(\text{line})$$

$$\text{length(line)} \equiv \sqrt{(x(r(\text{line})) - x(\ell(\text{line})))^2 + (y(r(\text{line})) - y(\ell(\text{line})))^2}$$

etc. for additional properties

Fig. 1 Plex Model

coordinate system, with labels. The data of the model are the words such as "POINT", the labels such as "p1", the numbers measuring coordinate values, and the pointer values. The structural aspect of the plex is indicated by the "n-component elements", i.e., the constituent building-block elements with variable numbers of components containing attribute values. Note that the components

are assigned mnemonic names, t, n, x, y, l, r, and e for type, name, x coordinate, y coordinate, left, right, and end of, respectively. The components "contain" the data values, and the total collection constitutes the data structure for the modeling plex. The algorithmic aspect of the model is only incompletely indicated in Fig. 1 by formulations expressed in an appropriate language showing that the left end point of a line is the value of the ℓ-component of that line; the length of a line may be obtained by the evaluation of an appropriate formula with inputs obtained from the appropriate x and y components of the point elements, etc. Notice in particular that it is these algorithmic formulations which provide the proper interpretation of the data structure, for a different set of formulations could give an entirely different meaning. For example, although the mnemonics of Fig. 1 would not be as appropriate, the same data structure could represent the top two books on the best seller list, with titles p1 and p2, and with the x and y components showing their position on the list of the previous week. Only when the interpretation rules are included is a plex model complete.

B. IDEAL PLEXES

For the sake of discussion, we sometimes take an extreme posture and consider all plexes to be degenerate in their data and structural aspects, so that all consideration may be focused entirely in the algorithmic domain. This is a generally useful viewpoint to take when one is primarily considering an ideal plex (an idealized plex) in a way which is as mechanization-free as possible. In this canonical, ideal form, all relevant attributes of a plex are represented by read-and-store procedure pairs, i.e., we ignore the mechanization of data values and degenerate the data structure with the assumption that there is associated with each read-store procedure pair a suitable "box" containing the unspecified datum. Then the entire consideration of the plex can take place in terms of reading and storing values of attributes by calling the appropriate read or store procedures. The mechanization of the "box" and its contents are of no interest.

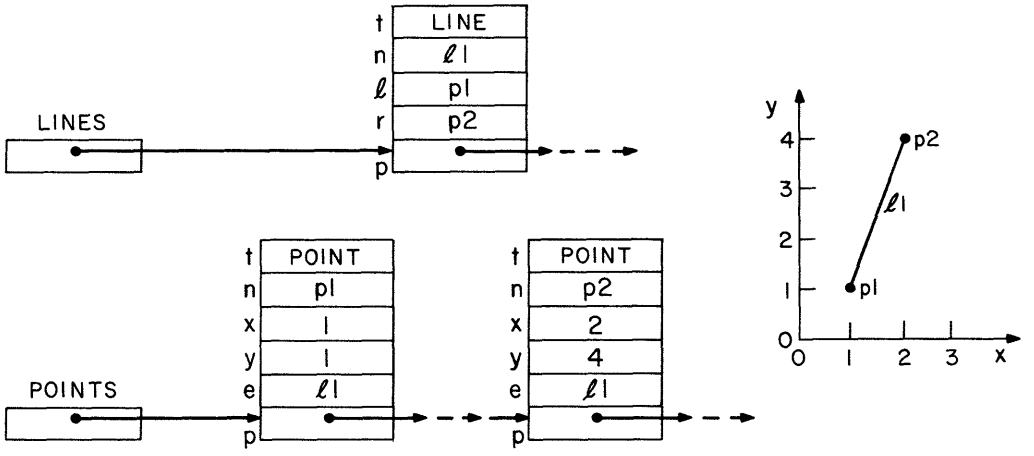
We refer to the read-store procedure pair as an ideal component. Ideal elements are then merely indivisible collections of ideal components. An ideal element is itself a suitable datum which may be the value of one or more ideal components in other elements. In addition to the read-store procedures of the component definitions, an ideal plex definition includes the definition of the ideal elements from which the plex is composed, including the specification of the element types which may be taken as values in the various ideal components, and a create-destroy procedure pair for creating and destroying specific instances of each element type. A specific instance of a plex then is some arrangement of ideal elements which is grammatical with respect to the type constraints and which allows the read-store procedures to function properly. An ideal element consisting of a single ideal component containing some unspecified datum is considered to be that datum itself. Thus the atomic levels of data and structure coincide; the atomic data are considered to be elements of specific types and thus are made to fit into the structural aspect of the plex definition as described by the component and element procedure pairs.

All plex operations may be expressed in terms of the procedures of the ideal plex definition. A plex definition defines not a single plex, but a general type of plex, of which there may be many instances. For example, the Peano axioms define the positive integers and their properties, but there are uncountably many integers which satisfy the definition and therefore are specific instances of the generic integer. Similarly, in Euclidean geometry, the definition of a triangle is generic and applies to uncountably many instances of triangle. In general, when a plex definition is applied, the algorithmic and behavioral aspects remain unchanged, as many different combinations of data and structure are constructed to yield specific instances of the plex. The establishment of specific data structure instances which are compatible with the plex definition is another major facet of the over-all plex concept, considered later.

C. PLEX MECHANIZATION

A plex is a model of a thing and therefore is a thing itself. As such, it must be given some physical form, i.e., the data, structure, and algorithm parts must be represented in some way. There are many different forms that a given plex may take, each form being merely a different way of representing all of the relevant aspects of the thing being modeled by the plex. We speak of these many different forms as various mechanizations of the same ideal plex. Just as there may be many instances of integers which satisfy the Peano axioms, there are many ways to physically mechanize integers themselves. Formulas for integer calculations may be written in terms of the ideal arithmetic operations for add, subtract, etc., and they are valid for any suitable mechanization. It is impossible to have a completely "mechanization-free" ideal plex in any form which has no mechanization at all. We must always have some mechanization. The idea of mechanization-free really involves the transformation from one mechanization to another "without loosing anything". In general, any such transformation affects all parts of the data, structure, and algorithm, since otherwise equivalence is lost.

Depending upon the use to be made of a plex model, one mechanization or another may be most appropriate. For example, Fig. 2 shows an alternate formulation for the modeling of a two-dimensional line. The plexes of Fig. 1 and Fig. 2 are different mechanizations of the same ideal plex. Notice that in Fig. 2 the lines and points are grouped separately, and the l, r and e components contain label values rather than pointer values. At the same time, the formulas for obtaining the left endpoint, right endpoint, and length have also been transformed, so that the total modeling characteristics are preserved. In order to find the left endpoint of a line in the second scheme, it is necessary to scan the entire list of points until the appropriate label is found. This is a time-consuming operation, but the physical locations of the point elements may be changed without modifying the values of the line element. In the scheme of Fig. 1, however, it is very efficient to locate a left endpoint, but if a point element is moved, the pointer value in every line element for which it is an endpoint must be updated.




```
left.end.point(line) ≡
    begin   z = POINTS;
    loop:   if n(z) is not equal to ℓ(line)  then begin z = p(z);
            go to loop end   else answer is z end;

    end;
```

i.e., search the point string for the named point.

etc. for additional properties

Fig. 2 Plex Model, Version 2

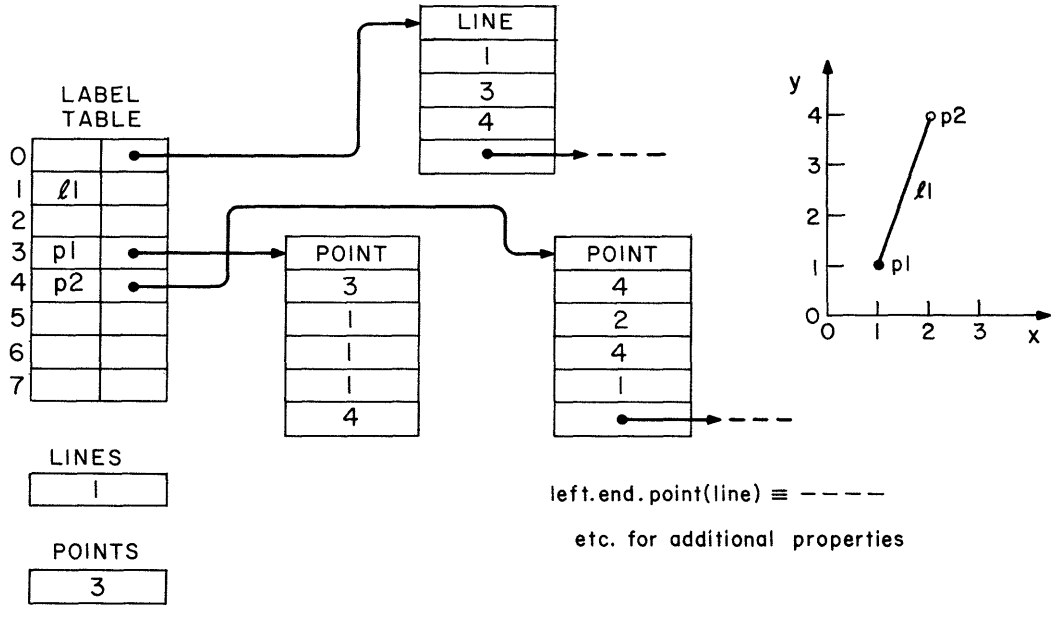


Fig. 3 Plex Model, Version 3

Figure 3 shows a third alternate mechanization in which the best characteristics of both schemes 1 and 2 are combined. In this case the pointer values are converted into an index into an array of actual pointers. In this scheme, only moderate inefficiency is introduced by indexing into the array to obtain a pointer value, while at the same time locations of point elements may be changed and only the single array entry need be updated no matter how many line elements may share the same index value.

Although no modeling plex can be entirely mechanization-free, some versions are more ideal than others. The design of an appropriate mechanization has much more importance than merely saving a microsecond or so, or a storage bit here and there. The more idealized the expression of a given modeling plex may be, the more permanence and long-term value it will enjoy. In terms of the example, the distinction between the "left-endpoint property" of a line and the "ℓ component" of an element representing a line is very important. The concept of a left endpoint is at a much higher and more stable level than is the ℓ component. Useful operations on lines expressed in terms of left endpoint properties and similar inherent characteristics of lines apply correctly to all three mechanizations, whereas equivalent formulations expressed directly in terms of ℓ components must be different for each mechanization. A single mapping of left endpoint into ℓ component can make any number of expressions apply to a new mechanization.

D. FACTORED PLEXES

The desirable and undesirable characteristics of the above three mechanizations have all been concerned with the trade-off between storage and execution time for a single plex type. Another important influence which affects not only mechanization but the ideal definition as well, concerns the use which is to be made of a plex model. A definition which is optimized for the use of a plex in isolation for only a single purpose may require considerable alteration if that plex is to be used in conjunction with some other plex for a new purpose. The reverse process also frequently occurs, in that a single definition plex is constructed for some rather gross

concept, when it is discovered that a more adequate model would result if that single plex were factored into more specialized sub-plexes. We have, in fact, already made use of this idea in the manipulations of Figs. 1, 2, and 3. The data, structure, and algorithm aspects concerned with the type, name, x-coordinate, and y-coordinate components were left unchanged in all of the transformations considered there. Thus, in effect, we had factored out those aspects as a separate sub-plex which was left invariant as we modified the mechanization of the remaining portions of the total plex. In general, any collection of concepts which is sufficiently rich to merit serious consideration in terms of plex modeling will be complicated enough that the best mechanization cannot be seen at once. An elaborate sequence of transformations in both directions, combining smaller units into larger concepts and factoring others into different modules, will take place as the many choices of possible mechanization are considered.

E. OPERATORS

From the plex viewpoint all computation, data processing, problem formulation, reformulation, and even cogitation are considered in terms of the interaction or transformation of plexes. Since it is the algorithmic aspect of plex which determines the interpretation of the data and structure aspects and specifies the behavior of the model so that it properly reflects the behavior of the thing being modeled, consideration of the interaction and transformation of plexes is primarily concerned with further elaboration of the algorithmic aspect of plex. Just as we may take two numbers and "operate" on them with an addition operator to yield a new number, or operate on a single number with a negation operator to transform it into a different number, we wish to consider operators applied to plexes to yield other plexes. There is a whole spectrum of kinds of operations, and we first consider two extreme cases, one very specific and one general, after which our main discussion centers on the techniques for the middle portion of the spectrum.

The simplest and most direct way to transform a plex is to alter one or more of its data values. This is accomplished merely

by calling a store procedure with an appropriate new value. Similarly, two plexes may be "joined" in the most trivial sense by an operator which makes them share a component value in common. All operations on specific plexes ultimately are composed of atomic operations of this form.

The other extreme of the scale of operators, which was briefly alluded to above, is indicated in Fig. 4. In this case, operations are

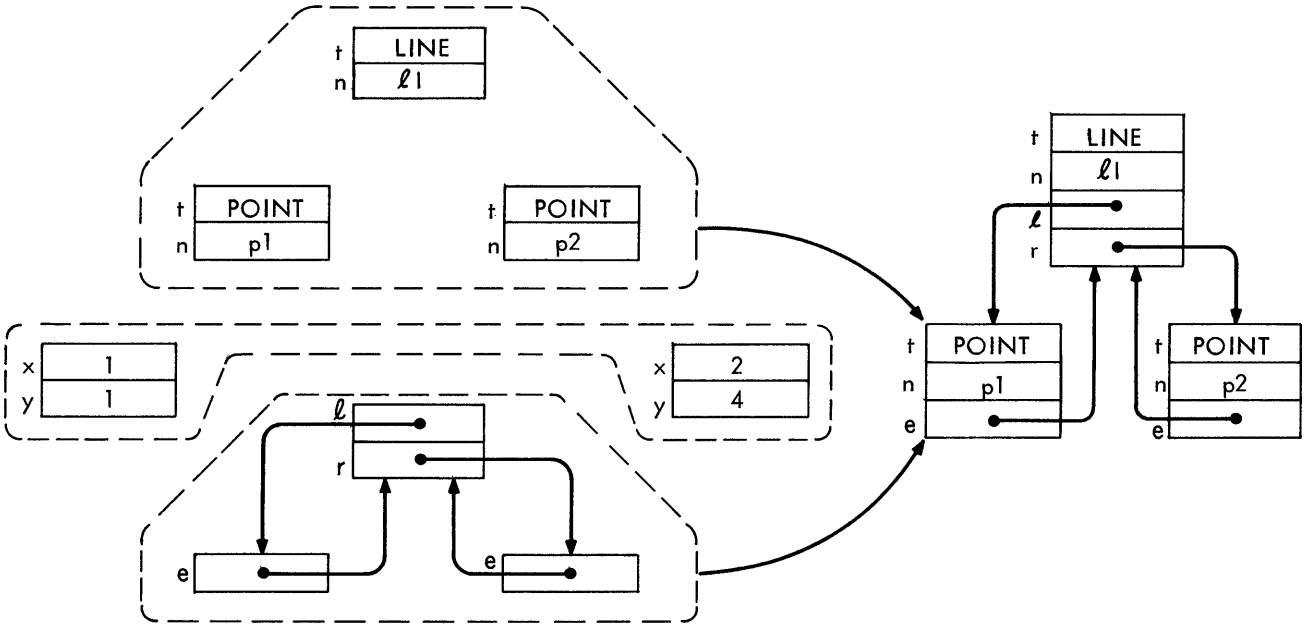


Fig. 4 Plex Factors

concerned not with specific plexes, but with manipulations of plex definitions themselves. Figure 4 indicates symbolically how the modeling plex for the straight line in cartesian coordinates may be considered to be the composition of separate plex definitions -- one for the type and naming conventions, one for the end-point relationships, and another for coordinate values for points. These subplexes may be combined in various ways to give other useful plexes. In particular, if the coordinate information is omitted, a topological, non-geometric modeling results. Here too, however, various constraints and grammatical rules must be in effect to ensure that we do not attempt to combine the elementary bead definitions in such a way that a line has coordinates or a point has endpoints. Notice,

however, that if suitable modifications are made to the algorithmic aspects of the elementary plexes of Fig. 4 (which are not explicitly shown, but are assumed to be of the same sort as we have previously considered), then such alternate compositions would be both acceptable and useful. The same form of idealized data structure could be used with different algorithm parts to stand for "line-passing-through relationship" for points as well as "end-point relationship" for lines.

Most plex operators are neither as specific as the value-changing type, nor as abstract and global as the plex definition-changing type. Usually pre-defined plex types are left unchanged and an operator says, "Combine a plex of type A and a plex of type B to give a plex of type C", without explicitly specifying the finest level of detail at which the operation is to take place (see Fig. 5). We will

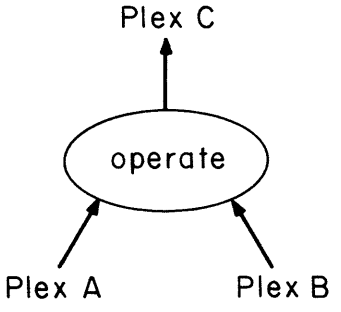


Fig. 5 Plex Operator

assume that ideal definitions have been given for the input and output plexes, and outline the general scheme whereby plexes and operators may be considered disjointly, and yet may be brought together in useful combinations. The idea is quite similar to the fact that the Peano axioms are not affected by the definition of various functions which map integers into integers, and similarly integer-mapping functions may be defined independently of the representation of the integers themselves.

F. MOUSE ALGORITHMS

Operators of the sort we are discussing do not work on plex definitions, but on specific instances of plexes. They may work on

the entire data structure of a plex or any sub-part, accessing the data using the read procedures of the component definitions. A specific data structure may be arbitrarily complex so that we must elaborate further the general plex concept in order to allow operators to be defined in a uniform way which will apply automatically to all levels of complexity. This elaboration gives a further subdivision to the algorithm aspect of plex, for it specifies how the plex interacts with operator definitions. We refer to this new feature as the mouse algorithm. In addition to element and component definitions, including their mechanization, a plex definition also must include a mouse in order to couple with operator definitions.

The general idea of mice and operators as a solution to the complexity problem is quite simple. In order to handle arbitrary levels of complexity the over-all operation is broken into a number of constituent actions which are applied repetitively as often as necessary to get the job done. An active operator working on a plex is composed of two parts: 1) a mouse algorithm which comes from the plex definition, and 2) an action function which comes from the operator definition. The mouse knows how to take a step from any place in the data structure of a specific plex to another place in that data structure, so that repetitive execution of the mouse algorithm sequences through the structure in some fashion. We call the sequence the mouse path, and the similarity between the action of this algorithm as it steps through the data structure and that of a mouse attempting to solve a maze is the source of the quaint terminology. We may think of the mouse carrying the action function of the operator. At each step along the path the action function is executed, after which the mouse takes the next step. In general, the mouse path will traverse the entire plex structure so that every aspect of the plex or any sub-part may be acted upon no matter how large or elaborate the structure may be. At all times, however, the action function is concerned only with local information.

For a given data structure, there are a great number of possible mouse paths, only a fraction of which can serve as a useful basis for operators. A useful mouse algorithm is one which allows the action function for an operator to be defined only in terms of the

simplest necessary structure, and the mouse automatically extends that definition to apply to the entire plex no matter how complicated it may be. This is the standard mathematical concept of a mapping on a sub-space inducing a mapping of the whole space, and is best illustrated by an example. Figure 6 shows an algebraic equation in

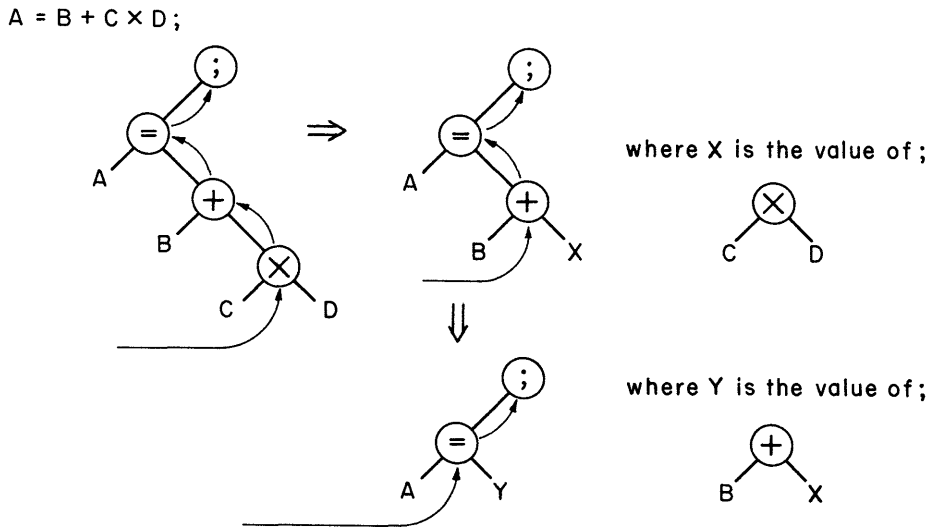


Fig. 6 Operator "Induced" from Atomic Operator by Action of a Mouse

"first-pass structure" form, (about which more will be said later). For each operator symbol the binary tree shows the expressions which constitute the left and right contexts whose values are to be combined. The chain of "precedence string" pointers shows a mouse path. Assuming that each atomic symbol is given a value, we may then define the action function for an evaluation operator solely in terms of operations on atomic values. As Fig. 6 shows, after each step of the mouse-and-action-function combination, the evaluation up to that stage provides an atomic value as the result of the calculation up to that point, and this atomic value may be used as input to the action function for the next step. No matter how extensive the algebraic expression may be, the appropriate mouse path yields precisely the right sequence, so that the atomic action is properly carried over to the entire expression. Different operators may call for different mouse algorithms, but in all cases it is this type of behavior that neatly resolves the complexity problem.

G. GROWTH ALGORITHMS

One very important class of operators is so universally required that it forms a final elaboration of the total plex concept. These operators are concerned with the construction of valid instances of plex structures which match a given plex definition. We refer to this aspect of the algorithm part of the plex concept as the growth algorithm of the plex. The growth algorithm is, in effect, inverse to the mouse algorithm in the sense that it knows how to construct a complex data structure step-by-step. Just as a good mouse allows the action functions of operators to be concerned only with atomic local contexts, a good growth algorithm works with atomic local actions, however large the total plex structure may be.

In terms of Fig. 5 we may consider the relationship between the mouse and growth algorithms, which are integral parts of plex definitions, and action functions which define operators independent of plex mechanization. Figure 5 represents

$$C = OPR(A, B)$$

where A, B, C are instances of plexes, and OPR is some plex-mapping function. OPR obtains input values from the A and B plexes via the read procedures and mice of their plex definitions. After manipulating these values in some way, perhaps using internally stored parameters and state variables, some output values are generated which are stored in plex C via the growth algorithm and store procedures of its plex definition.

Just as there may be several alternate mouse algorithms for different types of operations on a single plex type, there also may be several growth algorithms for various purposes. A given final plex structure may be grown in different ways, and the objective is, of course, to match the sequencing of the mice which are gathering input values to the proper growth sequence. All of these remarks apply at the ideal level, but take on added significance when particular mechanizations are taken into account.

H. SUMMARY

We have by now come quite a long way in the elaboration of the plex concept. A plex consists of data, structure, and algorithm parts which may be mechanized in many different ways. As long as changes in the data structure are reflected in compensatory changes in the algorithmic portion and vice versa, many different mechanizations may represent the same ideal plex.

Complicated plexes may be considered to be composed of sub-plexes. In particular, lower-level mechanizations may be considered as sub-plexes which elaborate and make more concrete the abstract ideal plex.

A plex definition actually defines a generic type of plex. There may be many instances of specific plexes which are of a given type, i.e., which satisfy that plex definition. These specific plexes have the same algorithmic portion in general, but differ in their specific data structures. An ideal plex definition is one in which the data and structural aspects are degenerated so that the definition may be given entirely in algorithmic terms. A specific instance of the ideal plex is composed of ideal elements which are individual collections of ideal components containing ideal elements as values. Data may be of any sort and a datum is assigned an element type by defining it a priori to be the value of an element with a single component. Thus data may be stored in components of elements and elements may be stored in components of elements, the resultant assembly constituting the data structure of the plex.

In an ideal plex definition, components, elements, and plexes are defined by in-out procedure pairs as follows:

item	in	out
component	store procedure	read procedure
element	create procedure	destroy procedure
plex	growth algorithm	mouse algorithm

All operations on plexes can be described in terms of these procedures. Making the data and structure aspects of the plex non-degenerate by selecting various implementations of the procedures

yields various equivalent mechanizations of the plex. Finally, various mouse and growth algorithms may be used with a single set of component and element definitions (including their mechanizations) to match the requirements of various plex-mapping operators.

With the above summary, this outline of the fundamental aspects of the plex concept is complete. There are other more detailed aspects which have not been discussed, such as constrained components, in which the value of one component depends upon values in other components, (in ideal terms this is merely an extra side effect of the store procedure for the affected components), and innumerable questions of mechanization. Some of these topics are considered in the subsequent chapters of this report, but others must await future expositions. As we said in the introduction,"The next step is to bring these fundamentals to life to create representatives of the important concepts in working form." This important job is performed by the AED language and compiling system which we take up next.

CHAPTER IV

THE AED-0 LANGUAGE AND COMPILER SYSTEM

A. HISTORY

The cohesive view of the overall plex concept presented in the previous chapter did not spring full-blown as an easy generalization of the initial idea of plex. Although all of the essential features have been present at every stage of the evolution and refinement of the ideas, (for if any essential portion is omitted the overall concept is unworkable) the specific formulations and ways of talking about the ideas have changed considerably as the work has progressed. The concept of plex is philosophical, but it has always been a working philosophy upon which functioning problem-solving systems can be based directly. The transformation from the philosophical to the working domain is effected by writing programming systems which mirror the various aspects of the plex concept and thereby use the philosophical basis as a rigorous foundation. Such a transformation takes place in terms of a programming language and programming methodology which give physical form to the philosophical abstractions.

The first direct use of plex concept was in the design and programming of the "MIT ARELEM", the Arithmetic Element Program which was the last major activity of the Computer Applications Group in the development of the APT System. This work was carried out in 1959 and 1960 as part of the transition of the APT System from MIT leadership to industry responsibility. The Arithmetic Element Program is that portion of the APT System wherein a sequence of "cut vectors" is calculated such that the cutting tool will sweep out a path in three-dimensional space which approximates a specified geometric shape within specified tolerance. The new MIT ARELEM for the first time allowed consideration of completely arbitrary cutter shapes whose profiles were approximated by arbitrary concatenations of straight-line and circular-arc segments, and also permitted the use of any number of arbitrary geometric surfaces in the

determination of the appropriate tool motion. The use of plex techniques was mandatory and natural in order to represent the multitude of properties of the various geometric portions of the problem and to control the elaborate sequencing of actions in an efficient fashion. The programming language used for the ARELEM programming was basic machine code represented by the FAP assembly language of the IBM 709-90 Computer. By sophisticated trickery, the "middle-of-cut" portion of the new analysis was re-expressed in facilities of the Fortran language for the "APT III" system which has been in use since that time, but the "end-of-cut" analysis was omitted from the Fortran version, since it would have necessitated large changes which did not fit the desired time schedule. In any case, the initial baptism of plex concepts and techniques took place in a very strenuous and rigorous environment with basic machine language as the expressive medium.

As the emphasis of the Computer Applications Group shifted from the APT Project to the Computer-Aided Design Project, the medium for carrying out plex programming also changed. In view of the Project mandate described in Chapter II, in which we hoped to provide generally useful basic techniques which would be applicable to many different problems using many different types of equipment, our attention turned early to the steps which would have to be taken within the Project in order to achieve machine-and problem-independence. Our experience with FAP and FORTRAN as expressive media for our ideas, coupled with various experiments which we had carried out using the early LISP language of McCarthy and the MIT Artificial Intelligence Project, as well as a short lived but deep investigation into true list processing* in an undocumented exercise named "META META", had convinced us that no existing language (nor any language then being proposed) would be able to serve as an adequate expressive medium for the rigors and generality of plex programming. It was at this point that it became clear that a major effort of the Project would have to lie in the area of general-purpose programming language and compiler system design.

---

*Simultaneous parallel processing of lists of values.

B. THE "BOOTSTRAP" COMPILER SYSTEM

Since we had no prior experience in compiler construction, all of our previous efforts in language design having been problem-oriented, we set about making an initial system which would serve both as a training ground for our own education and which would provide a basic initial system for programming problems in the way we felt they should be programmed. The resultant "Bootstrap Compiler System" is described in the sequence of Interim Reports of the period. Although over-ambition distorted the Bootstrap Compiler in some respects, so that the resultant system was much more powerful than we had originally intended and thereby lost its machine independence and bootstrap ability, we did in fact achieve most of our goals. The Bootstrap language, a sample of which is shown in Fig. 7, was a cross

```
•PROGRAM TO SUBSTITUTE BOX TABLE ENTRIES FOR BOX TRANSFERS
• CALLING SEQUENCE  -  'RID'(A,B,C,D)
•     WHERE      A  IS THE INITIAL BOX TRANSFER TO BE RID
•                   (NOTE BOX ZERO WILL NEVER BE RID)
•                B  IS THE FINAL BOX TRANSFER TO BE RID
•                C  IS OPTIONAL AND THE STARTING ADDRESS OF THE RID
•                  (NOTE IF C IS NOT GIVEN RID STARTS AT REGISTER TWO
•                D  IS OPTIONAL AND THE FINAL ADDRESS OF THE RID
•                  (NOTE IF D IS NOT GIVEN RID ENDS AT THE CAI)
•       FOR EXAMPLE  'RID'(440,444,EDT)  RIDS BOX TRANSFERS 440 THRU 444
•                BETWEEN THE ADDRESS OF EDT AND THE CAI

•CONSTANTS FOR THE PROGRAM
$•EDB,$,         /0              •TO ADD TO AGO 1 AND 2 TO MAKE BOX TRANSFER

•VARIABLES FOR THE PROGRAM
$•AAD,           0               •CURRENT REGISTER RID IS TESTING
$•EDT,           0               •FINAL ADDRESS OF THE RID
                 0               •SAVED CAI

•ENTRY TO THE PROGRAM
$•RID,$,         $-I=EDT=EDT(1)
                 ONE$=AGO(1)=AGO(1)+ZER
                 AGO(2)-AGO(1)=AGO(1)
                 THR$=QCT/441/440
                 AGO(4)=EDT
        440'NFM'AGO(3)
•TEST THE REGISTER FOR A BOX TRANSFER
        441'NFM'$+I=AAD
                 EDB+AGO(2)$=0(I)-AGO(1)/442/443
        442'NFM'$=0(I)/443+ZER
•SUBSTITUTE THE BOX TABLE ENTRY FOR THE BOX TRANSFER
                 0(I)$+J
                 0(J)=0(I)
•INDEX TO NEXT REGISTER AND TEST THE END
        443'NFM'AAD+ONE$=EDT/444/441/441
•EXIT FROM THE ROUTINE
        444'NFM'EDT(1)$+I/43
•     END  OF THE PROGRAM - 4/7/62
```

Fig. 7 Sample Program Written in Bootstrap Plateau Language

between an assembly language and a compiler language, and possessed a number of unique features, the most striking of which was the ability to physically incorporate a program just compiled directly into the compiler as the last step of the compilation process, so that following portions of the program being translated could be affected immediately by the new feature. The Bootstrap System was greatly elaborated from an initial system, which compiled only 12 distinct machine instructions into a complete system with elaborate input/output and macro expansion capabilities. These developments finally reached a logical conclusion, however, when we reached the limit of facilities which could be mechanized in a strictly one-pass translation scheme.

The Bootstrap language obtained its primary power for plex programming from the fact that it included the functional notation (inherited from LISP) for referring to a component of an element, so that the basic operations of plex programming could readily be expressed in a natural form. Aside from the initial FAP-coded Basic Bootstrap translator (a very compact program of only a few hundred words of instructions and table settings), the entire Bootstrap System was written in its own language, making use of each new feature as it was introduced. Our intent was that recoding of the Basic Bootstrap would then allow the entire system to be carried over to other machines, and in the early stages we did in fact perform this operation several times as we developed the system simultaneously on the TX-0 and 709 Computers. As the system became more elaborate, however, the programming techniques became more machine dependent, and the feasibility of the bootstrapping process diminished. It was at this time that the supreme importance of control of mechanization within the translation system was hammered home.

C. THE ALGORITHMIC THEORY OF LANGUAGE

In parallel with the later developments of the Bootstrap Compiler system, we had been making investigations into other areas, particularly verbal and graphical language definition and translation. Out of these investigations came the Algorithmic Theory of Language, which was the first real exercise in applying the total plex concept to

a particular modeling process. The success of these research ideas led us to the initiation in 1962 of a "Multi-Pass Compiler" development, which soon evolved into the AED-0 effort as we know it today. Having discovered that the techniques and power of the Bootstrap System were inadequate for the goals which we wished to achieve, our intent was to use the Bootstrap to make a more powerful system (AED-0) which we would use only within the Project to create a first system (AED-1) which we would attempt to make available for widespread industrial and scientific use. It is for this reason that the initial AED System was christened AED-0 because we wished to have AED-1 be our first public effort. As history worked out, AED-0 evolved into a much more powerful system than we had originally planned so that we did in fact make it public, and moreover formed the AED Cooperative Program, described in a later chapter, to spread its use and invest in its further development. Thus AED-1 is not our first public distribution, but it still is the first achievement of machine-independent techniques. In this chapter, we outline briefly the main features of the AED-0 language and some of the relevant AED-0 and AED-1 Compiler features for plex programming.

D. ALGOL-60 AS A BASE

Since AED-0 was to replace the Bootstrap System as the primary programming vehicle of the Project, it was desired that it be a completely general-purpose programming language not related in any way to any particular form or aspect of computer-aided design. From our participation in the programming language standardization activity of the American Standards Association we were well acquainted with the Algol-60 language and had in fact been using portions of it (along with various natural-language English constructions and graphical language forms) as test material for the research in the Algorithmic Theory of Language. Algol was by far the cleanest and most rigorously defined language at the time and, in fact, we were surprised by the completeness of the language which was disclosed by our processing of it by the techniques of the Algorithmic Theory of Language. We therefore decided to base our new language on Algol-60, deviating from Algol only where modifications would improve

processing by the language theory, or to omit some Algol feature which was not essential for system programming or which would be inordinately expensive to implement. Since AED-0 was intended to be used only within the Project, we did not expend any great effort in careful language design on the additions which we made to the Algol base.

The basic data types of AED-0 are the REAL (floating point), INTEGER (fixed-point), and BOOLEAN types of Algol. In the original AED-0 Language all other data types were treated as type INTEGER since code generation was the same for all other types. In the final AED-0 Language, however, an additional type POINTER was introduced when a distinction between machine address quantities and integers was required. In addition to having simple variables of these types, one-dimensional arrays as in Algol also are allowed. Multi-dimensional arrays are omitted as being unessential to system programming.

The most important innovation for plex programming is the introduction of components of these various basic types. Components may occupy either a full word or any portion of a word, down to individual bits for boolean components. The component declaration facility of AED-0 is crude, but effective. The general format for the declaration of components is shown in Fig. 8.* The COMPONENTS

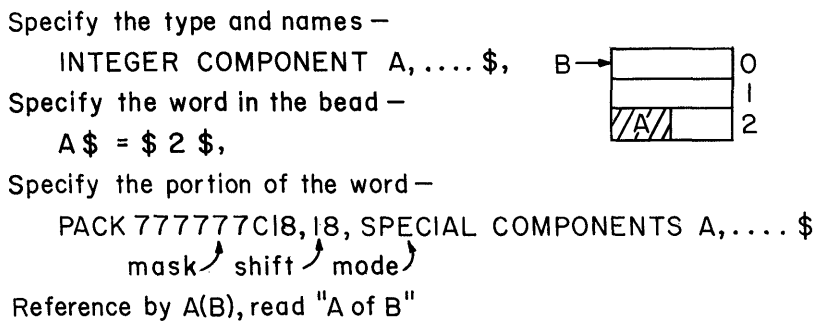


Fig. 8 Component Declaration in AED-0

statement declares the type associated with the mnemonic names, the "$=$" statement assigns a word location relative to the beginning of the element containing the component, and a PACK statement is given

*The "$, " is treated as a single statement separator like the Algol "; ".

if the component is not full word. The value of the component is assumed to be right-justified, and the arguments of the PACK statement give respectively a mask ("C" indicates octal number conversion), a left shift, a standard word portion if applicable for efficient processing, and the relevant component names. The same functional notation A(B) as used in the Bootstrap is used to access values of components.

The component mechanism is used in conjunction with the routines of the system-supplied Free Storage Package. The user has explicit control over all storage manipulations concerned with n-component elements, requesting and returning "beads" (or n-component elements) whenever desired by calls on the appropriate free storage functions. The Algol block structure is not used for automatic allocation control since it is not yet known how to accomplish these functions in a general way with plex programming. Similarly, although the Free Storage Package has a "garbage collection" mode, this merely involves conversion of contiguous small returned beads into larger beads. There is no automatic garbage collection such as is found in simple list processing languages. There are both simple and complex versions of the Free Storage Package, and the user may select features which yield complete control over the use of storage.

E. AN EXAMPLE

The general flavor of the use of AED Language to implement the growth algorithm which builds the data structure for a specific plex may be illustrated by means of a simple example as shown in Fig. 9. The objective is to insert a new value in an ordered list of values. The list is mechanized in terms of 2-component elements with the zeroth word a pointer component with mnemonic name NEXT, and the first word an integer component with the mnemonic name VAL. The growth algorithm itself employs four state variables, an integer variable I which is assumed to contain the new value to be inserted, and pointer variables Q, R, and LIST (whose value is the list in question).

Following the declarations and initialization, the program begins by reading a new value into I, using the generalized input function GIN. The program then scans the list, starting at LIST. When a value not less than I is found, a new element (or bead) is obtained from free

The most elegant and compact version-

```
        BEGIN ... FULLY NESTED VERSION //
        .INSERT MAIN  $,
        POINTER LIST, Q,R $,
        INTEGER I $,
        POINTER COMPONENT NEXT $, NEXT$=$0 $,
        INTEGER COMPONENT VAL $, VAL$=$1 $,
        VAL(NEXT(LIST=FREZ(1))=FREZ(2))=1C34 $,
LOOP$   ISI(GIN(I),STOP) $,
        R=LIST $,
SCAN$   IF VAL(R=NEXT(Q=R)) LES I THEN GOTO SCAN $,
        NEXT(Q=NEXT(Q)=FREZ(2))=R $,
        VAL(Q)=I $,
        GOTO LOOP $,
STOP$   FINISH( ) END FINI
```

Which can mechanically be expanded for people to understand-

```
        BEGIN ... FULLY EXPANDED VERSION //
        (Same Declarations)
        LIST=FREZ(2) ... GET A "LEFT BOUNDARY" BEAD $,
        NEXT(LIST)=FREZ(2) ... ATTACH A "RIGHT BOUNDARY" BEAD $,
        VAL(NEXT(LIST))=1C34 ... MAKE VALUE OF RIGHT BOUNDARY
                                                  INFINITE $,
LOOP$   ISI(GIN(I),STOP) ... READ AND TEST A VALUE. GOTO STOP IF
                                                  NOT INTEGER $,
        R=LIST ... INITIALIZE SCANNING VARIABLE TO START OF LIST $,
SCAN$   Q=R ... INSIDE SCAN LOOP, Q REMEMBERS PREVIOUS BEAD FOR
                                                  SPLICING $,
        R=NEXT(R) ... MOVE SCANNING VARIABLE TO CONSIDER NEW BEAD $,
        IF VAL(R)  LES I THEN GOTO SCAN ... IF VALUE IN LIST SMALLER, KEEP
                                                  GOING $,
        NEXT(Q)=FREZ(2) ... WHEN GET HERE VAL(Q)<I≤VAL(R) SO PUT NEW
                                                  BEAD AFTER Q $,
        Q=NEXT(Q) ... NO LONGER NEED OLD Q, SO POINT TO NEW BEAD $,
        NEXT (Q)=R ... TIE R BEAD AFTER NEW BEAD, COMPLETING THE
                                                  SPLICE $,
        VAL(Q)=I ... FINALLY PUT IN THE VALUE THAT CAUSED THE
                                                  COMMOTION $,
        GOTO LOOP ... NOW READY TO READ A NEW VALUE, SO GO BACK $,
STOP$   FINISH( ) END FINI
```

Fig. 9 Growth Algorithm for Ordered List

storage by the FREZ function. The NEXT components are updated to splice the bead in place, and the program loops to obtain a new value.

AED-0 embodies a powerful concept called phrase substitution which allows a phrase of any complexity to be substituted in the same place that an atomic symbol of the same type may appear. Since assignment statements (such as "LIST=FREZ(2)") take on the value being assigned, that assignment statement may be nested in the component reference, as shown in Fig. 9, yielding a compact expression and efficient code. The second version of the program shows how the "... REMARK" feature of the language may be used for clear program documentation. The expansion of the nested program is a mechanical process easily performed.

The simple example of Fig. 9 illustrates the main features of the mechanics of plex programming. All of the declared state variables may be considered to be attached to the framework of the growth algorithm program itself (which in turn is attached to the universe through the operating system of the computer), and the pointer state variables may be thought of as clamps which attach themselves to different pieces of data structure, holding them until connecting pointer components are set by the various assignment statements. An algorithm building a data structure is in a very precise sense an elaborate assembly machine, efficiently mating the various subparts of the overall resultant data structure in the proper sequence. Elaborate data structures require much more elaborate programs of course, but the principal of operation is the same.

F. ADDITIONAL FEATURES

The free storage and component declaration features of AED provide the basic machinery for the structural aspect of plex, and the real, integer, boolean, and pointer types provide for basic data. An additional form of data provided by the AED-0 language is the "quoted character string", of which there are various types, the most universal being the ".C." form. The AED-0 method of quoting is extremely convenient, for any character from the character set may be used dynamically to quote an arbitrary string of all the remaining characters. Thus for example

```
A = .C. 'ABCDE'
B = .C. FABCDEF
C = .C. /ABCDE/
```

are all equivalent representations of the character string ABCDE. The internal value of a .C. expression is a pointer to the character string so that character strings may be freely used as data. System-supplied routines allow various manipulations with character strings such as concatenation, counting number of characters, breaking into substrings, etc.

Some of the most powerful innovations of the AED-0 language are concerned with facilities for expressing the algorithmic aspect of plex. In addition to the basic programming language features inherited from Algol and illustrated above, various liberties have been taken with the program structural aspects of Algol to provide a much richer basis for structuring algorithms. Perhaps the most important of these innovations is the mechanism whereby values of arguments ("actual parameters" in Algol terminology) are passed to a procedure body when it is called. In AED-0 a procedure definition takes the following general form

```
DEFINE INTEGER RECURSIVE PROCEDURE F(X, Y)
    WHERE REAL X$, INTEGER Y TOBE ... $,
```

where the declarator RECURSIVE may be omitted if the procedure is not recursive. Following the word TOBE and terminated by the statement terminator $, is the procedure body, which is either a single statement or an arbitrary program bracketed by a BEGIN...END pair. X and Y are the arguments of the procedure and may be used in the expressions of the procedure body along with any other fixed variables.

If the procedure is a valued procedure or function as in the example, the body must contain somewhere an assignment statement in which the value of an expression is assigned to the name of the procedure, as in Algol. A procedure call then takes the general form

```
... A + F(C, D)+...
```

which indicates that the function F is to be evaluated using the current values of arguments C and D, and the resultant integer value is to be used in some arithmetic expression.

Instead of the "call by name" or "call by value" parameter mechanisms of Algol, AED-0 uses "call by LOC". Pointers to the location of the values of C and D are transmitted to the procedure body. This method is both very general and ideally suited to plex programming, since arguments of a procedure call quite frequently have entire plex structures as values, so that the LOC pointer indicates an arbitrarily large amount of information with the same direct efficiency as a simple value.

The LOC concept is properly defined for all program entities, and in fact "LOC" is a vocabulary word in the AED-0 language. Thus pointers to arbitrary expressions or quantities may be generated by the programmer any time they are needed. For straight-forward programming, LOC is almost never required, but for sophisticated software system programming it can be used very effectively. Since the LOC operator is defined for all types of entities, including labels and procedure names, all of these entities become valid data for manipulation within AED programs.

The use of call-by-LOC is complemented by a single legal loophole (referred to as legitimate AED-0 "pornography"), which allows "user beware" manipulations of even the inner workings of the system, with the AED-0 language. The loophole is simply the fact that the AED-0 compiler purposely does not check the type of an argument used in a procedure call against the type of that same argument in the procedure definition. Thus, for example, an argument may be declared to be of type integer in the procedure definition and yet a pointer may be used in a call on that procedure. Thus the nature of the purposeful pornography has a very crisp definition, easily understood by the user of the language, and with this loophole the high-level AED-0 Language may be used for very lowly operations which are machine and implementation dependent. Lack of such a facility in other high-level languages is one reason why they frequently cannot be used as effectively as can AED-0 for system programming. It should be remarked that in future AED languages, this feature will not be a loophole in the entire system. Careful type checking will be the normal mode of operation, removing the user-beware aspects, but a key will still be provided to the user to lock out the checking when the loophole would serve a useful purpose.

Another innovation of AED-0 regarding the procedure mechanism is the ability to declare a procedure separately from its definition. This not only facilitates the combination of procedures whose definitions lie in separate compilations into a single running program, but also it is useful in making packages of procedures in which calls on certain procedures influence the behavior of others. Consider the following simple example:

```
PROCEDURE F1, F2 $,
DEFINE PROCEDURE SET(A) WHERE POINTER A TOBE
    BEGIN DEFINE PROCEDURE F1 (X) WHERE INTEGER X TOBE
        BEGIN...body using A and X...END$,
        DEFINE PROCEDURE F2(Y) WHERE BOOLEAN Y TOBE
        BEGIN...body using A and Y...END$,
    END$,
```

In this example, the body of procedure SET consists solely of the procedure definitions of F1 and F2. The definition of SET serves also as its declaration, but F1 and F2 are globally declared to be procedures separately from their definitions. This construction would be useless in Algol since F1 and F2 could only be called within the BEGIN...END block of the SET procedure body. The prior declaration in AED-0, however, makes it possible for F1 and F2 to be called from the same block in which SET is itself defined. Thus if SET is called with a particular pointer as argument, then all subsequent calls on F1 and F2 will use that pointer in evaluating their bodies. Calling SET with a different value for A will change the behavior of subsequent calls on F1 and F2. This general technique may be elaborated to give very sophisticated program behavior in a very efficient way.

G. THE MACRO PREPROCESSOR

In addition to the compiler proper, the AED compiler system includes an elaborate macro preprocessor which precedes compilation. The macro preprocessor converts an AED-0 source language item* string into another AED-0 item string which then is translated into

*An item is a single unit of symbol, vocabulary word, or punctuation.

computer instructions by the compiler phase. The features of the macro system provide an important addition to the capability of AED-0 to serve as an expressive medium for plex programming. A macro definition has a format similar to that of a procedure definition, and the macro call notation is identical to that for procedure call. The definition of a macro with arguments has the general form

```
DEFINE MACRO M(X,Y) TOBE IF X NEQ FIXED THEN BEGIN Y END
ELSE CALL.ALARM(X) ENDMACRO $,
```

Notice that for macros no WHERE portion is needed, since all arguments are merely item strings and have no data type. The body of the macro is the item string bracketed by TOBE...ENDMACRO. Whenever a macro call of the form M(A, B) appears, where A and B are item strings (portions of program), the macro preprocessor will replace that call by an exact copy of the macro body with the item strings A and B substituted in full wherever the arguments X and Y appear in the definition form of the macro body. As with procedures, macros may be defined within macros, in which case the inner macro becomes active only following a call on the outer macro, and very elaborate and useful structures may be built. The macro preprocessor also includes a SYNONYM feature whereby the spelling of any item, including the reserved words such as THEN of the language itself, may be changed to alternate spellings, so that very powerful manipulations are possible.

The fact that A(B) may represent a function of an argument, a component of an element, an array with index, or a macro call with argument, enables many changes of mechanization of a plex from one form to another to take place merely by altering the declaration portions of a program. Although the facilities of AED-0 are not complete and we have many further features which are being held in abeyance pending the design of a still more powerful AED-1 Language, the AED-0 features are sufficiently complete and self-consistent to enable many of the most useful manipulations to be carried out automatically and reliably even now.

The AED-0 language also includes many additional non-Algol features which are useful for plex programming and system programming

in general. Examples are automatic stack declaration and manipulation, bit manipulation, character string manipulation, and a preset facility which allows variables and arrays to be initialized at compile time.

H. INTEGRATED PACKAGES

Modification of AED-0 as a language ceased in 1964 and further development of language features has been made in the form of "Integrated Packages" of procedures which represent the raw semantics of language features which will be given appropriate syntax in future AED languages. The "culture" of AED usage depends heavily upon these packages, several of which interlock directly with the compiled features of the language itself. For example, the ISARG Package permits the use of optional arguments in procedure calls; the DOIT Package permits procedure names to be stored in and executed from data structures (including dynamic loading) so that program control and data structures are interlocked, enabling generalizations of such techniques as "coroutines" to be easily performed; the GENCAL Package permits dynamic compilation of molecular procedure calls at run time. Various techniques for using these features in combination with the ability to nest procedure definitions and declare procedures separately from their definitions permit many sophisticated control structures such as multientry procedures to be developed. Other packages provide facilities for system building. The RWORD Package gives sophisticated free format input and the ASMBL Package gives free format output of character streams; the Generalized String Package creates and manipulates arbitrary "string" structures of arbitrary elements in any combination, including ordered and unordered uni- or multidirectional lists and rings, stacks, queues, hash-coded tables, or other more elaborate specialized data structures; the interface for word- and character-oriented files for arbitrary storage or input/output devices including control of logical and physical records, buffering, timing, etc.; the Delayed Merge Package and the Generalized Alarm Package permit segmentation of large program actions including alteration of control and a wide spectrum of error-handling facilities. Although many of these packages are more closely related to software engineering than language, their use is so integral with the direct linguistic

features of AED-0 that they form a significant part of the pragmatics of AED-0 as a language.

In summary, the AED-0 Language compiler system includes macro preprocessing and compiling phases and an extensive library of integrated packages for extending the language and providing major building blocks for system building. It provides a very adequate expressive medium for the physical realization of the philosophical plex programming concepts which are at the roots of our computer-aided design effort. There are numerous half-baked ideas, partially written programs, and completed packages and subsystems which have not yet been worked into the overall fabric, so it is clear that the current level of achievement is far from the final word on these matters. Much further research and development in these important areas is planned and underway, but the current capability is quite adequate to meet the current and immediate future needs of ourselves and of the various industrial and scientific organizations for whom the system is intended.

CHAPTER V

SYSTEM-BUILDING SYSTEMS

A. INTRODUCTION

The preceding chapters have introduced the plex philosophy as a theoretical basis for representing all of the relevant facts and attributes of a problem and the AED-0 Language and Compiler as an initial but useful expressive medium for making that philosophy into a working tool of sufficient power and generality to suit the needs of computer-aided design. By themselves, however, the plex philosophy and the AED Language in their raw state are an insufficient response to the mandate of the Project. They require far too much detailed knowledge and individual creative insight to be broadly useful in the construction of specialized computer-aided design systems for the numerous potential areas of application. Their elegance and power are quite essential for providing the cohesive foundation which is required for computer-aided design, but they must be embedded in a larger framework more specifically directed to the creation of specialized user-oriented systems if large numbers of users are to be able to make use of them. We therefore turn momentarily from abstract generality to a direct consideration of the man-machine problem-solving process.

According to Webster's, to design is "To plan mentally; to outline; to scheme." Thus if we are to have "computer-aided" design rather than "automatic" or "computed" design, or "computer aids to" design, we must make the computer a partner to the scheming process. This requires blending the man and computer into a problem-solving team intimately coupling the best characteristics of each so that the team works better than either one alone. As has already been mentioned, "design" is a special term for some ill-defined type of problem-solving, but no distinctive features are reflected in a system for design versus a system for general problem solving. A single design or problem-solving system to be used for all applications would be impractical and inappropriate. Many

systems are needed, each of which must

1. use the specialized jargon of its field of application,

2. require little or no knowledge of computer programming to be used effectively,

3. be evolutionary and able to adapt to the changing needs of its users, and

4. be created and maintained by the users themselves or by skilled local staff who are in intimate contact with the users.

In order to achieve these goals, the efforts of the Computer-Aided Design Project have not been directed toward a computer-aided design system, but rather toward a systematic way for making specialized systems.

B. THE AED APPROACH

Actually we have evolved a system of systems for making systems, along with an orderly method for applying them. This collection of concepts and working tools we refer to as the AED Approach. In this chapter we outline the main features of the AED Approach, indicating how it provides a general framework within which the comprehensive foundation of plex programming may be brought to bear on the system-building problem.

In order for a man and a machine to be coupled to form a problem-solving team, they must be able to communicate meaningfully. The AED Approach is based directly upon a universal model for the communication process in terms of which any man-machine system may be viewed. According to this model any communication requires several distinct steps or phases.

The four major phases of the problem-solving process are shown in Fig. 10, and are summarized below:

1. The lexical phase allows the input signal to be broken into discrete items which constitute the atomic units of the message.

2. The parsing phase enables the atomic items to be grouped into phrases and sentences so that the structure of the message (and indeed whether it is in fact a well-formed message) can be determined.

3. The modeling phase extracts the meaning from the message and formulates an understanding of the problem posed by the message.

4. Finally, the analysis phase carries out the solution to the problem which has been understood.

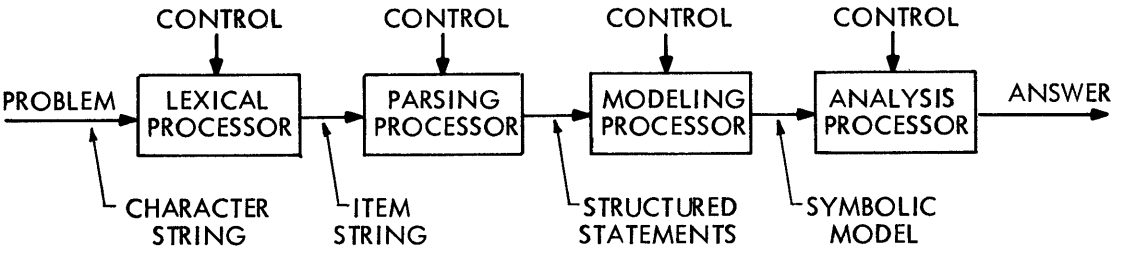


Fig. 10 General Problem-Solving Scheme

One of the principal goals of the MIT Computer-Aided Design Project is to reduce this general idea to concrete form by devising and implementing a generalized problem-solving system of the form shown in Fig. 10, and which is efficient, practical, and economical for general use. The four phases listed above provide the basis for systematizing the system-building process, for it is possible to devise generally useful basic operations for each of the four phases which are independent of the particular application area. Thus these common portions may be constructed once and may be used to provide the major structure of any number of specialized systems.

C. GENERAL PROCESSORS

There are many possible techniques for physically representing the skeleton framework of the four phases of the problem-solving system. All of these forms may be thought of in the same way, however -- as generalized table-driven processors. In other words, these processors may be constructed in such a way that merely by supplying the appropriate control information, the detailed behavior of the generalized processor may be adapted to fit the specific needs of a given application system. The massive intricacies of the complex behavior required for satisfactory functioning of each phase is, for the most part, automatically provided by the generalized processor, and the setting of the control information is a much simpler and less exacting task than creating a specialized processor for that phase from scratch.

Even though setting up the control information for generalized table-driven processors for the various phases is much easier than constructing those phases from scratch, nonetheless for reasonably rich application languages the setting of the control information is itself a complicated task. We may think of the generalized processor for a given phase as a strange kind of computing machine and the setting of the control information as writing a machine language program for that strange device. Just as compilers are developed to ease the programming problem for ordinary computers, high-level languages and compilers can be constructed to ease the problem of programming the control information as well. This is the route which has been followed by the MIT Computer-Aided Design Project. The generalized processors for the four phases are the target machines for distinct high-level programming systems. Thus the AED Approach physically takes the form of a system of systems whose outputs are specialized processors for the required four phases of a desired specialized system.

Figure 11 shows how the general AED Approach scheme has been applied to the design of the AED-1 Compiler itself. AED-1 is a

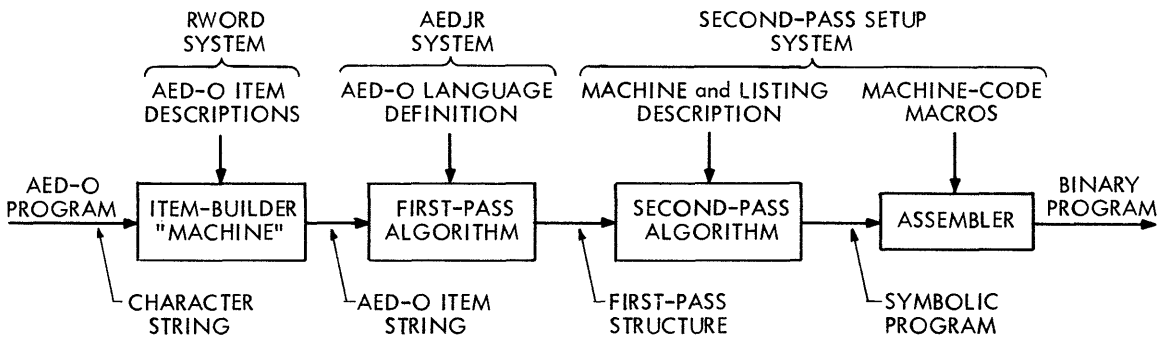


Fig. 11 General Structure of AED-1 Compiler

machine-independent and language-independent compiler system. Merely by changing the control information it is possible to compile many languages for one machine, one language for many machines, or many languages for many machines. The figure illustrates how the general AED Approach scheme has been set up to translate the

AED-0 programming language into binary programs for various machines, in order to bootstrap the AED System onto various computers.

The high-level system which sets up the lexical phase is called the RWORD System ("read-a-word"). The input language to RWORD is a version of the "regular expression" language of atomata theory which may be used to describe how sequences of characters group together form the items (vocabulary words, symbols, and punctuation) of an AED-0 input string. The output of the RWORD System is a lexical processor in the form of a specialized "finite state machine" program which converts a character stream into a stream of items ready for parsing.

The high-level system for setting up the parsing processor is called the AEDJR System. It is based on the Algorithmic Theory of Language which has been developed as part of the Project activity for this important phase of language processing. The AEDJR System has a number of distinct command languages which permit the metalinguistic properties of a new language to be described and checked out. The processing involves introducing the vocabulary words of the language, ascribing "types" to those words, and describing which types "like" to group together to form phrases and what is the type of the resulting phrase. This and other information is automatically transformed into the appropriate type of control table to control the First-Pass Algorithm of the language theory. This algorithm and the resulting control tables form the specialized parsing processor for the second phase of the system being built. This parsing processor converts the item string supplied by the lexical processor into the "first-pass structure" which shows both the syntactic and semantic parsing of an input statement in the new language. The syntactic parse shows the grouping of the words and phrases into larger phrases, and the semantic parse shows the sequence in which the phrases are to be considered in order to build up in an orderly fashion the meaning of an entire statement from the meaning of its subparts.

D. AED-1 COMPILER STRUCTURE

The RWORD and AEDJR Systems are sufficiently well developed for general use, but generalized processors for the modeling and analysis phases are available at present only for the specialized problem area of compiling computer programs. In the AED-1 compiler system, the high-level system for setting up these phases is called the Second Pass Setup System. The facilities of the AED-0 Language itself are used to establish the control information for this system, and a systematized Second Pass Algorithm provides the basic framework for translating the AED-0 Language into various machine assembly languages. Using high-level formats, the translation of the various atomic units of semantics of the language are expressed symbolically in terms of output character strings suitable for the target assembler, including transformation operations to convert various coded quantities into the appropriate forms and values. A generalized framework is also provided for the important Selector Function which embodies the strategy of code generation. The output of the modeling phase is a character string in the same format as a hand-coded machine language program. The target assembler of the target machine, which represents the specialized analysis phase, then produces the desired binary program ready for loading and execution.

Figure 12 gives sample intputs to the RWORD, AEDJR, and Second Pass Setup Systems for describing portions of the AED-0 Language. It would be impractical to attempt to give any sort of detailed description of these languages in this report. In general, however, the systems which represent the AED Approach provide a broader spectrum of capabilities than are found in any other systems which perform similar functions, and it is possible to produce high-quality software systems which are sufficiently refined for economical production use. Without this degree of sophistication, specialized systems produced in this manner would be inordinately expensive for heavy application.

Specify character classes –

LET = /ABCDEFGHIJKLMNOPQRSTUVWXYZ/

DIG = /0123456789/

Define lexical types of items by regular expressions –

SYMBOL (1) = LET/(LET U DIG)*$,

NUMBER (2) = DIG/DIG*$,

where / = concatenuate, U = or, * = none-or-more

(a) RWORD Inputs to Define Itemization of Language

Specify vocabulary word and initial type –

VIN word itype

For each type, specify list of "like" types, output type, and special attributes –

LIKE	itype	otype 1	ℓ type 1	ℓ type 2	ℓ type 3
	= =	otype 2	ℓ type 4	ℓ type 5	
		ATTR	code1	code2	

(b) AEDJR Inputs to Define Parsing of Language

Write set of special Type Functions to be performed before standard operations.

Write set of special Action Functions to process standard arguments.

Combine in PRESET statements to translate atomic semantics into Assembly Language output.

PRESET MRBTBL=... TABLE TO BE SET//

17$/$.C.'.LA.AC, $A0.$C0/'...LOCATM//

23$/$.C.'.ORG.DATA+$D1.$C0/'...ORG//etc.

where number $/$= Type Function specification,

.C.'-----' = Driver for Assembly output,

.= Tab, / = Carriage return,

$ LET NUM = Special Function LET applied to argument NUM

All other characters are output directly.

(c) SECOND PASS Inputs to Define Code Generation

Fig. 12 High-Level Inputs to Set Up AED Systems

E. RECAPITULATION

Before continuing, it is worth while to give a brief recapitulation of the over-all viewpoint of the AED Approach to man-machine problem-solving systems. There are a few more embellishments which form an integral part of the "system of systems" which must be introduced before we can outline the making of systems for practical purposes. A word of apology is due for the seemingly endless layers of elaboration and complexity which are being presented here, but this inescapable confusion is primarily due to the newness of the field which we are discussing. It is as though we were to describe the field of symbolic part programming for numerically controlled machine tools at a time when it was necessary to describe not only the computer programming aspects, but also the concept of a general-purpose machine tool and all of its underlying technology, including production of precision lead screws, lapping of sliding joints, stress-relieving of forgings, refining of lubricants and coolants, etc., etc. Any high-level technology is built upon a fantastically broad pyramid of underlined technologies, and whenever even a modest portion of the underlying technologies cannot be taken for granted, the process of description becomes very involved.

So the recapitulation is as follows:

A single computer-aided design system suitable for all classes of problems would be undesirable even if it were not inconceivable and impractical. What is needed is a very large number of highly specialized man-machine problem-solving systems in which the user of the system need learn only a natural and expressive specialized lanquage based upon the thought patterns and jargon of his specialized field of interest. Use of simplified languages to control sophisticated calculations implies that a high degree of automation must be incorporated directly into each of these specialized problem-solving systems. In order to achieve this goal of many sophisticated specialized systems, a high degree of automation is also required in the manufacture and modification of those systems themselves. It is therefore necessary to have an underlying complete philosophical approach to man-machine problem-solving in general, in order to provide a basis for

this automation of the system-generation problem. This common backbone consists of recognizing that the first step is to place the man and machine in communication with respect to the desired problem area or area of discourse. Communication in turn is based upon processing of a time signal constituting a message transmitted from one intellect to another. The common backbone envisions four general phases to the processing of this message. The lexical phase recognizes the words and syllables (generally called "items") of which the message is composed. The parsing phase organizes these items into phrases, sentences, and paragraphs in a richer structure then the linear time sequence of the message, providing a basis for extracting the meaning of the message. The third phase, the modeling phase, performs this extraction by constructing the understanding of the message, and the fourth phase, the analysis phase, then analyzes the resulting model to complete the communication process.

The four phases of the backbone provide the basis for automating the system-building process, for each phase may be represented by a generalized processor which is capable of performing the right kind of actions if properly instructed. Since these generalized processors are elaborate and sophisticated in themselves, automation also is required to set up the control information for each of the phases. Thus there are four systems with specialized input languages whose outputs control the generation of four specialized processors which compose the required phases of a specific problem-solving system. The resultant system will take the user-oriented language as input and provide the desired answers, results, or behavior as output.

Let us refer to the particular sequence of lexical, parsing, modeling, and analysis phases as a translator. It is clear that a translator translates some input string message into some output result. The input may be any time sequence of any sort of things, and the result may be any desired effect, including control of equipment, setting of variables or quantities in other programs, or the generation of some output message sequence. We will now make use of this general idea of translator to elaborate on the concept of the translator itself, and to show that actual translators for useful languages

are composed of interconnected sets of translators for many sublanguages of the total language for which the o verall translator is designed.

F. PARSING

As a starting point, we will examine the action of the parsing phase as exemplified by the First-Pass Algorithm of the Algorithmic Theory of Language in more detail. Our objective is two-fold: 1) to clarify the important role played by parsing in the translation of a language, and 2) to show how a stream of items constituting a message may be considered to be a mixture of words from many languages, all of which work in concert to constitute a rich over-all language amenable to smooth and efficient processing.

Recall that the purpose of the parsing phase was to restructure the linear time sequence of the items in the input string into a more elaborate form showing how items are grouped together into meaningful phrases and sentences (the syntactic parse), and also how the meanings of the individual items and phrases are to be considered in order to build up the meaning of the total from the meaning of its subparts (the semantic parse). This important function is carried out by the parsing algorithm in one of its many forms. In general, the output of the parsing algorithm is a <u>first-pass structure</u> which consists of syntactic and semantic parts interlocked into a single data structure as shown in Fig. 13.

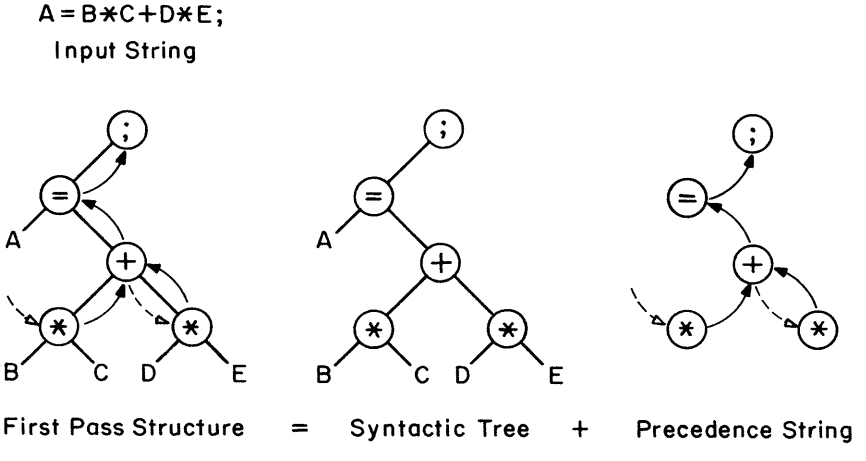


Fig. 13 The First-Pass Structure

Figure 13 illustrates a simple first-pass structure for an algebraic language for which the only active vocabulary words are the mathematical operators and the statement terminator punctuation character. Notice that only these words participate in the semantic parse or "precedence string" structure. In the syntactic parse, the right and left context of each of these vocabulary words is shown by the binary tree structure. The literal symbols in this example are not properly part of the language, but instead are the atomic data upon which the meaningful operators of the language are to act. The combined first-pass structure shows both the syntactic structural relations and the time sequence in which the semantic evaluation is to take place. By following the precedence string pointers (taking a dashed minor precedence pointer to form a fresh beginning whenever a new one is present and otherwise following the solid major precedence pointers), the expression may be properly evaluated to drive the modeling and analysis phases.

G. NESTED LANGUAGES

To bring out the fact that input strings for many languages may be intermeshed into a single message string, we need merely note that the atomic symbols in Fig. 13 need not consist only of single items as shown there. Instead, they may be arbitrarily long strings of items constituting the data and vocabulary words of some "foreign" language, i.e., all items which are not vocabulary words of the language currently being translated. The "second-pass" operators constituting the modeling and analysis phases for the language currently being processed may feed these item strings through to the output of the translation process for further translation in the context of some other active vocabulary tables, or they may treat the item strings as data and process them in any other way. In particular, notice that the second pass for one language may be an entire translator for another language, i.e., once the proper context in the outer language has been determined by the completion of its parsing phase, its second pass may consist of taking the multi-item "atoms" as inputs to another entirely separate translator. The output from that translator

then can supply the "meaning" for use by the second pass of the original language.

A converse type of operation also is possible, in that part of the processing of the lexical or parsing phases of one language may involve the invocation of an entire translator for a foreign language. In such a case, the output from that translator may constitute a single item of the original language (which appears in its first-pass structure as a single node) even though the input string contained a multi-item message. This is in fact the way that quoted character strings are processed in the AED-0 Language. When the item ". C. " appears, the AED-0 lexical processor turns control over to a special quoting translator. This translator proceeds to read the quoting character and compose a multi-character single item, stopping when that quoting character appears once again. At that time, the character string body which has been quoted is "wrapped up" as a single item and control is returned back to the AED-0 lexical processor for further processing of the input string.

In addition to dynamically controlling the processing of character strings and item strings in the above manner, translators may also be embedded in the processors for the phases of another translator in yet more devious ways. In particular, result of translating some message in a sublanguage may affect the processing behavior itself of the translator of the parent language. Thus for example, the occurrence of a word or phrase in a foreign language may trigger the change of the parent vocabulary to a different sublanguage, so that the subsequent processing of the remainder of the message takes place in a different language. Although the mechanism is quite different, the idea is similar to the above example of turning on and off the quoting translator when processing a character string.

H. MACRO PREPROCESSING

All of these techniques are used in a general way in the overall generalized translator scheme which we have been discussing, and in fact their use introduces a fifth general phase into the overall translator concept called the macro preprocessing phase. The macro preprocessing phase is interposed between the lexical phase and the

parsing phase, so that the completely general backbone scheme for a translator consists of lexical, macro, parsing, modeling, and analysis phases. After describing briefly the role of the macro preprocessor, we show how these techniques may be used for graphical language forms as well as verbal language forms, in terms of which full-blown computer-aided design systems may be constructed.

The preprocessing phase takes place between the lexical and parsing phases of our previous description. It accepts as input the item string supplied by the lexical phase and yields as output another item string which is used as input to the parsing phase. Like the other phases, the preprocessing phase incorporates a general processor which is table-controlled. Some of the control information is set up when the main characteristics of the translator are adapted to a given language, and other control information is generated dynamically by the actions of the preprocessing phase itself as it operates on an input string. Since the preprocessor is an item-string-to-item-string converter, it provides several more highly useful degrees of freedom for the design of user-oriented languages. The preprocessing phase is itself a complete language translation system, translating the message received from the lexical phase into a modified message for the parsing phase. The preprocessor does a complete translation of only part of the message. Those portions of the item string which are not in the language of the preprocessor constitute the data which are used to compose the output message.

The basic mechanism of preprocessing is that of macro definition and macro call. By means of a macro definition statement the user of the language says, "Whenever I say this..., I really mean that...", where "this..." is a macro call specification and "that..." is the macro body. A macro call consists of a particular instance of "this..." in the input string. At the corresponding position in the output string, a suitable version of the macro body will appear in place of the macro call. In general, both the macro call and the macro body conventions may be extremely elaborate, with argument substitutions and conditional expansion depending upon argument values and various forms of context. The particular syntactic forms which control the semantics of macro definition and expansion may be set up to suit the

"flavor" of the host language, so that all languages defined using the AED Approach may have powerful macro preprocessing capability as a natural and integral feature.

The MACRO System is itself set up using the AED Approach. The facilities of the macro preprocessor allow a language to be "augmented" by the user without actually being "extended" into new areas of discourse. Although the average user of a specialized language will not know how to extend that language, a smooth incorporation of macro preprocessing capabilities can still allow him to make many useful individualized changes in the physical format of the language to suit his purposes. This is possible because macro preprocessing merely is an automation of a form of shorthand for manipulating the words of the language itself. The scope and power of the language are not extended, but its utility and expressiveness may be greatly augmented to suit the user's desires.

I. THE CADET SYSTEM

Throughout all the preceding discussion it has repeatedly been emphasized that the entire AED Approach to the translation of languages is based upon the processing of a time sequence of events. Even though all of the examples have been given in terms of processing of character strings as the physical mechanization of that time sequence, the approach is not limited to the processing of verbal languages represented by character strings. Quite the contrary, it has been the aim of the MIT Computer-Aided Design Project from the very beginning to provide a scheme which is equally applicable to the processing of other language forms, including most importantly the "graphical languages" associated with the use of on-line display consoles. Elements of graphical language include all forms of input which a man may generate at a CRT display console. Typical of these are: switch activations; button pushes; light pen "sees" on displayed items; x-y coordinate information resulting from pen tracking; stylus manipulation on an input tablet, or track-ball rotation; and knob rotations. Whatever their source, encoded "characters" from any or all of these devices may be treated in the same manner as characters from a keyboard.

Historically, the Algorithmic Theory of Language, upon which the whole AED Approach is based, was the outgrowth of initial studies of mixed verbal and graphical language processing which began early in 1961. Although primary emphasis has been given to the processing of verbal language because the programming language and compiler studies underlie all of the other activities of the Project, the graphical language influence has permeated the Project throughout this period, and a modest effort at implementation has also proceeded in parallel. Since 1964 these efforts have progressed under the acronym CADET, standing for Computer-Aided Design Experimental Translator. The CADET System is intended to play the same role for generalized computer-aided design that the AED-1 Compiler plays for the compilation of computer programs for high-level programming languages. As such, it embodies the entire AED Approach, including the use of mixed verbal and graphical language, and a systematized approach to the generalized modeling phase.

The basic translator structure of CADET is the same as we have been discussing, with a few embellishments. In particular, the incorporation of graphical input necessitates the elaboration of the basic lexical processor concept to include provision for multiple input sources, such as character strings coming (directly or indirectly) from a keyboard, button pushes, light-pen data, and any other actions of a human operator at a graphical display console. As mentioned in Section E, all messages consist of a time sequence of events, and the mechanical form of the "characters" in a message is immaterial. Thus, there is no difference between accepting "A = B + C" from a keyboard or card reader, and "pen-position button-push pen-position button-push..." from a display console with light pen and push buttons.

In the Graphical RWORD of the CADET System, the facilities of the macro preprocessor described in the previous section are elaborated slightly in order to provide the appropriate mechanism for intermeshing graphical and verbal item streams into a final canonical item string form which is independent of the mechanical source of the original input. In other words, a given type of phrase may be input verbally at one time, graphically at another time, or in mixed verbal and graphical form at yet another time. These various mechanical

forms are merely different forms of macro call structures, all of which end up producing the same macro body output item string for later parsing. This is actually a greatly oversimplified description, however, because the sequence of actions appropriate to the verbal expression may be quite different from those appropriate for graphical expression, and the parsing process itself may be dynamically changed in order to result in the same inputs to the "second pass" modeling phase. In general, however, it is the intent of the CADET effort to provide the necessary generalized processing to allow the most natural expressive medium to be selected by the user for a given area of application. Development and enrichment of the basic CADET mechanism is far from complete but the efficacy of this approach has been demonstrated in work to date.

J. GENERALIZED MODELING

The other primary innovation which is being incorporated into CADET is a generalized approach to the modeling phase of the over-all AED Approach translator concept. We expect that the analysis phase will always be in the province of a general-purpose programming language such as AED-0, but the internal representation of the data structure aspects of the total modeling plex for arbitrary application areas appears to be amenable to a systematized treatment. In other words whatever the area of discourse may be, the resultant specific modeling plexes must be composed of a structure of structures built up in an orderly fashion. We are attempting as part of the over-all processor for CADET to capture the essence of this structure-building aspect in such a way that the generalized facilities thereby made available can provide the basis for building arbitrary data structures. In effect, this structural part of the CADET effort reduces some of the basic philosophical plex concepts from abstract to concrete workable terms. Just as the first-pass structure of the Algorithmic Theory of Language constitutes a basic mechanism for modeling statements in arbitrary language, this structural part of the CADET will provide a general basis for constructing the "understanding" for generalized problem areas.

At the same time that this CADET effort is attempting to represent the generalized structural aspects of modeling, we also are giving careful attention to modeling of the design process itself, and are considering the trade-offs and economies of data storage and execution time. The focal point for this research has been the "Polyface Package" which considers the modeling of complex objects composed by joining polygonal faces along their edges. It is important to note that most of the techniques being used are representative of general "structure" which we expect to find in arbitrary modeling and are not dependent upon the particular polyface example.

The following list of features of the Polyface Package illustrates the facilities which hopefully will be available in a generalized form in the not too distant future.

1. The model is fully "common subexpressioned", i. e., each distinct entity occurs only once in the system no matter how often it may be used.

2. The system only keeps track of incremental changes by means of "variation beads", so that there is no redundancy.

3. The complete history of generation of a model is recoverable from the model.

4. Not only the structure of the total model (which may represent several alternate designs) is available, but also any substructure is uniquely isolatable at any time.

5. A generalized mouse algorithm has been devised which leaves no "tracks" in the data structure of the model, i. e., the complete state of the mouse is contained within itself, so that any number of mice may be running simultaneously over the same model without interference.

6. The encoding of variations so that a mouse knows which of many variations to obey is done in a very compact optimum binary code which uniquely identifies the precise location of a bead in the structure of the entire model.

7. The beads of the model are successively generated in the natural construction sequence and are unchanged from the time of creation, i. e., there is no necessity to read back old information and make modifications.

8. The structure of the model naturally matches the concepts required to handle various storage media.

Needless to say, the ambitious nature of the CADET effort places a severe strain on all of the facilities, techniques, and concepts which have been developed in the Project up to this time. The magnitude of the multiple tasks which are being attempted makes it impossible to estimate when the progress of this research will yield plateau systems which may be directly employed in the intended manner in industry.

Appendix I to this final report mentions numerous computer-aided design application systems which have been built using various building blocks and packages which have been generated by the Project to date, however, so that even though it may be a few more years before the entire scope of the AED Approach as represented by CADET will be available in a useful form, further evolution of larger and larger partial building blocks may be anticipated.

We have now completed the general description of the "system of systems for making systems" which represents the major output of the Project. In the next chapters we consider how these systems are used to set up particular computer-aided design systems, and the vitally important aspect of graphical display consoles and coupling of on-line display techniques into a multi-computer remote and/or local time-sharing system.

CHAPTER VI

APPLICATION OF THE AED APPROACH

The preceding chapters have presented the philosophy, techniques, and systems for generalized computer-aided design which have been developed by the MIT Computer-Aided Design Project. We now consider how these results can be used in the construction of specialized man-machine problem-solving systems covering particular application areas.

A varied mix of systems (listed in Appendix I) have already been created using the AED facilities, but only a few of these have exercised the full sweep of the AED Approach. This is because most of these efforts were begun before various AED systems were in usable form, and also because many applications interfaced with systems of programs generated separately with other techniques. This situation may be expected to continue in the foreseeable future, for depending upon the interests, skills, and choice of application, the system-building user of AED may select different portions of the total facilities for use in a given situation. Some will use only the AED language and compiler facilities as a programming vehicle for independently designed system programs, writing critical portions in machine language subroutines and perhaps using only the simplest form of Free Storage Package. Others will make use of a wider variety of integrated packages, such as the String Package for data structures, the Assemble Package and a standardized version of the RWORD Next-Item routine for free format input and output, or add display facilities to an existing set of programs using only the Display Editing Package.

There are advantages and disadvantages of using isolated building blocks from the total AED facility in this manner, but it is nonetheless a welcome and fully legitimate use of Project results. Since, however, this type of building block usage is so individualized, we will not attempt to describe the pros and cons of the various techniques which have been or could be used. Instead, we will direct our attention to the more systematized usage which results from the application of the full AED Approach as facilities exist today for its implementation.

A. THE THREE-MAN TEAM

For the sake of discussion, we will assume that there are three people involved in the construction and use of a specialized computer-aided design system--a designer, a programmer, and an analyst. These may all be the same person or there may be several persons in each category, but it will be useful to make a distinction between three kinds of interests in the following discussion.

By designer we refer to a person who is interested in the ultimate use of the design system to solve particular problems in the specified application area. By programmer we refer to a system programmer skilled in the use of all of the AED system-building systems. He knows how to use the high-level languages of the RWORD and AEDJR Systems, knows their limitations, and knows how to use the general AED language and compiler facilities to overcome limitations of the systematized approach to accommodate required variants which fall outside the scope of the generalized treatment. He also knows how to track down bugs in all regions of the systems to be used, and in general is a professional software technologist with high-powered tools and specialized skills at his disposal. The analyst possesses a less detailed knowledge of the actual controls and detailed behavior of the various system-building systems, but is stronger in the abstract mathematical domain. He is an expert not only in the calculations needed to solve problems in the application area, but also is expert in plex thinking. He is able to visualize the essence of each of the various aspects of the total class of problems included in the problem area. He can express these relationships in total modeling plex terms, and is expert at modifying the definitions and declarations of portions of the model to alter the mechanization to achieve maximum generality and flexibility. We will assume that except for the designer, these individuals have already gained experience from the design and construction of several prior systems, and will consider their activities in preparing a new system to satisfy the needs of the designer.

We assume that the designer has been plying his trade for some time within the company, perhaps using various computer aids in portions of his daily activity. His design speciality is an important part of the activity of the company and in his work he has come in contact with

other design or engineering specialists in the company who have recently begun using new user-oriented systems prepared for them by the programmer-analyst team. Seeing their initial successes and listening to their plans for exciting evolutionary improvements in their systems, he has taken some time to plan such a system for his own area of interest. He has arranged a meeting with the programmer and the analyst to present them with the specifications for his system and is interested primarily in how rapidly they can place it in operation.

B. JOINT MEETINGS

The meeting begins with a presentation of the designer's specifications, including charts, diagrams, samples of the mixed verbal and graphical language statements which he will use to solve problems, and descriptions of the existing and new computer programs which will be needed. He is an intelligent man with several years of experience in his design speciality, and is well tuned to the political and economic realities of life in a carefully run industrial organization, so that his brief is well prepared and carefully thought out. The presentation ends with a listing of specific direct questions concerning availability of manpower and computer time to resolve the few open questions, and a skeleton schedule chart upon which he requests the programmer and analyst to fill in reasonable target dates for completion of the various phases of the system preparation.

The programmer and the analyst also are intelligent and experienced and acquainted with the political and psychological realities, and furthermore they have been through similar sessions several times before. They begin their response by sincerely complimenting the designer on the thoroughness of his preparation and the exciting feasibility of a specialized design system for his application area. They then begin the delicate task of maintaining the designer's enthusiasm while at the same time educating him to a few additional realities which lie outside the scope of his previous experience, but which will play a critical role in the success of the venture. The gist of this initial phase of the response, stated gently and with supporting examples drawn from previous system-building efforts, is that the programmer-analyst team may in no way be considered as a subordinate service organization to

the design and engineering groups of the company--not for any prideful or political reason, but because of the nature of the job to be done. Instead, all three aspects of the system-building problem represented by the designer, the programmer, and the analyst must be equally represented in a highly democratic team structure if a reasonable tapping of the potential for the designer's problem area is to be achieved. The net result of this first lengthy meeting is the acceptance of the designer's specifications by all three members of the team merely as an opening presentation of the general outline of the kind of system that is to be achieved. The effort of careful preparation was not wasted, for a hasty initial description of the ideas and interests of the designer would have left many loose ends to be unraveled in the future. None of the team members can at this time present significant deviations from the original "specifications", but all (including the designer), have an open mind as the meeting disbands. The programmer and analyst will study the designer's proposal in more depth, looking up specifications on existing programs and attempting to view the proposal in the light of their own specialities. The designer will read various descriptions of the system-building tools which the other team members will use, and will check back in more depth with his other designer friends to compare notes on how they fared in the earlier system-building efforts.

At subsequent meetings of the team, the focus of the group branches out from the original "specification" prepared by the designer. The analyst has found various auxiliary features in some of the computer aids which the designer has been using but which were not adequately covered in the original specifications. The programmer has studied the language features proposed by the designer in sufficient depth to show alternate forms which will yield more flexibility, entail less writing, and in general will provide a smoother user interface than the designer had realized would be possible. After a short time, the three members of the team find that they are in fair agreement as to the general outlines of the area of discourse which is to be covered by the computer-aided design system. There is confidence that sufficiently efficient computational techniques will be available, and that useful immediate results can be expected with an initial plateau of capability which is established as the initial goal. It is at this stage that the

"wouldn't it be nice if..." phase is completed, and attention focuses on the task of actually beginning the preparation of the system. Notice that new rigid specifications to substitute for the initial efforts of the designer have not been generated. There still is much to be learned by the members of the team. Greatly deepened insights into both the problem area and the kind of system which will suit it best are bound to evolve as the initial target plateau is approached. The team attempts to maintain flexibility at this early stage in order to let this developing understanding mold the final form within the guidelines of the informal outline of the target plateau.

C. THE ROLE OF THE ANALYST

At this stage the analyst and his activities become the focal point of the group. It is his job to create the semantic package for the system--the package of procedures which will carry out the actions of the modeling and analysis phases of the final system. With the target plateau of initial capability roughly outlined and bounded by the preceding generalized considerations, the starting point for real work in constructing the system lies with the explicit treatment of just what can be done in the way of formulating and solving problems in the area of discourse, independently of how the user language may look. The existence of the RWORD and AEDJR Systems for preparing lexical and parsing processors give confidence that nice user-oriented language can be incorporated into the system at an appropriate time. But whatever that form may ultimately be, it will be meaningless language if semantics is lacking. Therefore the semantic package comes first.

According to the AED Approach, the semantics of a language must be represented by a collection of actions. In other words, the only way to represent the complete set of all meanings of all statements in a given language is to have a collection of discrete actions so arranged that the act of comprehending the meaning of a statement is an act which is the composition (in some appropriate fashion) of a subset of actions selected from the total set of possible actions. Since language communication involves only a time-signal message, with no physical transport of any material of any kind, it is inescapable that the mechanism of comprehension on the part of an intellect, be it man or machine, must be an act performed by that intellect.

The importance of the analyst's role in designing the semantic package lies in the fact that the subroutines in the package neatly bound and delimit the actual area of discourse of the resulting system. Any statement which is processed as an input string will receive a meaning determined by the semantic package, and no meaning outside the domain of the package can possibly be communicated. No amount of "syntactic sugar" can overcome limitations of the semantic package.

D. THE SEMANTIC PACKAGE

The routines of a semantic package form a special-purpose library, and are used to compose the actions of the modeling and analysis phases in the final translation process. There are three main varieties of action involved: modeling, analysis, and control. A given procedure belongs to just one of these categories and is atomic in the sense that it performs some smallest indivisible unit of modeling, action, or control. The atomic procedures of a package also satisfy another most essential property; namely, almost all of the procedures are valued procedures or functions, and the values taken on by these functions are of such type and representation that the value of one function may be used as an input argument in a call on another function. Thus it is possible to make nested function calls to compose molecular functions out of the atomic functions. Atomic functions also share a common set of global variables which interlock them into a cohesive whole. For example, the fact that one atomic procedure was called in a certain way may result in the setting of a global variable which will affect the behavior of a number of other procedures if and when they are called in the future. Because each atomic procedure performs some smallest essential function, and because the entire collection of procedures gives a complete set of actions, any mode of behavior which is appropriate for the area of discourse can be achieved by the proper composition of atomic function calls, or calls upon molecular functions defined in terms of the atomic functions.

The above general principles of semantic package design are not easy for the analyst to achieve. Many attempts at formulation and reformulation are usually required before the truly essential features of the area of discourse may be seen clearly enough to be separated into

forms appropriate for atomic semantic units. Notice that in semantic package design the objective is to explicitly discard the specific attributes of any given problem within the problem area in an effort to uncover the common structure which underlies every problem. This is quite a different style of problem-solving than that to which most people are accustomed, for we usually begin the attack on a new problem by looking for specific features which may lead to a solution. Quite the opposite is true in semantic package design, and it is virtually impossible to achieve the desired result in a straightforward manner. Usually many iterations are required.

The analyst designing a semantic package must play the plex programming game to the hilt. He must consider at all times all three aspects of plex--data, structure, and algorithm. The objective, of course, is to find the most appropriate set of building block algorithms so that the modeling and analysis phases for the processing of any particular problem statement will involve the most elegant application of growth algorithm atomic functions to build the data structure to which just the right analysis functions may be applied to give the answer. In these matters, storage space and execution time must be taken into account and the full spectrum of AED capabilities for altering the mechanization of a plex model are important tools in the hands of the analyst.

In addition to designing the appropriate data structure mechanizations and the atomic growth algorithm steps for constructing them, the analyst must also design interfaces to existing analytic routines which are to be taken from the existing body of computer aids which the designer has been using. If any of the existing programs are found to be too monolithic and rigid, it may be necessary to rework them to obtain better modularity for use with the system. Many compromises must be made, and a frequent variety is the decision to postpone the reworking of some major older program until a later plateau. The analyst, programmer, and designer know that at a future time more flexibility can be available, but may accept or purposefully incorporate certain awkward limitations for the initial plateau in order to expedite matters.

E. THE ROLE OF THE PROGRAMMER

In parallel with the semantic package design, the designer and programmer have been working on the establishment of the lexical and parsing features of the language. Major portions are directly available from previous application systems, but variations to suit the new design areas are worked into the same general framework. Early in the game the programmer sets up a test subset based upon the original "specifications" of the designer and using initial partially debugged semantic package routines from the analyst to throw together a preliminary system for the designer to try out. This activity serves an important function of deepening the level of meaningful communication between the designer and programmer in preparation for the final push to prepare a truly satisfactory user language to match the semantic package being prepared by the analyst.

As the analyst completes the modeling, control, and analysis functions needed for the initial plateau, the primary focus of the team shifts to the programmer and his area of activity. The routines of the semantic package impose implicitly a very rigid structure on the use of the atomic procedures to achieve a given end. Nesting a call on procedure A as an argument of a call on procedure B will give quite a different result from calling B as an argument of A. The analyst has designed this sequence to give the most efficient modeling and analysis behavior by the system, but the designer throws up his hands in agony when he sees the awkward sequence in which successive pieces of information must be supplied to build the proper model. It is important to realize that each bit of factual information required by some elaborate molecular action of comprehension must either come into the system from the input string supplied by the user, or it must be built into the system in the form of normal case settings which are used if a required piece of information is omitted from the input string. In general, the natural sequence in which these bits of information come to mind as a user of the language composes a message will be quite different from the rigid sequence imposed by the semantic package structure. The very real and understandable concern of the designer when he first recognizes this awkwardness lies in the domain of the programmer member of the team, for one of the most essential functions of the parsing phase of translation is to map the desired message

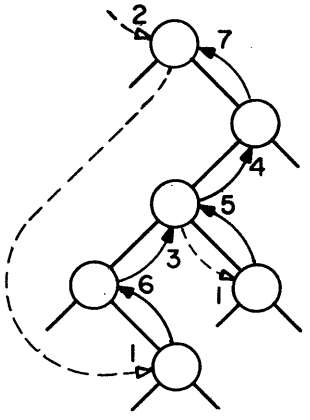
sequence into the required interpretation sequence. The mapping is carried out by the precedence string semantic parse of the first-pass structure composed by the parsing algorithm. We now consider this mechanism in more detail.

F. THE FIRST-PASS STRUCTURE

The final modeling and analysis phases of translation are accomplished by means of an operator following the precedence string in the first-pass structure representation of a statement. The operator composes atomic and molecular actions from the semantic package into an overall complete act of comprehension. It is the job of the programmer to utilize the facilities of the AEDJR System to set up the parsing controls for the First-Pass Algorithm so that the proper sequence of operator actions will ultimately take place. There are many possible parses for a given input string, each of which would supply different data to the second-pass operator. It is the job of the programmer to use the "remote control" features of language-definition to make sure that the final translation is correct for every possible input string. Just as the analyst designing the semantic package had to consider entire families of problems and thereby took a different approach than the average problem-solver, the programmer setting up the language must consider entire families of statements to ensure that all possibilities are covered correctly.

In order to appreciate the job of the programmer in selecting the appropriate first-pass structure for each form of statement, we first consider the "precedence follower" mouse algorithm which corresponds to the first-pass structure. The First-Pass Algorithm which is set up and controlled by the AEDJR command languages is the growth algorithm and the Precedence Follower is the mouse algorithm which forms the backbone of the modeling and analysis phases of the AED Approach. The Precedence Follower mouse follows a minor precedence component (shown by dashed lines) if one is present and has not been followed previously, and otherwise it follows the major precedence, component, (shown by solid lines). At each step along the precedence string, the mouse algorithm also computes an integer code which indicates the local structural context of the current node in the first-pass

structure. This code indicates whether the left and right context are atomic or not, the direction from which the node was encountered, and whether the node represents a "modifier" or not. The codes currently in use are indicated in Fig. 14.



CODE	LEFT	RIGHT	HOW	FROM	MODIFIER
1	ATOM	ATOM	MINOR	——	——
2	ATOM	NONATOM	MINOR	——	YES
3	NONATOM	NONATOM	MAJOR	LEFT	——
4	NONATOM	ATOM	MAJOR	LEFT	——
5	NONATOM	NONATOM	MAJOR	RIGHT	——
6	ATOM	NONATOM	MAJOR	RIGHT	NO
7	ATOM	NONATOM	MAJOR	RIGHT	YES

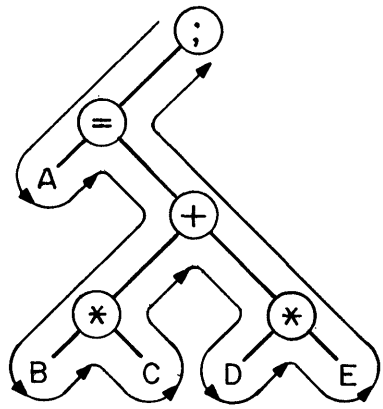
Fig. 14 Contex Codes of the Precedence Follower

The table control for the generalized processor represented by the mouse takes the form of a selector function supplied by the programmer which is called by the mouse once for each step along the precedence string. The mouse supplies two arguments to the selector function: (1) a pointer to the current node in the first-pass structure, and (2) the numerical code representing the local structural context. Using this information (including any or all of the data obtainable from the first-pass structure node as well as any global variable settings), the selector function chooses an appropriate atomic or molecular function from the semantic package and executes it. Once the execution is complete, the selector function returns control to the mouse, which then takes the next step along the precedence string, repeating the process. In this way the modeling, analysis, or control actions of the selected atomic and molecular functions carry out the complete act of translation in an orderly and efficient fashion.

It was mentioned above that a given input string may have many possible parses. On the other hand, not all possible binary tree and precedence string combinations can be generated by the current

First-Pass Algorithm. It is the job of the programmer to determine, for a language which is satisfactory to the designer, what is the "proper" parsing of any given expression to match the semantic package prepared by the analyst.

In order to tell whether a given binary tree can be generated by the current First-Pass Algorithm, there is an elegant print algorithm which operates as shown in Fig. 15. "Given a binary tree, scan around



A=B*C+D*E;

Fig. 15 The Print Algorithm

the tree counterclockwise. Whenever you go around a symbol, print it, and whenever you encounter an operator from below, print it." As Fig. 15 shows, the input string "A=B*C+D*E;" is thereby regenerated.

Figure 16 shows the possible trees that can arise from an input string containing from one to five vocabulary words, if the symbols and vocabulary words are placed in each tree in a manner which satisfies the print algorithm. A number of possible parses for 0, 1, 2, 3, ... words is given by the sequence 1, 2, 5, 14, 42, ...,

$$P_n = \sum_{i=0}^{n-1} P_i * P_{n-i-1}, \ldots$$

This is a strange series similar to the Fibonacci series and may be compared with the sequence of factorials 1, 2, 6, 24, 120, ... which gives

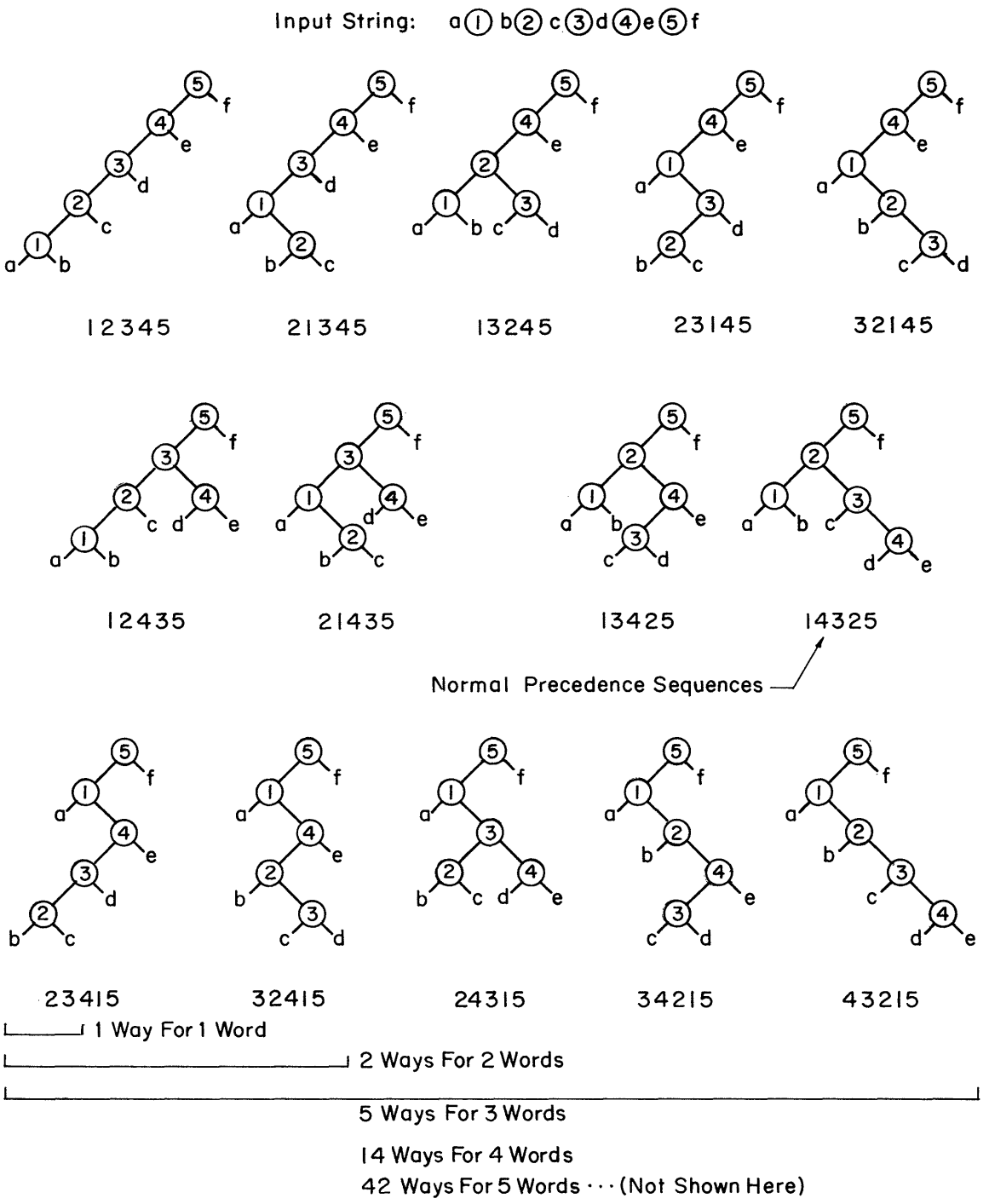


Fig. 16 Possible Parsings

the number of possible permutations of the corresponding number of vocabulary words. Thus although only a subset of the possible syntactic trees are available, there are plenty to choose from. The deciding criterion among the possibilities for a given statement lies in a very strong relationship between the precedence string and the syntactic tree. This is the "remote control" handle by which the programmer controls the mapping of the message sequence into the execution sequence required by the semantic package.

If we refer back to Fig. 13 (in Chapter V) we notice that the precedence string of that example may be described in the following way: "Evaluate the syntactic tree from left to right and bottom to top." In other words, as the example illustrates, we always evaluate the most nested expression first, preserving the left-to-right sequence of the input string. This is called the normal precedence sequence, since it is directly derivable from the syntactic tree. For simple languages, the normal precedence sequence gives the proper sequence for evaluation. With the normal precedence sequence in mind, examine Fig. 16. If we consider the subexpression consisting of the first four vocabulary words (vocabulary word number 5 always serves as the statement terminator in Fig. 16), we notice that the first parse gives strict left-to-right execution sequence, 1234, and the last gives strict reverse execution sequence 4321. The remaining 12 parsings give all possible combinations of left-to-right and right-to-left execution subsequences which are possible if the Print Algorithm is obeyed. In setting up a given language, the proper parse is the one for which normal precedence gives the appropriate mapping from the input sequence to the required interpretation sequence.

Normal precedence is adequate only for the simplest of languages in which context plays no role. The current First-Pass Algorithm provides one more feature called modifier precedence which is beautifully simple and provides a mechanism which is strong enough to satisfy the needs of virtually all artificial programming languages. This mechanism is brought into play by declaring a vocabulary word to be a "modifier", causing the precedence string to be altered so that the modifier word always occurs twice on the precedence string bracketing a context-dependent phrase. An example of this is shown in Fig. 17.

These, then, are the tools which the programmer has at his disposal. Since the first-pass structure itself is composed by the First-Pass Algorithm, the actual controls which the programmer uses in

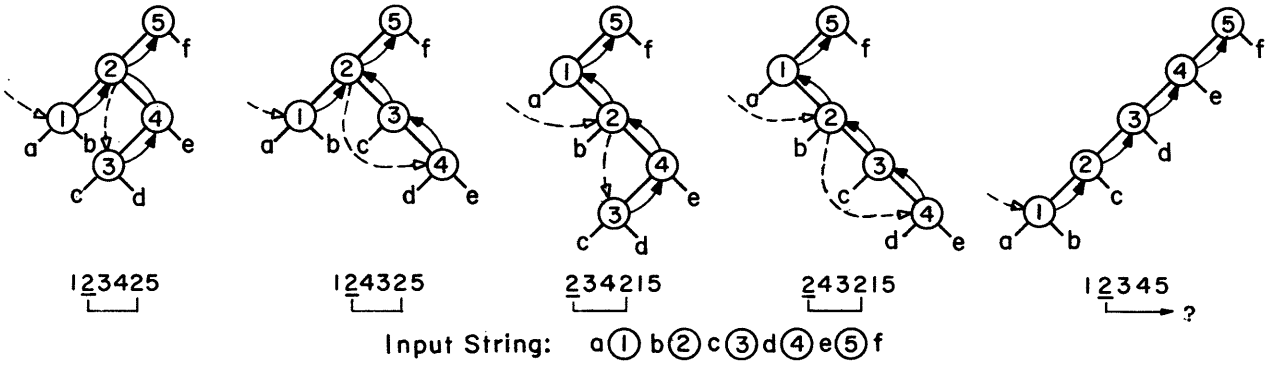


Fig. 17 Modifier Precedence

language definition are still one further stage "remote" from the final selector function which calls on routines from the semantic package. But the programmer is skilled in their application and the debugging facilities of the AEDJR System assist greatly in the language definition process. If the analyst has done a good job at achieving modularity, flexibility, and completeness in the semantic package design, the programmer and designer may work out a wide variety of language forms, both verbal and graphical, all of which are amenable to correct translation.

G. THE ROLE OF THE DESIGNER

Once the designer-programmer-analyst team has completed its first task of creating an operational initial plateau system, further elaboration and evolutionary development are merely natural extensions of the system-building task. In this phase, the designer once again assumes the primary responsibility, but in a much more effective way than at first, for he now knows well the capabilities of the analyst and programmer and their system-building tools. If the macro preprocessing and graphical capabilities have been incorporated, the designer-user and his other friends may, themselves, augment the features of the initial plateau and personalize it to suit their own work

habits. Further true extensions beyond the plateau into successive plateaus are possible by the addition (by the analyst) of new subpackages to the semantic package of the system. If a good job has been done on the initial design, these extensions and elaborations will not drastically change the character of the initial system but will appear to be natural and desirable additions of a maturing system to the designer-users.

One feature of AEDJR parsing which plays an important role in this regard is called phrase substitution. Since all parsing control depends upon the "type" of syntactic units, it makes no difference whether the right context of a given vocabulary is an atomic symbol or an entire first-pass structure, as long as the type of the result is the same. It is no more difficult for the system to handle a sentence such as,"The boy who lives in the green house on Smith Avenue who has red hair and an older brother serving in the army has a toothache", and "John has a toothache". Thus additional sublanguages may be added to the original language to permit elaborate descriptions of quantities which previously existed only as atomic symbols, without causing drastic upheaval in the system. As long as the phrase structure represented by the mouse processing the precedence string is used as the primary control medium, it is assured that no matter how complicated or elaborate a given phrase may be, the result will be indistinguishable from that which would be obtained from an atomic value. Thus semantic package routines may be written to treat only the simplest fundamental cases, and the systematic structure of generalized parsing and precedence following operations will automatically allow the composition of arbitrarily elaborate complex cases in an orderly fashion. Even highly specialized user-oriented languages can be economically very rich in possibilities.

CHAPTER VII

DISPLAY HARDWARE

From the beginning, the MIT Computer-Aided Design Project has been interested in permitting the use of on-line displays for man-machine interactions. In addition to the theoretical, programming, and system-building studies which have been the subject of the preceding chapters, the Project has featured a pioneering activity in both the hardware and software aspects of what is now known as computer graphics. This and the following chapter outline Project accomplishments in these fields.

A. HISTORY

The interest and experience of the Electronic Systems Laboratory in computer displays predates the Computer-Aided Design Project by several years. Beginning in 1953 the Laboratory had used the display facilities of the Whirlwind Computer in sophisticated on-line control of a large-scale data-reduction system. In conjunction with this work, the first tracking program for free-hand graphical input was written in 1954, an on-line keyboard was added to the Whirlwind facility in 1956, and a full-scale man-machine console incorporating the first usage of a Charactron display tube outside of the Sage System was placed in operation on the 1103 Computer at Eglin Air Force Base in 1957. Three-dimensional cutter path display was one of the earliest features of the APT System development in 1956, and during the formative years which led up to the change of emphasis from APT to computer-aided design, the important role of on-line displays, especially for three-dimensional geometric problems, became deeply ingrained in our thinking.

When work began on the computer-aided design effort, the Whirlwind Computer was being phased out, so that the only display facilities available to the Project were the high-performance display incorporated into the small TX-0 Computer which was transferred to M.I.T. from Lincoln Laboratory, and a rather inadequate output-only

display attached to the IBM 709 Computer of the Cooperative Computer Laboratory. Both of these displays were strictly point-plotting cathode ray tubes driven directly under central computer control. The TX-0 Computer had a light pen and toggle switch registers which could be used for program control, but the small size of the computer limited its use to basic experiments in tracking program logic and similar studies.

In order to obtain a larger facility for experimentation, the Project in 1959 developed and installed auxiliary equipment for the 709 Computer to provide control buttons and enable the use of tracking with the existing scope. This facility, (described in Reference 2) also was used only for basic experiments with a "light button" compiler and similar studies because the point plotting limitation of the display and the relatively slow speed of the vacuum tube 709 Computer made large-scale experiments infeasible.

With these facilities available for basic experimentation, the attention of the Project in the display area was focused on various in-depth studies of improved hardware facilities. One study, carried out by a visiting IBM fellowship student in 1961, involved the design of a display console system based entirely upon magnetic drum technology, including built-in pen tracking, sub-routine structuring of the display file, and other advanced features. In order to obtain reasonable performance, however, it was necessary to incorporate high-speed circulating register techniques and multiple reading heads to give the effect of increased drum rotation speed. The resulting system, described in Reference 19, was not constructed but the many aspects of the display problem which were investigated provided valuable background information.

Another study carried out at the same time concerned the development of more elaborate facilities for use with 709 Computer. In this study the use of incremental digital techniques (binary rate multipliers) for linear interpolation was taken over from the numerically controlled milling machine technology and was proposed as a feasible method for vector generation (in both this and the previous study, vector generation within the display hardware was of the highest priority to increase the display speeds over that which was possible with

only point-plotting commands from the computer). The resulting system, described in Reference 20, also was not built, however, because it was decided that a more cohesive design would be needed for supporting the experimental needs of the Project.

At this stage, the idea occurred of expanding on the binary rate multiplier concept for two-dimensional line generation to provide three-dimensional line generation and automatic axonometric projection, as well as hardware light-pen tracking. It was decided to design a complete integrated man-machine display console incorporating these ideas for attaching to the 709 Computer at the Cooperative Computer Laboratory. The design of the system was the subject of the Master's thesis (Reference 23) of Robert H. Stotz, now Assistant Group Leader in the ESL Display Group, and the resulting system became known as the ESL Display Console, or "Kludge".

At the time of completion of the thesis study, Project MAC was being organized at M. I. T. for large-scale time-sharing research and the Computer-Aided Design Project was one of the largest cooperating groups which would be using its more modern facilities. The design of the ESL Display Console was therefore modified slightly to attach to the time-shared IBM 7094 Computer of Project MAC. The console was constructed and in operation in the time-shared environment by mid-1964. The more general display interests of Project MAC led to the establishment of the ESL Display Group, which has since that time carried out display research and development for both the Computer-Aided Design Project and Project MAC, as well as other groups at M. I. T.

Details of these display hardware developments have been adequately covered in various Project reports and technical papers listed in the references. The purpose of the present discussion is to describe the general characteristics of the equipment, the underlying motivations and philosophy of approach, and the outlook for the future.

B. THE ESL DISPLAY CONSOLE

The ESL Display Console, shown in Fig. 18, was designed to operate from the Direct Data Connection (Channel D) of the 7094, which has 36 data lines and 10 sense lines in both directions. A 36-bit format was therefore chosen for display command words. Also, the console

Fig. 18 The ESL Display Console

was designed without a memory of its own, and was operated for four years in the MAC time-sharing system in a "semi-buffered" mode in which information to be displayed was set up in a "display list" in the A-core (supervisory) memory of the computer, i.e., a portion of the A-core memory was used for continually rewriting (refreshing) the display. Although this function is now served by a PDP-7 display buffer computer, as will be discussed later, the following description of the mode of operation pertains whether the console connection is to the 7094 channel or the PDP-7 channel.

Each time the display is to be regenerated -- roughly 30 times per second -- a programmable "alarm" clock in the display console interrupts the computer and causes the list to be outputted in the channel mode. Once the channel output command is given, individual words in the list are accessed sequentially as needed by the console by memory "cycle stealing" without interrupting any program which may be running in the computer. Since each word in the display list contains both a data field, and a command field which tells the display system what to do with the data, the "picture" described by the list is displayed without further attention by the computer until the next regeneration time. The hardware capabilities of the display generation system in the console permit the information in the display list to be in a highly coded form, and a small number of display words can generate quite complex displays. Further reduction in the number of display words required results from the use of "sub-pictures", similar to the concept of subroutines in normal programming. Thus, if the same picture element (perhaps a resistor in an electrical circuit, or a bolt in a mechanical assembly) is to appear in various locations on the display screen, its description appears only once in the display list and is called by sub-picture jumps as required. As will be explained later, the use of sub-pictures is aided by the fact that symbol and vector commands are in terms of relative screen coordinates.

The display console is based on a digital incrementing scope designed in collaboration with the Digital Equipment Corporation and identified by DEC as the Type 330. The Type 330 is capable of displaying closely spaced points every 1.5 μs (about 20 times faster than point-plotting displays) when driven in the incrementing mode. The display

generation system which provides the incrementing pulses was built by the Electronic Systems Laboratory, and performs with hardware such display functions as: line and character generation; rotation, translation, and scaling of pictures; edge detection; and pen tracking.* A digital approach was maintained throughout for purposes of accuracy in plotting, flexibility, and for ease of implementing the desired functions. In designing the computer interface, control logic, and display generation system, considerable thought was given to the tradeoffs between hardware and programming in order to minimize display word accesses on the data channel and reduce the computational load on the computer, and it is felt that the hardware, although quite complex, more than pays for itself in computer time saved.

A block diagram of the ESL Display Console is shown in Fig. 19. The following paragraphs describe the essential features of the display generation system. Note that two coordinate frames of reference are established: x, y, z referring to the fixed coordinate space in which the computer "knows" a figure being displayed, and h, v referring to the actual display coordinates. This distinction is necessary because of the coordinate transformations performed in the console.

1. Line Generation

The Type 330 incrementing scope contains horizontal and vertical buffer registers (h and v in Fig. 19) which directly control the scope deflection amplifiers. These can be set by parallel bit transfers from the computer in the normal point-plotting manner. Their unusual feature is that they can also be incremented or decremented at 1.8 μs intervals by pulses from the display generation system. The line generator uses a three-channel binary rate multiplier (BRM) to convert the signed 10-bit Δx, and Δy, and Δz vector components contained in a line-generate command into synchronized x-rate, y-rate, and z-rate pulse trains. The pulse rates in the channels are such that in the BRM

*DEC also offers the Type 340 and 338 Incrementing Scopes, derived from the Type 330, which are complete display systems with line and character generation, but which do not have all the features described here.

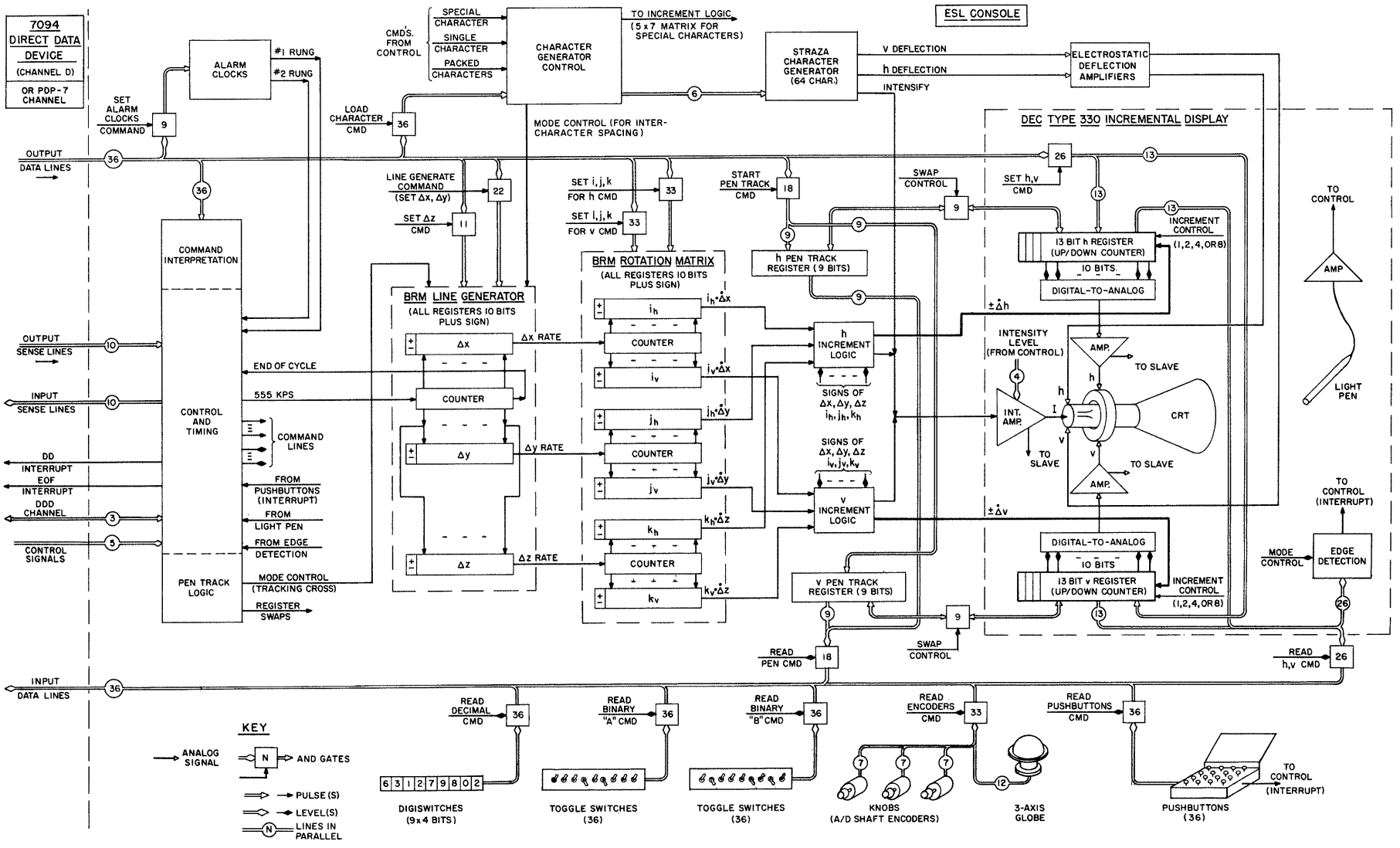


Fig. 19 Block Diagram of ESL Console

generation cycle, each pulse train contains a number of pulses exactly equal to the numerical value of the corresponding vector component. Thus the vector components are still digital in nature, but have been converted to an incremental form without loss of accuracy.

Assume for the moment that the x-rate and y-rate pulse trains were applied directly to the incrementing inputs of the h and v deflection registers. To start plotting a picture element, the computer first sets the h and v registers to the desired initial set point (x, y), and then sets the registers of the binary rate multiplier to the desired Δx and Δy values for the terminal point of the first line. Line plotting proceeds automatically, with the scope being intensified for 0.5 μs each time the buffers are incremented (unless a no-plot bit is given in the line command). At the completion of the line generation cycle, the h and v registers contain the terminal point for the line, and each succeeding line starts at the terminal point of the previous one (unless a new initial set point is given, which defines a new picture element). Blank vectors (not plotted) can be used to connect non-contiguous parts (alphanumeric labels, etc.) of a picture element. Note that if the computer wants to move a picture element made up of these connected vectors around the scope face, all it has to do is to change the single initial set-point command in the display list for the element. It need not alter in any way the remainder of the data for the element.

Actually, the x-rate and y-rate pulse trains are not applied directly to the h and v registers as in the above, but are first passed through a rotation matrix as described below.

2. Display Rotation

Rotation of displayed figures is an essential feature for many applications and would normally require the computer to recalculate the data for the entire display for each new angular position. This process has been greatly simplified in the display console by further incremental manipulations. As shown in Fig. 19, a group of additional binary rate multipliers (called the rotation matrix) has been placed between the line generator and the scope deflection registers, which permits multiplication of the Δx, Δy, and Δz pulse trains by numbers set by the computer. Considering a vector A as being specified by its

three components A_x, A_y, and A_z in the fixed coordinate space, the components of the vector in the display coordinate space are determined by the following relations:

$$A_h = A_x \cdot i_h + A_y \cdot j_h + A_z \cdot k_h$$
$$A_v = A_x \cdot i_v + A_y \cdot j_v + A_z \cdot k_v$$

where i_h, i_v, j_h, j_v, k_h, k_v are 10-bit numbers set by the computer in the six rotation matrix registers:

If the (i, j, k) matrix settings are the components of the unit vector specifying a desired rotation between the two coordinate systems, the outputs of the matrix will be the components A_h and A_v of the vector in the rotated coordinate system, and these will be the inputs to the h and v registers of the scope. Thus to create an arbitrary projection of a complex three-dimensional picture element made up of a sequence of connected vectors (which can include unplotted ones), the computer need only change two commands at the head of the display list for the sequence, which set the six rotation matrix registers. The rotation will be relative to the initial set point for the sequence. Arbitrary centers of rotation can be established by including an initial blank (unplotted) vector in each picture element. The matrix is also used to change the size of displayed figures by applying a common scale factor (where 2^{10}-1 represents full scale) to the matrix settings.

Figure 20 shows the format of a display list for the ESL Display Console. The "header" consists of three words which control the

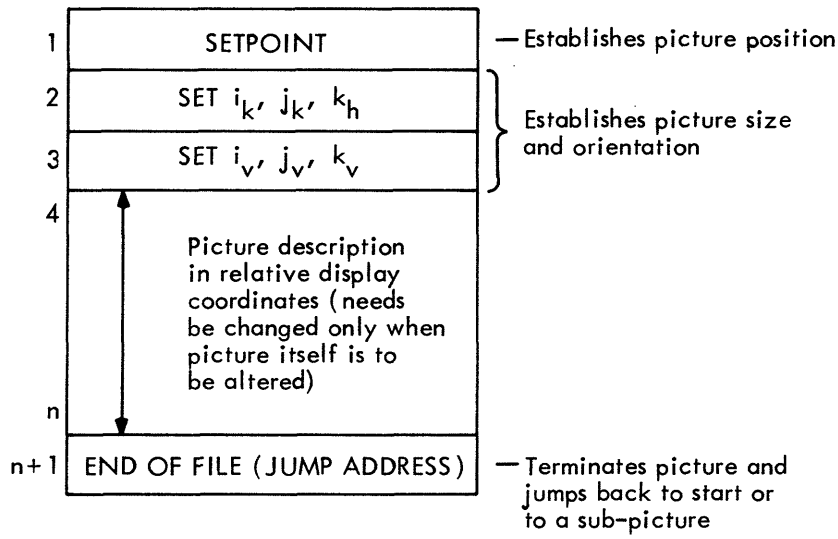


Fig. 20 Typical Display List

position on the screen, the size, and the viewing angle for the picture or picture element defined by the body of the list. The list is terminated by a jump instruction. This saves the computer from having to compute end addresses or word counts for the data channel, and also facilitates picture sub-routining, or "sub-pictures". Usually, the header information is changed each "frame" (1/30 second), to cause smooth translation, rotation, and scale changes of a complete picture. In other cases, different rotations or scale changes are applied to parts of an overall display list to create dynamic mechanisms, show a single picture element in several orientations simultaneously, etc. The important point is that for these manipulations, the computer never has to alter the display data itself, only the matrix settings for the console and the initial reference point (setpoint). This effects a substantial reduction in computer time.

3. Automatic Edging

A problem in programming most display systems is "edging", i.e., masking display data to fit on the display screen. Many displays will "wrap-around", i.e., display modulo the maximum screen coordinates. Also, many vector generators require modification of end-point coordinates in order to display part of a vector which extends beyond the visible area. Thus, the computer must usually be continuously concerned with preventing any portion of a dynamically changing picture from overflowing the display area. Since in the ESL Console the display list information is incremental in nature, and rotations, scale changes, and translations are performed external to the computer, such monitoring of display data would be a formidable computational task, and would negate much of the gains of external display generation and manipulation.

In the ESL Console, the horizontal and vertical display registers were specified to be each 13 bits in length, with only the low-order 10 bits being displayed. Thus the "console dimensions" are eight times those of the visible display area, and lines can be incremented off the display area without wrap-around and without loss of data. With the enlarged buffers the computer need not prevent overflows of the display area, but it can be informed of overflows by programmable edge

detection circuits which, when enabled, cause an interrupt either at the edges of the visible display area or at the edges of the larger buffer area. The computer can also read the display h and v registers at any time it desires. Thus it can use the console as a special-purpose "geometrical masking computer" to monitor display operations, and provide the information needed for modification of a display list when desirable or necessary.

4. Automatic Light Pen Tracking

Another important function of a display system is graphical input, and a very useful technique for accomplishing this is light-pen tracking. The usual programmed tracking (periodically displaying a tracking pattern, testing light-pen response, and calculating new coordinates, all under program control) typically requires about 10 percent of a computer's time for adequate performance. In order to reduce this drain on computer capacity, the pen tracking was made an automatic hardware function in the ESL Console. When the computer enables pen tracking (by a pen-track command added to one display list cycle), the console is placed in a mode in which it automatically stops the display process at regular intervals (currently every 10 ms) and inserts a tracking cycle.

The pen track logic utilizes the line generator to generate the four fixed-length arms of the tracking cross at the (h, v) location determined by the "pen location" registers in the console, which are set initially by the pen-track enable command. Based on pen response for each point in the tracking cross, four points on the field-of-view circle of the pen are found, and from these Δh and Δv movements of the pen since the last tracking cycle are determined and used to update the pen location registers. At the completion of each tracking cycle, the display process automatically resumes at the point where it was stopped. Thus pen location is always available to the computer in the pen location registers. Each tracking cycle takes 200 microseconds, and since there are 100 tracking cycles per second, the tracking function uses only two percent of total display time.

The automatic pen-tracking hardware described above was essential for operation of the ESL Display Console directly from the

7094 time-sharing system, since programmed pen tracking would have placed an unacceptable demand on CPU time. However, it is now apparent that future displays operated from a central time-shared computer facility will usually have a satellite display-buffer computer (such as the PDP-7 now used with the ESL Console) and that there may be sufficient CPU time in the satellite computer to permit programmed tracking -- eliminating the need for tracking hardware. This is an example of the hardware-software tradeoffs which must be taken into consideration in any particular situation.

There have also been a variety of new input devices developed since the ESL Console was designed that can substitute for the light pen, and in some applications provide superior performance. Typical of these is the Rand Tablet.* This unit, which provides continuous measurements of the x-y position of a stylus as it is moved over a writing surface (independent of the display operation) is now commercially available, and similar devices are offered by other manufacturers. The light pen is still a simple but powerful tool, however, and the choice of input device and its mode of operation is primarily a function of the intended application.

5. Character Generation

Display systems intended for output of textual material are usually designed to operate in "typewriter format", i.e., with inter-character and line spacings fixed as a function of character size. This is a reasonable approach for many uses. However, it imposes a rigid format on data sequence and possible character positions that would be incompatible with the display manipulation system of the ESL Console. Thus, a more flexible system was devised, making use of the line-generation system to provide programmable character locations and spacing. Character strings can be spaced any amount in any direction. Vertically spaced strings, for example, have been found convenient for labeling the ordinate of graphs.

---

*"The RAND Tablet: A Man-Machine Communication Device", Davis, M. R. and Ellis, T. O., AFIPS Conference Proceedings, Volume 26, Part I, Spartan Books Inc., 1964, pp. 325-332.

The stored character generator is a standard commercial unit (Straza Industries Type 11-64), which provides a font of 64 characters matching that used on the KSR 35 Teletypes of the Project MAC Time-Sharing System. Characters are selected by 6-bit codes which can be given in two ways: (a) a single character with its h, v location per word, and (b) character strings packed six to a word. The packed-character command includes character spacing information, and places the console in a mode wherein subsequent words are interpreted as six characters each (except for the first word following the command, which contains five characters plus a count of the number of characters in the string -- necessary to terminate the mode). In the packed-character mode, the initial h, v location is established by the contents of the h and v buffers resulting from any previous display operation. Thus labels can be "attached" to picture elements and will remain so during picture manipulations.

Since 64 characters is an insufficient repertoire for many purposes (it does not provide lower-case letters, for example), the console also has a "special-character" mode which generates arbitrary characters under program control. The special-character command causes the line-generation logic to draw a 5 x 7 matrix, and the points in this matrix to be intensified are determined by 35 bits in the following computer word. Again, strings of characters can be given (one character per word) with arbitrary spacing, and the 5 x 7 matrices can be stacked vertically and horizontally to form larger matrices for large symbols, such as integral signs, etc.

6. Manual Inputs

In addition to the light pen described previously, the console has a number of other forms of manual input necessary for communication with the computer. As shown in Fig. 19, these include switches, pushbuttons, and knobs. Perhaps the only unusual feature of these input devices is the philosophy of their use in the console-computer system, which is explained briefly below.

Swtiches have long been used for manual input and the console has 72 toggle switches (two 36-bit words), and 9 decimal switches (one 36-bit word). These are convenient for various purposes, such as

giving a picture an identifying number, calling for a previously stored picture, program control, etc. The switch registers can be read only by the computer, and have no wired connection to any operation in the console. Also, no action of these switches has any effect on the computer unless it is programmed to look at them. For purposes of signaling the computer that some action is desired, a box with 36 pushbuttons is provided. Pushing any button causes a computer interrupt. The computer then jumps to a subroutine which reads the push-button register (which again has no wired console function) and determines which button was pushed. Depending on the conventions established by the programmer, particular buttons can be assigned meanings, such as "read the decimal switches", "start the light pen tracking mode", etc.

For purposes of display manipulations such as rotation, translation, and scaling, switches are an inconvenient form of manual input and it is now becoming common to provide some form of continuous input in real-time consoles. (Note that real-time continuous input has long been a much-touted advantage of analog computers, but is now also available in digital systems.) In the ESL Console, continuous input is provided in two forms -- "knobs" and a three-axis "globe". Three knobs, each driving an analog-to-digital shaft encoder, permit convenient control of such functions as scaling, translation, rotation about an axis, etc. Individual knobs are very inconvenient, however, when two or more actions must be coordinated, such as rotating a figure about more than one axis simultaneously. Thus a three-dimensional input device was devised. This unit, called the "globe" (shown in front of the display in Fig. 18), has spring-loaded limited rotation about three mutually perpendicular axes. Each axis has a simple cam and microswitch encoder, with three discrete codes each side of neutral.

The globe is usually used as a three-step rate control, with the rate scaling entirely under program control, and provides a very convenient means of maneuvering a three-dimensional object into a desired orientation. Coupled with the ability of the console hardware to display a new projection of an object each pass through a display list, the smooth apparent motion provided by rate control gives a striking visual effect which enables instant visualization of the three-dimensional shape of a displayed object, even though no perspective is involved.

This "motion effect" was unanticipated in the design of the console, since it was thought that perspective projections, or even stereoscopic views would be necessary for three-dimensional visualization.* Stereo and perspective experiments have been conducted, but since no user of the equipment has felt a strong need for such facilities, these proposed additions to the console hardware have not been implemented.

Like the switches and buttons, the knobs and globe are read only by the computer, and have no wired console function. Programmed interpretation of these devices, keyed to manual input rates, gives maximum flexibility at the cost of a very small amount of computer time. For example, some users have programmed rotation to be under the control of the knobs (direct angular control about each axis), and others have used the globe for rate control of picture scale and translation.

7. Command Summary

In summary, the ESL Console may be looked on as a special-purpose graphic computer, with instruction words similar to the "immediate" commands found in most general-purpose digital computers, i.e., the commands refer only to data contained in the same word (or words immediately following). Since the console was designed for use with the 7094 computer, the same command format was chosen as is used in the 7094, i.e., the Prefix (bits S-2) and the Tag (bits 18-20) define the command. A typical command word (to plot a line with Δx and Δy components) is shown in Fig. 21. (If a line also has a Δz

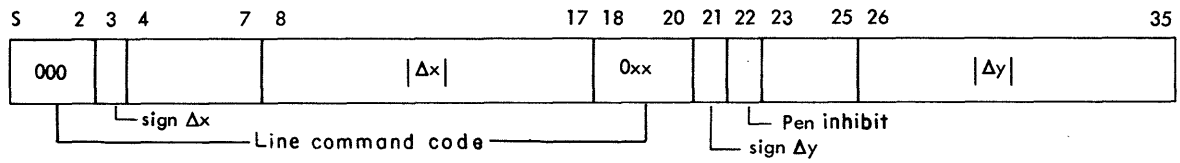


Fig. 21 Typical Display Command Word

component, this must be specified in a separate command preceding this line-plot command.)

*See Reference 23.

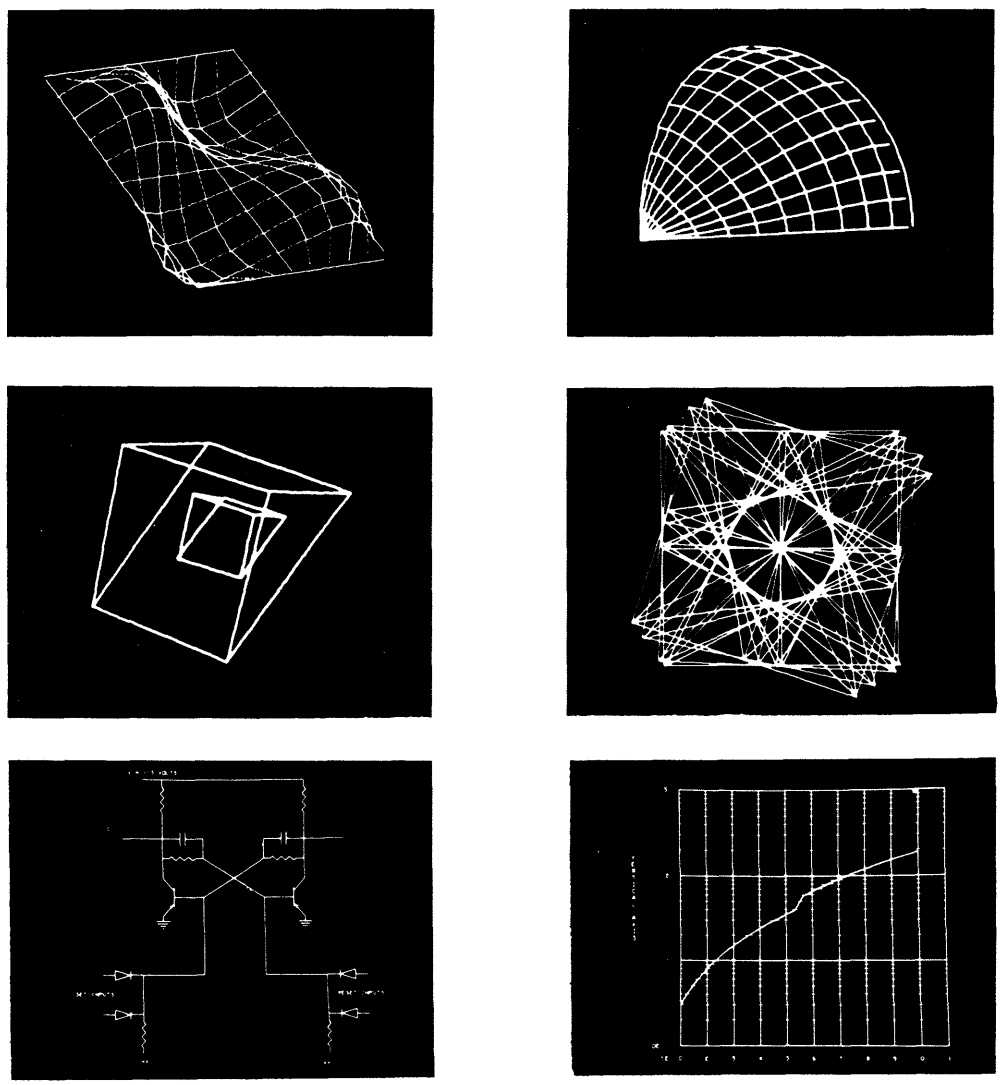
The command repertoire of the ESL Display Console is shown in Table 1. Note the console actually includes two display stations called "Master" and "Slave". These are always deflected in parallel, but the computer can control which one (or both) is to be intensified at any given time. Two sets of manual input devices are provided, and two unrelated display problems may be run simultaneously by alternating data for the Master and Slave stations each frame. This "time sharing" of the display generation equipment of course tends to increase flicker for each user, but this has not been a problem except when both users have very complicated displays. For more detail on the command structure and other details of console operation, References 31 and 32 should be consulted.

The specification summary for the ESL Display Console is shown in Table 2. Figure 22 illustrates typical displays from various graphic applications programs.

C. DISPLAY BUFFER COMPUTER

1. Background

In the original design of the ESL Display Console in 1963, it was recognized that a central computer should not have to be bothered with repetitively supplying data for purposes of display maintenance, i.e., continuous rewriting of a picture at a flicker-free rate. The solution, of course, is to provide a buffer memory in the display system itself. Many buffered displays had been built over the previous decade, mostly using drum memories or special-purpose sequentially-addressed core stacks. Memories of this type are relatively inflexible, however, and do not fit in with the dynamic computer-display interaction desired for computer-aided design. It was therefore decided to build the ESL Console initially without a memory, and to buffer display data in the core memory of the central computer until such time as proper requirements for a buffer memory could be determined. At the same time, the console was designed to minimize computer load by building in specialized hardware to perform many normally time-consuming computer functions, such as picture rotation, pen tracking, etc.



The ESL Console is a specialized computer which automatically converts three-dimensional drawing commands into arbitrary two-dimensional projections. Real-time rotation, translation, and scale change are possible even in time-sharing. Light-pen tracking is fully automatic, and either picture elements or character information may be displayed.

Fig. 22 Typical Displays on ESL Display Console

Table 1

COMMAND FORMAT FOR ESL DISPLAY CONSOLE

7094 Bits							Function	Option
Prefix			Tag					
S	1	2	18	19	20	4		
0	0	1	X	X	X		Set ΔZ component of line	
0	0	0	0	X	X		Plot line ΔX, ΔY, (ΔZ)	Visible
			1	X	X		" "	Invisible
0	1	0	0	S_1	S_0	0	Character Generation	Packed Mode
			0	X	S_0	1	" "	Special Symbol Mode
			1	S_1	S_0		" "	Unpacked Mode
0	1	1	0	0	0		Plot Set Point X, Y	Visible
			1	0	0		" "	Invisible
			0	0	1		Load Control Word C	Set various console parameters
			1	0	1		Load Control Word F	Incremental beam control
			0	1	0		Start Light Pen Track	Master or Slave
			1	1	0		Stop " "	" "
			0	1	1		End of File	Jump Address
			1	1	1		Not Used	
1	0	0	0	A_1	A_0		Set Alarm Clock	Slow
			1	A_1	A_0		" "	Fast
1	0	1	X	X	X		Set i, j, k, for h	
1	1	0	X	X	X		Set i, j, k, for v	
1	1	1	X	X	X		(Reserved to set i, j, k for depth coordinate, d)	

Notes: S_1 and S_0 specify character size (4 sizes available).

A_1 and A_0 specify which of four Alarm Clocks, (only two installed).

X indicates "don't care".

Table 2

Specification Summary for ESL Display Console

Input Power	115 ± 10 volts, 60 cycles, single phase at 18 amps.
Active Scope Size	9 3/8 inches by 9 3/8 inches containing 1024 points by 1024 points (one scope increment = 0.009 inches). P7 phosphor.
Memory	None (operates from display lists stored in the computer memory, and accessed through the Direct Data Connection).
Clock Rate	Normal 555.55 KC (1.8 μsec between clocks). Slow 69.44 KC (14.4 μsec between clocks, used for experiments with storage CRT's).
Line Plotting	
Line Plotting Rate	a point each clock (1.8 μs). This point can be a step of 0, 1, 2, 4, or 8 scope increments in ± Δh and in ± Δv (0, 1, 2 for lines to be rotated).
Line Length	0 to 1023 increments in ± Δx and in ± Δy without magnification. 0 to 2046 increments by steps of 2 in ± Δx and ± Δy with magnification. Thus, Δx and Δy require 10 bits plus sign each.
Random Point Plotting	
Point Plotting Rate	a point every 40 microseconds
Point Plotting Range	2^{13} = 8192 horizontal and vertical positions. Of this only 2^{10} = 1024 will appear on the scope. h = 0, v = 0 is center of screen.
Straza Symbol Generator	
Symbol Code	6 bits to produce one of 64 symbols matching KSR-35 Teletype.
Symbol Size	0.1, 0.15, 0.23, or 0.35 inches high.
Symbol Plotting Rate	a character every 20 μsec.
Special Character Generator	
Symbol Code	35-bit code to produce any symbol on a 5 x 7 dot matrix.
Symbol Size	0.14, 0.28, or 0.56 inches high.
Symbol Plotting Rate	a character every 72 μsec.
Refresh Timing	Programmable "Alarm Clocks" to cause real-time interrupt between 50 microseconds and 6.4 milliseconds by 50 microsecond increments (fast clock), or between 1 millisecond and 128 milliseconds by 1 millisecond increments (slow clock).

The console has been running in this "semi-buffered" mode on the Project MAC Time-Shared 7094 (CTSS) since January, 1964. Although operation has been quite successful from a functional viewpoint, there have been some minor problems. First, there is a mutual interference problem which has been of increasing concern as the load on the Project MAC computer increased: (1) the five to thirty percent of 7094 memory cycles (data accesses plus interrupts) taken by the console on typical graphics is out of proportion to the average two percent or less taken by a user at a Teletype station, and (2) the new high-speed 7094 drum system installed in 1966 blocks display data whenever a CTSS memory swap is occurring, which causes the display to "blink" every few seconds. Second, the allocation of display list space in the A-core (Supervisor) of the 7094 has been limited to 1200 words. This is divided between the two console stations according to user requirements, but many users have had to sign up for both console stations in order to get sufficient display space for a complicated picture. These problems, plus the clear fact that display buffering is a requirement for any displays located remotely from the computer (such as would most probably be desired in any industrial application), caused interest to be intensified in solving the display buffer problem.

Experience in operating the ESL Console in the semi-buffered mode described above, led us to the conclusion that in order to provide the desired flexibility in real-time interactions at a remote display console, the display buffer system should in fact be a small general-purpose computer. This conclusion (now shared by many other organizations) was strengthened by the commercial introduction during this same period of quite powerful small computers in the $20,000 to $50,000 price range, and the clear indication that even these low prices would drop in the near future. Thus the Display Group recommended in November, 1965, that a small computer be purchased by Project MAC for buffering the ESL Console. The particular computer chosen was the Digital Equipment Corporation PDP-7, an 18-bit machine with convenient input-output provisions and a 1.75 μsec memory cycle. The PDP-7 circuitry is, of course, directly compatible with the ESL Console, which was constructed with DEC logic modules. Also, the 18-bit word length permitted convenient packing of the 36-bit 7094 words into

two PDP-7 memory locations. The choice was also influenced by a desire to be software-compatible with other groups at M. I. T. and elsewhere that had already chosen the PDP-7 for display buffering, and were interested in cooperative efforts on design of communication formats, executive routines, etc.

2. Design Considerations

In designing the PDP-7 display buffer system for the ESL Console, it was of course desirable to avoid if possible the obsoleting of existing software for the ESL Console (both system software and user programs). At the same time, a solution was sought which had as much generality as possible, i. e., one which could serve as a test bed for developing techniques and procedures for future buffered display systems.

In a buffered display system, there are two data-transfer interfaces to consider: that between the central computer and the buffer computer, and that between the buffer computer and the display. The latter needs to be a high data rate connection with almost autonomous access by the display controller to the buffer computer memory. Thus, this will usually be a parallel word-transfer connection via a data channel of the buffer computer. The connection between the two computers, however, is now freed from the high data rates and real-time service demands of display maintenance, and choice of data rate and type of connection is open to cost/performance tradeoffs. For example, the primary consideration in choosing the bandwidth of the link between the two computers is how long a user is willing to wait for a new picture after requesting it, or stated another way, how much is he willing to pay for a certain maximum delay in system response.

Studies to date indicate that a serial connection with a data rate of 50, 000 bits per second will be a good choice for most applications, since this will permit sending data for even the most complicated picture over the link in a second or two. Also, this type of link is relatively economical and easy to implement, and is an available service from common carriers, should one wish or need to operate a display through such services.

Although a serial communications-type connection appears to be the proper one for most applications, the decision was made to initially forego any possibility for remote operation and connect the PDP-7 and 7094 on a channel-to-channel basis. By building a special interface which makes the PDP-7 look to the 7094 like the ESL Console, and to the ESL Console like the 7094, the majority of existing software was preserved intact, and no hardware changes were required in the 7094 or ESL Console.

It is important to note that this choice, made primarily for expediency, does not violate the general scheme outlined above. If, at a later time, it is desired to convert the link between the PDP-7 and 7094 to a serial communication line, this can easily be done by procuring suitable communications adapters for the two computers, and changing the communications modules in the respective software.

The overall block diagram of the special interface designed to "splice" the PDP-7 in between the 7094 and the ESL Console is shown in Fig. 23. The primary parts of this interface are: (1) the Word-Forming Buffer, which automatically packs 36-bit words from the 7094 and ESL Console into two PDP-7 memory locations (and vice versa); and (2) the Buffer and Interrupt Registers which enable the PDP-7 to intercept and interpret all control, sense-line, and interrupt messages. Except for the standard PDP-7 options shown within the dashed outline, this interface has been constructed in the Electronic Systems Laboratory.

The PDP-7 was ordered with an 8K memory, which was expected to provide approximately 4000, 18-bit words of space for display lists (assuming that executive and real-time programs would require about 4K). Actually the initial programs (described in Chapter VIII) occupy only 3000 words, leaving 5000 words for display lists. This is the equivalent of 2500 words of 7094 memory, twice the amount which has been available in the A-core of the 7094. In general, it is our feeling that a display buffer computer should have about 8K of core. This appears adequate for most purposes, provided that an adequate data link to the main computer is available, and that service response of the main computer is fast enough. There is a tendency on the part of users to want more memory, particularly when the above provisions are not (or cannot) be met. However, memory is the single most expensive

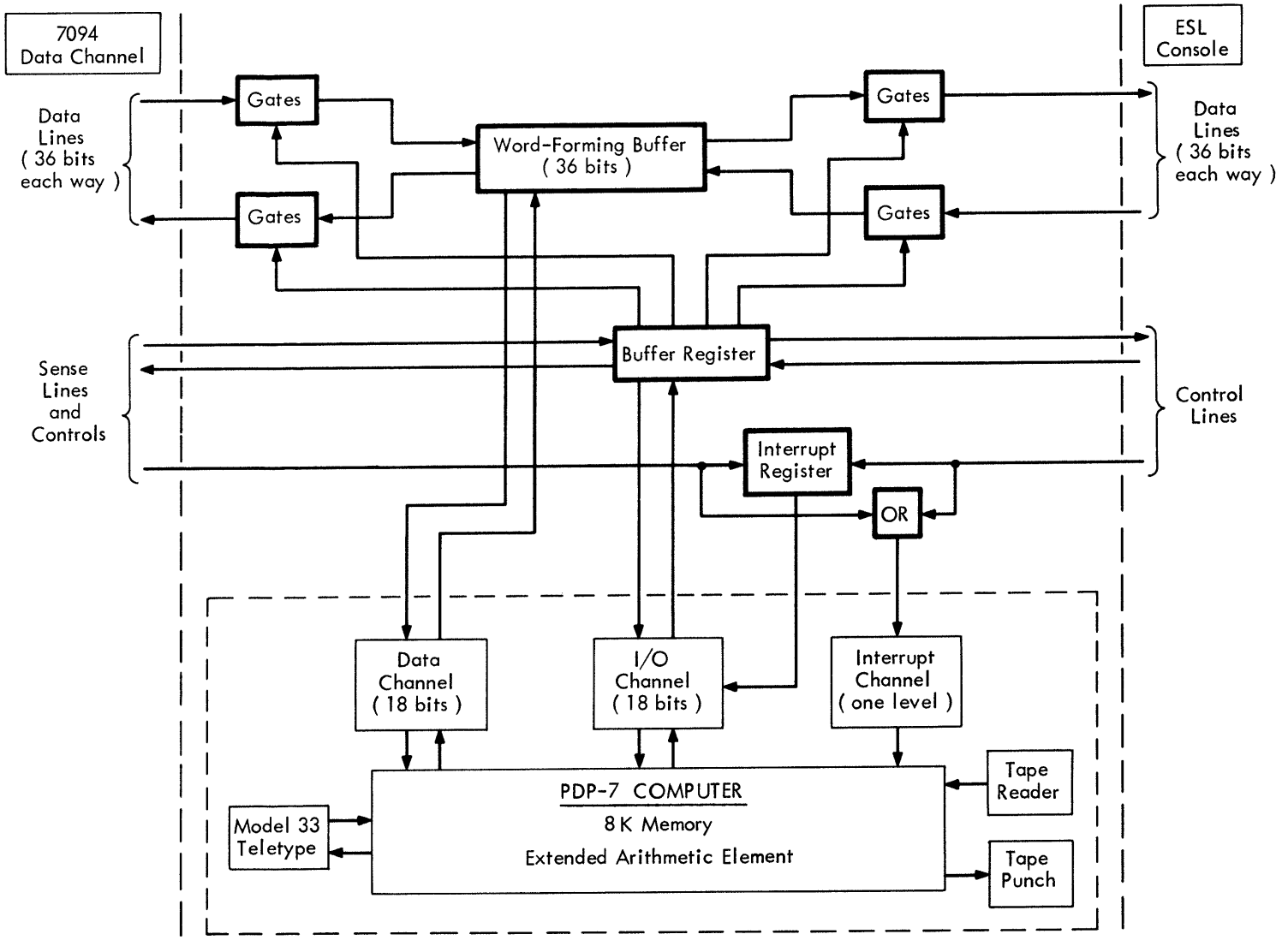


Fig. 23 Addition of PDP-7 Display Buffer Computer between the ESL Console and the Project MAC 7094

item in small computers today, and an optimum system would seem to require a proper balance between total memory capacity and bandwidth of the communication links.

3. Status of the Buffer System

As of the end of the contract (May, 1967), the buffer system was very close to being operational. Information could be blocked back and forth between the 7094 and PDP-7, and the display could be operated from the PDP-7. However, certain combinations of events, particularly simultaneous (or near-simultaneous) PDP-7 interrupts from the Console and the 7094 would hang up the system. This type of problem is difficult to isolate, and it was not until August, 1967, that these problems were overcome and the buffered system released for routine operation.

D. DEVELOPMENT OF LOW-COST REMOTE DISPLAY

Although the low-cost display development reported in this section was primarily supported by Project MAC at M.I.T. through Contract NOnr-4102(01), the hardware and software techniques employed are closely intertwined with those of the ESL Display Console. Also, the design goals were largely drawn from the interactive graphics needs of computer-aided design in a time-shared environment. It thus has been appropriate to include discussions of the low-cost display development in the Interim Reports of the Computer-Aided Design Project submitted in the past, and to include a brief description in this Final Report. The following is based on Reference 109.

1. Background

Experimental computer time-sharing systems now have been in operation for several years. This type of computer service appears ideal for computer-aided design, and from all indications, will soon become widely available. With a few notable exceptions, however, users of these systems must communicate through mechanical teletypewriters of some form (e.g., Teletype, IBM 1050). Although teletypewriters are generally satisfactory as input devices, they are woefully inadequate for computer output due to their slow speed and rigid format. Alphanumeric output is typically at 10 to 15 characters per second, which is well below human scanning speed. Also, input and output of graphic data is

cumbersome, if not impossible. Teletypewriters do have the advantage, however, that they can operate over standard switched telephone lines and can thus be located at virtually any remote site. Also, they are relatively inexpensive, renting for $100-150 per month.

At the other end of the spectrum is the full-fledged graphic display console, such as the ESL Console with its PDP-7 computer-buffer. This type of system provides graphics and text at very high speed, but represents a capital investment of $150,000-200,000, or perhaps $5,000 per month if rented. Such a price level precludes having more than a few displays in an installation, and thus limits graphics access for the average system user.

As a result of the above dilemma, the Display Group set out to try to devise a terminal which could replace teletypewriters at hopefully comparable cost, and provide high-quality alphanumeric and graphic display at higher speed (but not necessarily the speed of regular display systems). The research effort has resulted in a unit called ARDS (for advanced remote display station). The design goals, the implementation chosen, and a description of ARDS are given in the paragraphs below.

2. Desired Characteristics

In considering the requirements for an improved time-sharing terminal with graphics capability, the following "specifications" were set down in early 1965. It was realized that these were beyond the state of the art at that time (particularly the cost figure), but it was felt that they were reasonable as goals for a development program, subject of course to engineering compromise.

First of all, it was concluded that the new terminal should be a CRT display device capable of handling characters, points, and lines in a free format. It should also be capable of operating as a stand-alone unit from a standard telephone line as teletypewriters do now, but that it should make use of the full available data rate (today's teletypewriters operate at 130 baud over telephone lines capable of operating as fast as 2000 baud). The desire for telephone-line operation is a result of the fact that the telephone system represents the only communication network widely available to the public, and it appears that its services will be used in any computer public utility. Clustering of consoles about a

central control unit is a time-honored way to bring the cost per terminal down. However, this can be done only in certain environments and is not suitable for a general-purpose time-sharing terminal.

Graphic output, which refers to the ability to draw pictures, is essential to the full development of the potentials of the time-shared computer. When the computer can communicate in pictures, a whole new dimension is added to man-machine "conversations". Lists of numbers become graphs, bridge structures, and electrical circuits. In the vast areas of computer application where the real world is modeled (e.g., computer-aided design, simulation, process control), use of graphics is of particular importance. A highly desirable adjunct to graphic output is graphic input, such as may be effected with a light pen or input tablet. This capability is required to truly "converse" in graphical language.

It is important that hard copy of displayed output be available to the user when he needs it. For most applications, however, it is not required at the local console, and not for all output. At Project MAC, for example, over 90% of the teletype output paper produced by the 150 consoles in the system goes directly into the trash basket. The hard copy that is taken away is generally for record purposes -- a need which could be fulfilled by centralized hard-copy generators. The output paper is occasionally used while at a console to refer to previous data or conversation, but if appropriate system programs can retrieve this information quickly and easily, this "need" for hard copy vanishes. Thus it was concluded that a device which produced only "soft copy" could serve as a terminal.

The display area should be at least 100 square inches, which is about the area of standard 8 1/2 x 11 paper. The resolution should be at least that of high-quality CRT displays today (50 black-white line pairs per inch). This resolution and screen size should permit the display of 4000 characters simultaneously. Many users would be content with less, but anything less than 1000 characters is considered to be inadequate for a general-purpose time-sharing console.

The display should accept and plot new data as fast as it comes over the line from the computer, and the data should be as highly coded as is economically feasible in order to minimize time required to

present new information. If a screen or other device which stores the image for viewing is employed, it should be possible to erase this rapidly (less than 1 second) for display of new information. Since a user may work for hours at a time and he will often want to refer to other papers, the console must be easily legible in a moderately lighted room without flicker, blink, or eyestrain.

To allow proper operation and control in a computer time-sharing environment, the console should also have such features as computer controlled keyboard lock, an interrupt capability, and a unique identification code which the computer can read. These are requisite properties for terminals in the MIT time-sharing system, and for most others as well.

Finally, in order for such a console to fulfill its role as a computer time-sharing terminal, it must be inexpensive. The target price cost which is felt to be reasonable is from $3,000 to $5,000 in production quantity. Although somewhat arbitrary, this figure looks reasonable with the cost of components coming down, with proper design, and with the projection of large-volume production.

The major design constraints imposed by these goals may be summarized as follows:

- Bandwidth-limited data input -- telephone line speed.
- Need for a CRT display with high quality, fast text display, and random vector drawing ability.
- Need for a low-cost unit in a stand-alone configuration.

3. Design Considerations

In considering possible approaches to meet the above goals, TV displays were ruled out because of the difficult transformation of generalized graphics to video format. If this is done at the remote unit, it is expensive. If it is done at the central computer, the mass of data (10^6 bits per picture) requires much too long a time to transmit over a telephone line. Repetitively regenerated random-access digital displays were also ruled out, because while they are technically feasible for telephone-line operation, they require high-cost deflection amplifiers and high-speed memory to keep large amounts of data refreshed without noticeable flicker.

In view of these problems in attaining low cost in refreshed displays, the decision was made to investigate the feasibility of using a display device in which a picture is stored in image form as written. This decision was influenced by the introduction in 1964 of the Tektronix Type 564 direct-view meshless storage tube (DVST). Although this tube has only a 3" by 4" screen area, the resolution and other characteristics largely met the goals stated above, and there were indications that larger tubes were under development.

The use of a direct-view storage CRT was a key design decision.* With such a device, there is no need to provide either an electronic memory or high-speed electronics, such as are required for rapid picture regeneration in displays using conventional CRT's. New data is written only once and may be entered randomly on the display screen at rates compatible with the relatively low-input bandwidth; thus speed requirements on the electronics are quite modest.

The low-input bandwidth implied that efficient data coding be used to generate complex displays in a reasonable time -- say 10 to 20 seconds for a full screen. Unfortunately, such coding increases the amount of data processing equipment at each display station. In this case, it was concluded that a vector and a symbol generator must be included in each display terminal. The design problem then was to perform these functions with low-cost equipment, exploiting if possible the low requirements on speed of display generation, as compared to regenerative displays.

The first task was to choose an appropriate beam-positioning technique that would be capable of random point plotting, vector generation, and symbol generation. Because of the success of the incremental scheme employed in the ESL Console and its basic simplicity, it was decided to use a similar approach in the low-cost display. Recall that

---

*There are a number of other promising image-storage techniques under development (e.g., photochromics, EL-PC panels, photoplastics) which may someday permit large, high-resolution image-storage displays at very low cost. However, none of these at this time permit the instantaneous viewing of material as it is written. Since it is an important characteristic of a computer time-sharing terminal that data be immediately visible as it is written, image-storage techniques that require developing and fixing of a latent image (e.g., photography) will not perform well in a man-computer conversation.

the ESL Console uses binary rate multipliers (BRM's), in conjunction with up-down counters and digital-to-analog converters, to move the CRT beam in small, discrete voltage steps. By intensifying the beam after each step, a constant intensity line, made up of a series of closely-spaced dots, is drawn on the screen. BRM's can easily be made from readily-available, inexpensive, digital-building blocks. The up-down counters and digital-to-analog converters, however, appeared to be high-cost items and an alternate solution was sought. Noting that these simply integrate pulses, it was decided to adopt an unconventional, hybrid approach. Pulses produced by each BRM are shaped to have constant amplitude and duration and then are fed to an operational amplifier connected as an integrator. Polarity of the pulses is controlled in accordance with the sign of the vector component. The resulting amplifier output produces the same discrete-step characteristic as the more expensive up-down counter and D/A converter combination. Thus in the low-cost display, all beam motions are controlled by pulse inputs to the integrators.

The major circuit problem with this approach was that the two integrating capacitors must "hold" the voltages impressed upon them, without significant drifting during the 10 to 20 seconds required to produce a typical "picture". Because of various leakage paths, the capacitors will tend to discharge and cause the beam position to drift. However, with careful design -- use of very stable amplifiers, very low-leakage FET input gates and high-quality capacitors -- drift has been held to one screen position per two minutes. This is quite adequate. Despite the requirement for high-quality components in the integrators, costs are substantially less than the equivalent up-down counter and D/A converter, primarily because of the present availability of low-cost solid-state operational amplifiers.

4. Description of ARDS-II

To confirm our conviction that a low-cost display using image storage and incremental plotting techniques was feasible, we have designed and constructed a prototype unit, which has been designated ARDS-II (Advanced Remote Display Station - II), a block diagram of which is shown in Fig. 24. A previous breadboard output-only unit

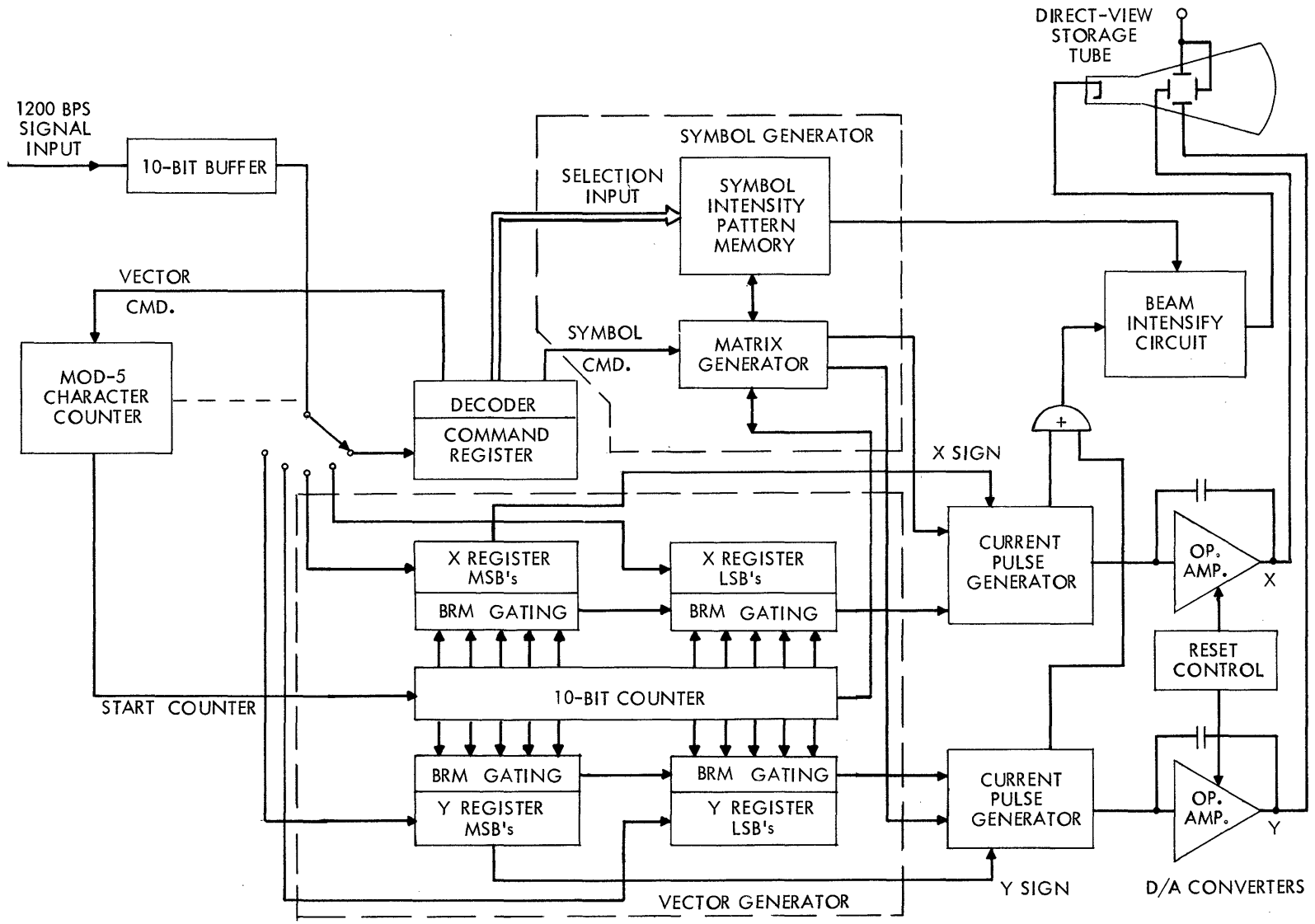


Fig. 24 Block Diagram of ARDS-II Low-Cost Remote Display Terminal

(ARDS-I) was constructed in 1965 (partly as a thesis project) to demonstrate the feasibility of the basic display generation techniques.

a. Vector Generation. Pictures are drawn on the ARDS-II by using incremental vectors -- that is, picture elements are constructed by connecting straight-line segments end-to-end. In a Long Vector command, the incremental vector lengths are defined as 11-bit sign-magnitude numbers, which are loaded into horizontal and vertical registers. The values stored in these registers program the binary-rate-multiplier to produce synchronized Δx and Δy pulse trains of 0 to 1023 pulses each (there is also a Short Vector command which uses 6-bit sign-magnitude numbers to produce up to 31 increments in Δx and Δy.) The clock rate is 100 kHz, i.e., the maximum plotting rate is a point every 10 microseconds. These pulse trains are fed as positive or negative pulses, depending on the Δx and Δy sign bits, into the two operational amplifiers where they are integrated and cause the CRT beam to trace out the desired line. Vectors can be visible or invisible, as specified by an intensity bit in the command.

In the Vector command described above, the beam motion is incremental, i.e., each vector starts at the beam location resulting from the previous command. In order to produce starting points for vector and symbol sequences, and to perform random point plotting, we also needed to be able to position the beam at absolute screen locations. Since the absolute screen location x, y is the same as the end position of a vector with incremental values Δx, Δy plotted from starting location 0, 0, it was possible to use the vector generator to also act as a "set-point" generator.

A Set Point command operates in the same manner as a Vector command except that reed relays short the integrating capacitors and return the beam to the screen center (0, 0) before the vector generator is started. No penalty in display speed is incurred because the slow data input rate provides more than enough time to accomplish both the zeroing and vector drawing functions in one instruction cycle.

b. Symbol Generation. Symbol generation was another important problem, since this is usually quite an expensive function in display systems. Quite often, graphical displays are designed with symbol

generation and vector generation being treated as entirely separate problems. By sharing certain circuits between vector and symbol generating tasks we have achieved important cost reductions. Given the basic incremental beam positioning system which has been described, all that is needed to create a dot-matrix symbol generator is a pair of counters to step the beam through a matrix pattern, and a memory to hold the specific intensity pattern for each symbol, which causes the CRT beam to blank and unblank as it moves through the matrix. The matrix pattern includes inter-symbol spacing, i.e., it leaves the CRT beam in the proper screen position to plot the next symbol, so that no beam repositioning is needed between symbols in a text line. Initial positioning for the first symbol in a line is established by a previous vector or set point, thus there is full flexibility in choosing symbol locations. Design of the symbol generator is discussed in Reference 60.

Because we are limited to telephone-line transmission rates, plotting speed is not a problem, and a 7 x 9 dot matrix was chosen to display the 94 printable symbols of the ASCII code with reasonable fidelity. This requires a 96-word, 63-bit read-only memory for the symbol patterns (a blank symbol is used for "space", and a filled-in pattern for "delete"). The symbol memory is a 6720-bit diode matrix array contained on a single one-half-inch square integrated-circuit chip, manufactured by the Autonetics Division of North American-Rockwell.*

c. Communications and Control. The function of the communications and control portion of ARDS-II is to accept a continuous, serial bit stream from the telephone line, and decode this into commands and data to enable the machine to perform useful functions -- such as drawing a line or printing an alphanumeric symbol. For practical reasons, transmission is in the form of fixed-length "characters" with "stop" and "start" bits to insure continuous synchronization between source and receiver. The character set chosen is the American Standard Code for Information Interchange (ASCII) which will be used in the new Project MAC MULTICS System. This choice was also prompted by the desire

*Electronics Magazine, May 30, 1966, pp. 152A-152D.

to build a console that would be compatible with other computer systems. In the ARDS-II, therefore, characters are 10 bits long (with a start bit, 7 data bits, a parity bit, and a stop bit).

ARDS-II currently has four modes, entered by the ASCII group separator codes as follows:

Control Character	ARDS-II Mode
FS	Symbol
GS	Set Point
RS	Long Vector
US	Short Vector

In Symbol mode, each succeeding character is treated as a symbol to be plotted, and ARDS-II acts just like a teletypewriter (it is in fact directly compatible with a Model 37 Teletype). A 10-bit input shift register, driven by a clock that is resynchronized by each new character, serially stores an entire character. When the input register is loaded, the data bits are immediately shifted out at high speed to one of five 7-bit registers. A mod-5 character counter gates the first character into the command register. If this character is one of the 96 text symbols of the ASCII code, the symbol generator is activated, and the symbol is plotted on the display screen at the current beam location. At the same time the character counter is reset and a new character is assembled in the buffer.

In the Set Point and Long Vector modes, 22 bits are required to define the x and y components, and an additional bit is required for intensity control. This binary information is transmitted in groups of four ASCII characters, using six bits of each character (this scheme avoids problems which arise when binary data includes the ASCII control codes). The Short Vector mode was added to improve efficiency for very short vectors. It uses a two-character sequence to define vectors up to 31 increments in each component. In all of these modes, the plotting operation takes place during the time while the next character is being assembled in the input buffer (8.8 milliseconds with the present 1200 bit per second transmission).

d. Graphical Input. For graphical input, one would like to be able to move a pointer or "cursor" over a stored picture and yet not

store the image of the cursor. Fortunately, the characteristics of the DVST screen we are using are such that there is a stable gap between the "image-visible" intensity level and the "image-storage" level. Also, there is a maximum writing rate beyond which storage cannot occur. Therefore, by lowering normal intensity and moving the beam through the cursor pattern rapidly, a very visible but non-storing cursor is available.

The cursor pattern is generated locally, and its position on the display screen is controlled by means of a hand-held box that is moved about on a surface. This box, similar to a device called the "mouse" by its developers at Stanford Research Institute,* has two potentiometers mounted at right angles to each other. Wheels attached to the potentiometers contact the surface, and resolve the motion of the box into two orthogonal components which are fed as voltages to the CRT deflection inputs. Thus, the cursor on the screen "follows" the motion of the "mouse". A low-cost resistive tablet has also been investigated.

At the request of the operator (pushbutton control), an analog-to-digital converter digitizes the vertical and horizontal components of the cursor's (or tablet's) position and transmits them to the computer. The operator's program can then interpret these position values as it sees fit--as endpoints of lines when in drawing mode, as a pointer in locating one of a number of displayed objects, and so on. This type of input has only barely begun to be investigated, but it is felt that it will provide most of the functions now obtained with light pens and input tablets on the more elaborate, refreshed consoles.

e. Pictorial Results. The ARDS-II prototype as it existed in May, 1967, is shown in Fig. 25. Note that the storage tube used was the Tektronix Type 564, with a 3" by 4" screen. This permitted display of up to 1200 quite legible characters, but was considered to be too small for a use in a final terminal.

An engineering model of the new Tektronix Type 611 Storage Display Unit was obtained for tests in August, 1967. Although this is

*English, W. K., Englehart, D. C., Huddart, B., "Computer-Aided Display Control," Final Report, Contract NAS 1-3988, Stanford Research Institute, Menlo Park, California.

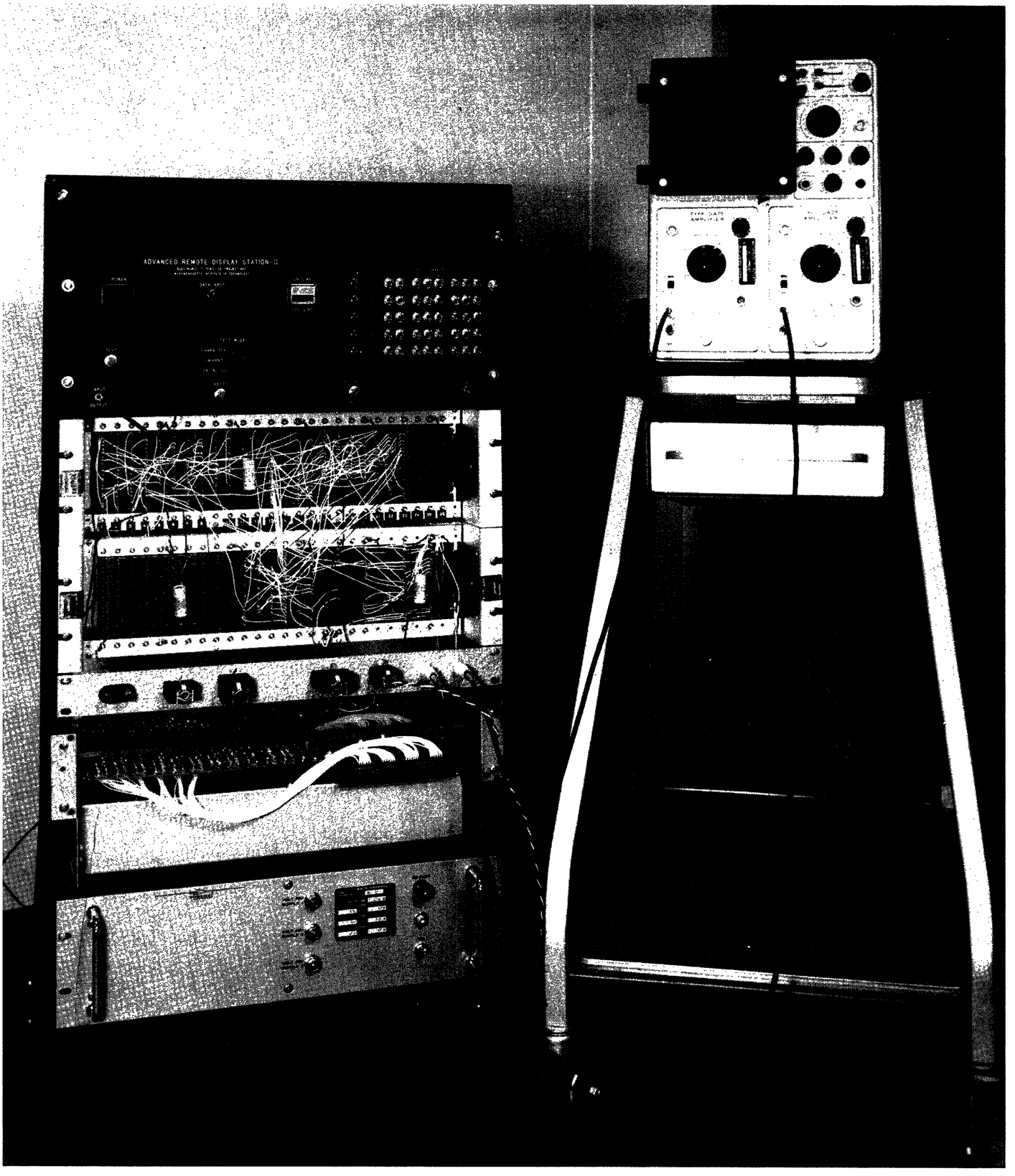


Fig. 25 ARDS-II with Tektronix 564 Storage Oscilloscope

not within the period covered by this report, the results were so striking that it is appropriate to conclude this discussion of the ARDS development with a sample display from this new tube. The Type 611 has screen dimensions of 6.5" by 8.5", and has a stored spot size of only 0.008". Resolution is about twice that of the older Type 564. Figure 26 shows a full-size photograph of an ARDS-II display on the Type 611. The text size shown permits 4000 characters per page. The next stages in the ARDS development will be to add an input keyboard and package the electronics and storage tube into a self-contained table-top unit.

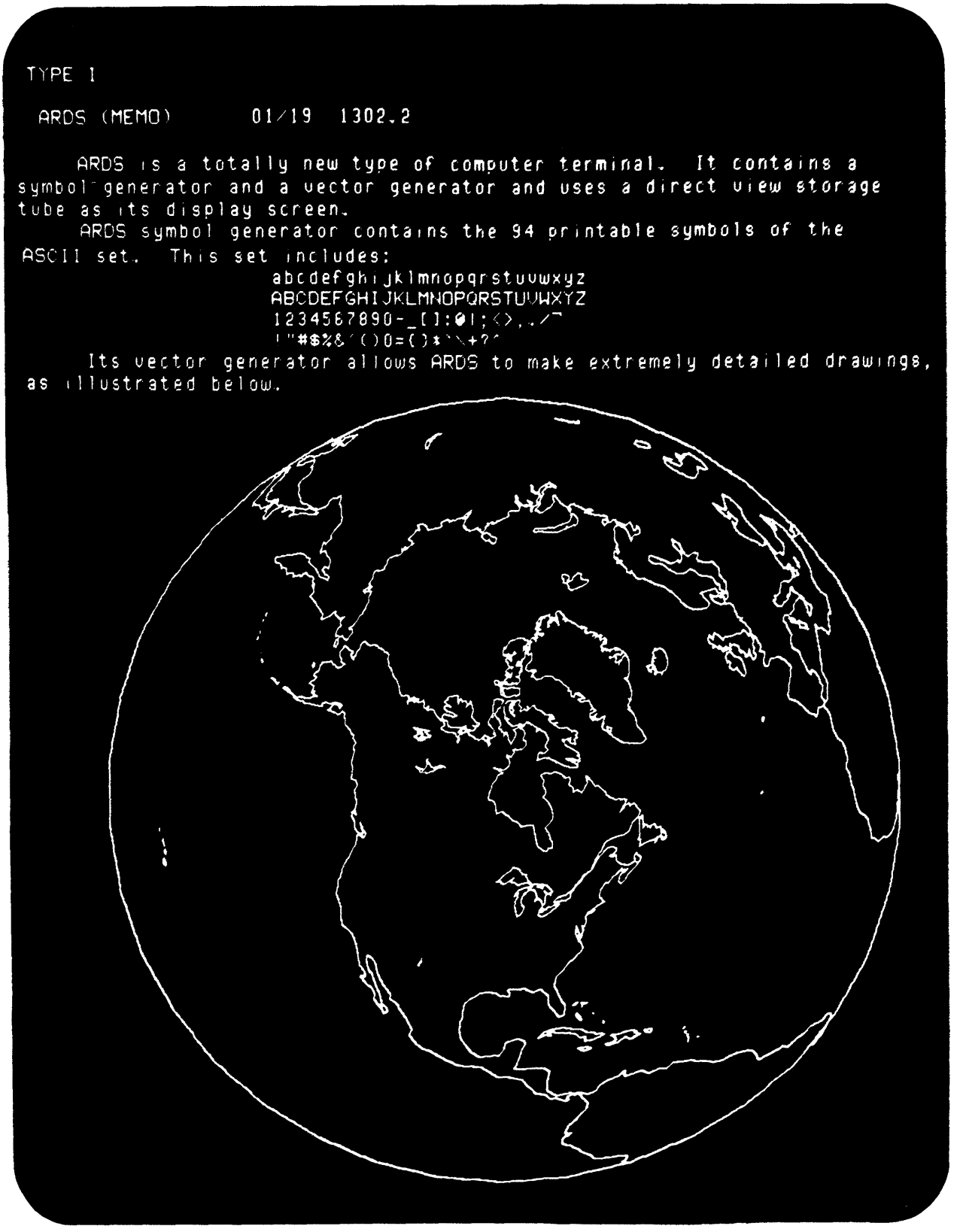


Fig. 26 ARDS-II Display (Full Size) with Tektronix **Type 611** Storage Tube

CHAPTER VIII

GRAPHIC SOFTWARE

In addition to its work in display hardware, the Project also has from the beginning been active in the study of generalized techniques for the use of displays in man machine-interaction -- the software side of computer graphics. We have never thought of this display-oriented activity as merely picture-making, but always have considered graphics in the total context of computer-aided design. Computer graphics, in our thinking, has always been synonymous with graphical language, with a heavy admixture of problem-modeling inextricably interwoven to encompass the meaning of graphical language in the total problem-solving process.

A. HISTORY

As was the case with other interests, activities of the Electronic Systems Laboratory in the area of graphic software predates the initiation of the Computer-Aided Design Project proper. In addition to on-line graphical input-output for data reduction and general problem-solving systems (including program control, situation displays, and an elaborate system for plotting graphs of functions), the three-dimensional cutter-path displays of the APT System have already been mentioned. One of the most pertinent and prophetic studies was carried out in 1959 as part of the transition from APT to Computer-Aided Design. This study, called the "Point-Line Diagram" study had three main purposes:

1. To provide a general model of the computer-aided design process as we envisioned it.

2. To study the applicability of the (then) new list-processing languages as a technique for carrying out our (then) new plex programming techniques without using machine language programming.

3. To provide a concrete example of a miniature computer-aided design system, both for demonstration and to serve as an initial vehicle for later, more elaborate investigations.

The view of the design process, which is still valid today, was that the composition of a design involves an iterative process in which various sub-elements are composed into interrelated structures, which then are forced to satisfy specified conditions. Before the precise conditions can be specified, the structure being built at any given stage is a loose structure with many degrees of freedom represented by unspecified parameter values. At some stage, sufficient structure will exist to allow the desired constraints to be imposed. The parameters of the various elements may then be adjusted by a constraint satisfaction routine until the desired condition is reached, or until it is necessary to add more structure, change some conditions, or relax some constraints.

The point-line diagram study illustrated this process in a very simple form. The basic elements were points, lines, and angles between lines, which were represented in list structure composed and manipulated using the LISP list-processing language, then under development by the MIT Artificial Intelligence Group. Coordinates of points, lengths of lines, and angles between lines could be constrained to fixed values, and various "hill climbing" algorithms were used for the multi-variable constraint satisfaction process. All input to the resulting system was in character form, although side investigations were made to show that the various light pen graphics studies (being carried on in parallel on the TX-0 Computer at that time) could easily supply graphical input and output if desired. The study was satisfactorily completed, confirming both the validity of our view of the design process and the expectation that general list-processing techniques were far too inefficient (both in storage and in execution time) for practical use in computer-aided design. It was on this basis that the Project undertook the development of plex programming and higher level language development leading to the AED Systems of today.

As has been described previously, the Bootstrap Compiler was our first step in language development. After this work was partially underway, we also began graphical language studies to accompany the display equipment being attached to the IBM 709 Computer of

the MIT Cooperative Computer Laboratory. Several isolated experiments were carried out in light-pen tracking, and a "light button" compiler was developed to permit arbitrary control actions to be associated with spots included in a display.

As these activities were drawing to a close, we began the design of a Bootstrap Picture Language, much along the lines of the Bootstrap Compiler, with the intention that such a system could then be coupled to design examples such as the point-line diagram study. The Bootstrap Picture Language system started out with a very small number of built-in light buttons displayed on the scope. One of these light buttons represented a geometric point and the others enabled geometric objects to be composed from other existing geometric objects, with specification of special display features, if required. Once a generic object had been created, any number of copies could be made and used to compose other objects. The bootstrapping process consisted of defining a straight line in terms of the built-in point, after which objects composed of lines could also similarly be bootstrapped into existence. In general, pictures were composed of sub-pictures and could themselves be treated as sub-pictures.

Part way through this development, interest developed in constructing a full-blown Multi-Pass Compiler to replace the Bootstrap Compiler. Out of initial investigations grew the Algorithmic Theory of Language upon which much of the remaining work of the Project has been based. (The language theory itself is an application of the "growth" concepts of plex programming, so that all of these activities have a common heritage.) During the early stages of development of the language theory, examples were drawn freely from algebraic programming language, structured geometric language (à la the Bootstrap Picture Language), and various constructions drawn from natural English language. Once the principles of the language theory began to solidify, attention focused primarily in the programming language domain, both because of the need for replacement of the Bootstrap system and also because many aspects of the graphical language requirements were being satisfied by the success of the thesis

investigation of Ivan E. Sutherland, which ultimately led to the Sketchpad System (see Reference 22 in Appendix II).

B. SKETCHPAD

Sutherland had several sources of sponsorship for his thesis activity. In addition to a National Science Foundation Fellowship stipend, he received generous and free use of the facilities of the powerful transistorized TX-2 Computer at Lincoln Laboratory. Also, from the MIT Computer-Aided Design Project, Professor Coons (who served on the Thesis Committee) and Mr. Ross provided guidance and encouragement. The Sketchpad development went through four principle phases. In the summer of 1961, initial display investigations were concerned primarily with plotting functions and investigating pen-track logics, and laid some preliminary groundwork.

In the fall of 1961 an initial computer-aided drafting system was written, based on an internal Project memorandum of Coons. This system implemented for the first time the "pseudo pen" or "nearest indicated point" technique whereby the system selects a point exactly on a displayed object whenever the tracking cross location is within some preset distance of that object. This technique allows precise manipulations to be performed with the inherently imprecise light pen. The structure of the initial drafting system was very rigid, with specific push buttons for constructing horizontal and vertical lines and performing normal drafting actions. The internal modeling consisted of fixed arrays of coordinate information.

The second phase of these developments followed Sutherland's introduction to the concepts of plex programming (as practiced in the Project at that time), the Point-Line Diagram study results, and the generic and specific instances of sub-pictures of the Bootstrap Picture Language, etc. Many of these concepts received very natural representation in terms of the indexing, partial word, and multi-sequence interrupt capabilities of the TX-2 Computer, and its large core memory of 65K words enabled elaborate data structures to be built and manipulated with ease. This first Sketchpad System was internally

structured in terms of interlocked ring structures, and included subpicturing capability. It also had a more flexible input language, departing from the drafting-like flavor of the initial system. For example, a basic constraint satisfaction facility enabled lines to be constructed at any angle and then a single push-button specified that an indicated line was to be made exactly horizontal or exactly vertical, depending upon which was most nearly applicable.

The final phase of Sketchpad development involved a reorganization of several internal features of the system and an elaboration of the constraint satisfaction capability, including the graphical display of applicable constraint conditions and use of the system for various applications to which it was well suited. Sutherland's skill, inventiveness, and diligence in expressing these powerful concepts in a smoothly functioning system, making maximum use of the powerful features TX-2 Computer, enabled Sketchpad to bring to life for many people the vast potential for computer-aided design. In particular, the widely distributed movies of Sketchpad in operation have had a profound influence on the whole field of computer graphics.

C. THE ESL CONSOLE IN TIME-SHARING

After the Sketchpad era, the generalized graphic software activity of the Project began, based upon the use of the ESL Display Console in time-sharing. As was mentioned in the previous chapter, the Console was placed in operation on the Project MAC 7094 time-sharing system (CTSS) in the spring of 1964, and within a few months preliminary software was available to enable its use in initial applications. Figure 27 shows the general system which was used from 1964 to mid-1967 for operating the ESL Display Console in the time-sharing environment. The time-sharing system has two 32K core-memory banks called A-core and B-core. A-core contains the time-sharing Supervisor programs which are always active. User programs are cycled in and out between B-core and the mass storage of the magnetic disk system by the Scheduling Algorithm of the Supervisor. In order to maintain a regenerative display on the console and in order to

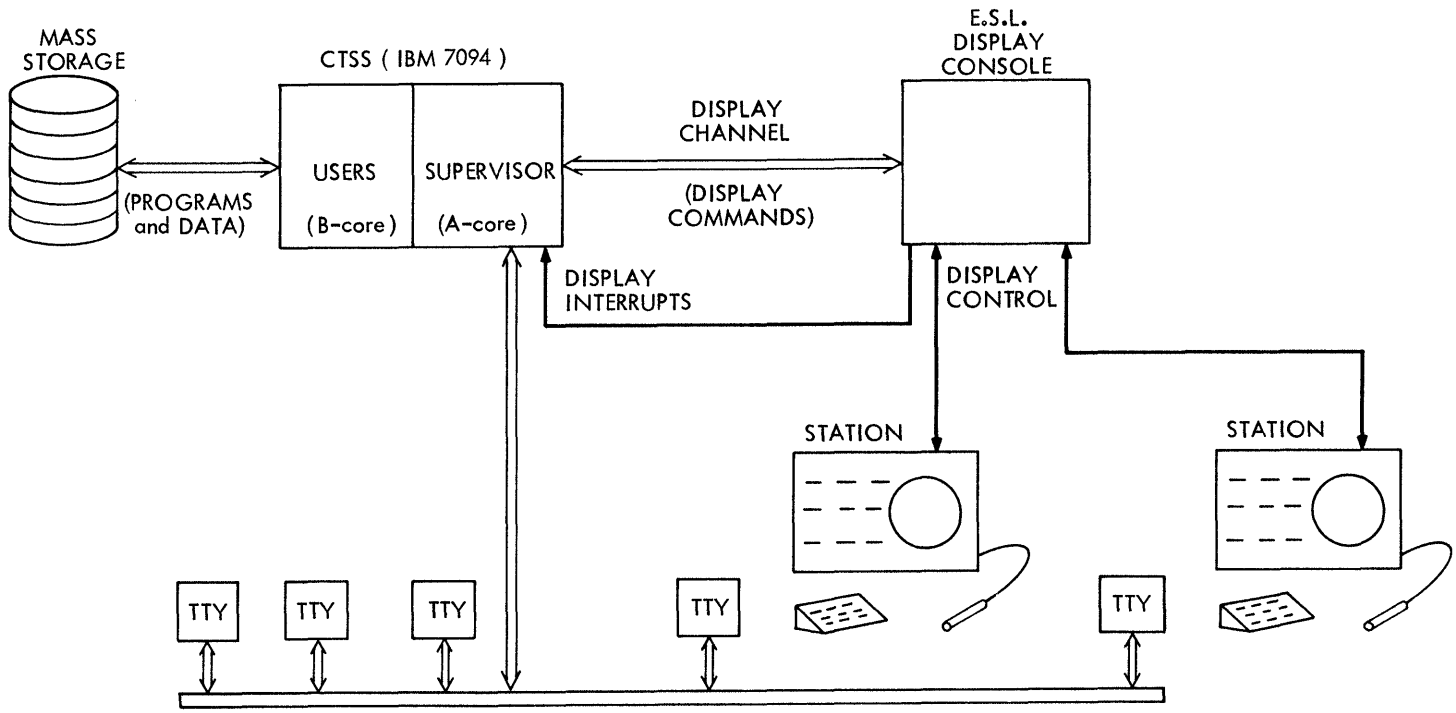


Fig. 27 ESL Display Console Connected to CTSS

provide real-time response to the electronic interrupts caused by real-time actions at the Console, a portion of the Supervisor memory in A-core was allocated to service display needs. This portion of the graphical software system was called the "A-core Package" and occupied about 4,000 36-bit words of the 7094 core memory. Of this amount, roughly 1/3 held the user display file of commands to the hardware display generation system, and the remainder was used for programs.

There are several categories of program in the A-core Package. Some programs control the timing and sequencing of the display file words through the data channel to the display console. Others physically construct and perform editing functions on the display commands. Others respond to interrupts from the console, composing messages called attentions for transmission to the B-core user program when it becomes active in time-sharing. The final category, called real-time functions, perform calculations and manipulate the parameters of various display commands to provide such features as real-time rotation, "rubber-band" lines, etc..

All communication with the display console must take place through procedures of the A-core Package, since only the carefully debugged routines of the Package can guarantee the integrity of the time-sharing system against user-generated errors. Cycling of the display is controlled by one of the real-time clocks in the ESL Display Console which generates an interrupt to A-core 30 times per second, at which time the display cycle is reinitiated if it is not already running. If a display cycle takes longer than 1/30th of a second it is restarted as soon as it finishes in order to minimize flicker. Before a recycle of the display begins, the binary and decimal switches and the "crystal ball" of the display console are sampled and their values are stored in A-core for use by programs if desired.

D. THE DISPLAY EDITOR PACKAGE

The routines of the A-core Package are quite specialized, and a fairly detailed knowledge of various technical details is required in order for a user to perform desired actions through A-core calls.

Since many of these features are dependent upon the characteristics of the ESL Display Console and the particular form of time-sharing being used, a complementary display file generation and editing package, called the B-core Package, was written to complete the user interface to the display system. As it's name implies, the B-core Package is not continually resident in core memory, but instead cycles in and out as à part of the user's own program in B-core. The B-core Package consists of a large number of atomic and molecular subroutines which permit any desired display file to be written and changed. Not only may the entire range of basic functions of the ESL Display Console be used, but in some instances the B-core Package provides a higher-level of approach to the problem of programming graphics. The B-core Package is described in Reference 32.

A family of B-core procedures also allow administrative functions to be performed, as well as generation and editing of the display file which composes a picture. All routines are written in accordance with the principles of integrated package design so that calls on the procedures may be nested one within another to compose high-level constructs in an efficient manner. In addition to constructing picture objects from points, lines, and circular arcs, messages composed of characters may be displayed, rotation matrix settings may be altered by real-time functions, and other console actions may be effected. A single call for a circular arc will automatically cause the required number of straight-line commands to be inserted into the display file to compose an approximate arc out of straight-line segments.

Other procedures in the B-core Package are concerned with the processing of attentions caused by push-button or light-pen interrupts from the console. Groups of display commands also may be composed into sub-pictures which may be called from many places in the display file, just as closed subroutines are called in a computer program. When a pen response is received, the user program must know which subpart of the total display was seen. The B-core Package provides a general and powerful solution to this problem in the form of display registers which may be imbedded in the user's own data

structure, so that the B-core Package can indicate directly (in the terms of the user's own problem) the object which was seen. This referencing system can become elaborate and very important when sub-pictures are used, for it is necessary to give the entire ancestry of the final object in order to receive an unambiguous indication.

These and many other features of the B-core Package provide a sophisticated high-level interface to the intricacies of display programming which is to a great extent independent of both the display system and time-sharing system. The technique is similar to the Postprocessing idea of the APT System which allows the APT System to generate control tapes for many different machine tools, merely by changing the Postprocessor Module of the system. We use the same system of B-core calls to drive both the ESL Display Console and the ARDS storage display, even though the systems are connected to the time-sharing system in quite different ways and have very different command languages and display characteristics.

The B-core philosophy not only provides display and operating system independence similar to the interchangeability of Postprocessors in the APT System, but the high-level approach to display programming is of great value in enabling powerful systems to be constructed efficiently and reliably. This feature was dramatically demonstrated during the preparation of the B-core Package itself. The ESL Display Console was initially constructed to provide its rotation capabilities only in two-dimensions, with the third dimension mechanization saved until the approach was proved out. The initial B-core Package was written and already in use for two-dimensional graphics when the third dimension was added to the display console capability. This necessitated the addition of a new routine to the B-core Package to enable display construction with three-dimensional lines. When this program was written, it was necessary to write a test case to check it out.

Mr. Charles Lang, author of the B-core Package, decided one Sunday evening to use the three-dimensional drawing technique illustrated by Sketchpad III as a test vehicle. In this scheme, the screen

is divided into quadrants with simultaneous display of plan, top, and side views, plus a three-dimensional perspective or axonometric projection. In that single evening, Mr. Lang wrote and debugged the entire system, verifying the correctness of the new three-dimensional line subroutine. Except for the fact that the B-core Package did not at that time include a pseudo pen capability, the performance of this test system was identical to that of the TX-2 Sketchpad III System, developed in a Master's thesis only a year before. (A three-dimensional pseudo pen package now exists as an adjunct to the B-core Package.) Thus, even though this application had not been planned in advance, the building-block facilities of the B-core Package (coupled with the powerful features of AED-0 and the time-sharing system) permitted a very high degree of almost casual automation to be applied to a problem which had only a short time before been a significant research effort.

E. THE DISPLAY INTERFACE SYSTEM

As discussed in Chapter VII, the ESL Display Console was initially directly connected to the A-core data channel only as an expedient to gain experience and to assist in deciding how much more than a simple buffer memory was required to adequately support a sophisticated display console. Even though the A-core/B-core system was relatively efficient, the combination of display generation and real-time computation on occasion placed an unacceptable load on the time-sharing system, decreasing the resources available to the more numerous nongraphic users of the system. Thus, in early 1967 a PDP-7 Computer was purchased by Project MAC for insertion between the A-core data channel and the ESL Display Console, to serve as a display buffer and minimize the load on the time-sharing system.

Figure 28 illustrates the revised organization of the Display Interface System to accommodate this two-computer organization. As is characteristic of the other activities of the Project, we have approached the problem of coupling a display console to a time-shared computer through a dedicated real-time computer in a manner which is as general as possible, and yet which is practical for economic use.

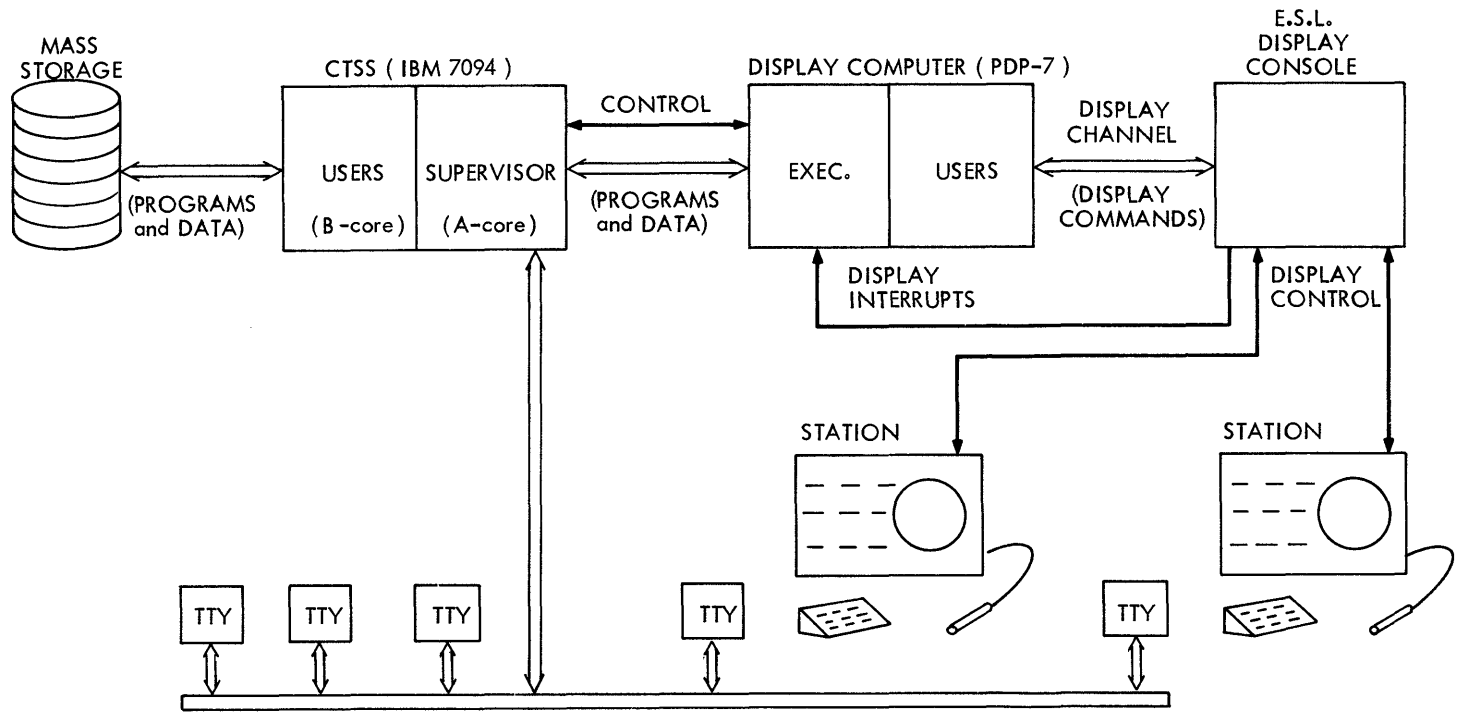


Fig. 28 Use of PDP-7 as a Buffer Computer for the ESL Display Console

The new Display Interface System is intended to be independent of display and computer hardware, system-organization, and application area, so that the same general approach can be applied to a wide variety of circumstances. As before, the B-core System Display Editor Package provides the outer interface to the user problem, so that very little program modification is required in the event that hardware and operating system characteristics are changed (even drastically).

The central idea in the Display Interface System, as described in a paper at the 4th Annual ACM/SHARE Design Automation Workshop in June, 1967 (see Reference 110 in Appendix II), is that the total storage and computing time of the real-time computer are continuously dedicated to the user's problem, even though the remainder of his system cycles in and out in time-sharing and is only occasionally active. For economic reasons, the real-time computer will in general be as small as possible to provide the required capability, and therefore real-time computer capacity is a limited, precious commodity. We seek to develop a system organization which will provide minimum overhead for all classes of users, so that each user may have the maximum amount of real-time computer capacity to use as he sees fit. This general idea translates into the concept of a minimal executive for the real-time and time-sharing computers.

In general, the minimal executive programs are concerned only with transmission of messages between the two computers. A message may consist of user information (display file, data, control, or program) or system program. The absolute minimum real-time executive for the small computer need only be able to

1. receive messages from the time-shared computer,
2. respond to physical interrupts from the display console, and
3. be able to tie in system programs in a simple fashion into its execution cycle.

In this way, the first messages from the big computer may be selected system subroutines to be added to the minimal executive to

give an augmented executive, capable of performing precisely those actions which the user requires. Among these actions in general will be the ability to send messages back to the big computer.

In the time-shared computer, the purpose of the minimal executive is to minimize the fixed overhead on the time-sharing system imposed by display system users. Virtually all of the previous A-core functions may now be performed in the real-time computer (or in the user B-core area), so that the display executive in the time-shared computer is strictly a communication coordinator which is only active in establishing links to and from the B-core Package and the augmented executive of the real-time computer.

Under this new organization, the combined functions of the old A-core and portions of the old B-core Packages are broken into well-defined modules, which are mechanized in two ways so that any function can be performed either in the real-time computer or in the time-shared computer. Libraries of these routines will be available, and new calls in the Display Editor Package will allow the user to request that a total system with specified limited characteristics be dynamically assembled for him and transmitted from the mass memory of the time-shared machine to the real-time computer. One user may desire to have dynamic storage allocation and large display structures, but very little real-time computation. Another user may have only simple displays but may wish to use the real-time capability for on-line dynamic problem simulation with moving displays. Under the new organization, each class of user will pay only the overhead required by his particular kind of usage. Notice also that the same system structure can apply even though the real-time computer becomes a very large computer, as might be desirable if the communication link costs become high because of great distance between the two computers.

An initial version of the system described above was placed in operation in 1967 in a simplified form which duplicates all of the functions previously performed in the old A-core/B-core system, but with a greatly-reduced load on the time-sharing system. A new version, with greater emphasis on machine-independence and versatility is currently under study.

CHAPTER IX

DESIGN DIVISION STUDIES

A. THESIS ACTIVITY

Early in the Project, as was mentioned previously, the Design Division of the MIT Mechanical Engineering Department was invited to join forces with the ESL Computer Applications Group in the work of the MIT Computer-Aided Design Project. The intent of this collaborative venture was to cross-fertilize by having the two groups approach the common goal from the points of view of mechanical design and computer applications respectively.

The work of the Design Division took concrete form in a number of wide-ranging investigations into the design process and into various specific application areas. Most of this work is summarized in the complete abstracts for the fifteen Mechanical Engineering Masters and Doctoral theses which resulted from this work given in Appendix II-C. Taken as a whole, this collection of theses served an important role in providing illustrations of the types of activities which can be foreseen once computer-aided design becomes an economic reality in the industrial scene. Because the studies were performed in the thesis context, however, they did not receive public distribution and are not sufficiently developed to be applied as actual production tools. They served their role well in providing demonstrations and an environment of increasingly sophisticated application of computers in the design process, in the formative stage of the evolution of the computer-aided design concept. We look forward with anticipation to the development of similar applications, carried out in sufficient depth to stand up to the rigors of actual production use, in the industrial context.

B. GENERALIZED SURFACES

A very significant development of the Project was the extensive work carried out by Professor S. A. Coons in the generation of sophisticated mathematical techniques for three-dimensional shape description

and high-level parametric functions. The bulk of this work was carried out throughout the period when Prof. Coons was a leader of the Design Division efforts, and received its final publication as MIT Project MAC Technical Report No. 41, "Surfaces for Computer-Aided Design of Space Forms," [Ref. 39]. The following sections are taken from that report and present the basic techniques involved.

1. Introduction

The purpose of this work is to present the mathematics of a certain class of surfaces which are suitable for the design and description of arbitrary shapes. In the past, the subject of surface mathematics has been investigated, in analytical geometry and in differential geometry, from the standpoint of the analysis of geometric properties of surfaces that already exist, but very little literature has been produced on the subject of the creation of such surfaces. As a typical example, the design of the hull of a racing yacht requires the description of a surface of considerable subtlety and complexity, and the process is traditionally carried out by purely graphical procedures which are exceedingly laborious, since they entail a large amount of trial-and-error iteration in order to assure that the surface is completely described, and is smooth and "fair". The design of automobile bodies and airplane fuselages is similarly tedious and time consuming, although mathematical techniques have been applied to aircraft design for a number of years.

The mathematical structure of the surfaces to be described in the following discussion has been devised to implement the surface design process itself, so as to make it, from the designer's standpoint, extremely natural and easy. The designer himself need not know or care about these internal mathematical details, any more than he needs to know the specific composition of the pencils with which he writes or the mechanics of the splines with which he now draws curves. The mathematics is relatively simple, but it is nevertheless too complicated for hand calculation, and is designed for use on a computer.

In the design of a three-dimensional object, whether it be an airplane fuselage, an automobile body, a ship's hull, or a single sculptured part of a machine, the designer requires a system which will permit him to define a surface with a minimum of input information,

and then to modify this surface, if he feels so inclined, either by changing the original input, or by adding more design constraints to the system.

The following sections describe a very simple, flexible and general class of surfaces which are able to fulfill these requirements. It will be shown that a single algorithmic structure provides the following features:

1. Smooth, fair surfaces can be defined by a minimum number of curves, and then adjacent surfaces can be designed to match position, slope, curvature, and indeed any desired order of derivative along the adjoining boundaries.

2. The design curves that define the surface can be of any kind whatsoever, including circles, second-degree curves, polynominals, transcendentals, and also sketched curves with no known mathematical formula whatsoever.

3. Some classic surfaces are not necessarily members of the family of surfaces to be described; nevertheless, these classic surfaces can be matched along their boundaries to any order of derivative desired.

4. The arithmetic involved in constructing these surfaces is extremely simple and easy to implement on a digital computer. It also lends itself to special-purpose computing hardware, such as digital or analog differential analysers. In addition, by virtue of the form of the algorithm, the parameters that define the shapes are extremely easy to compute.

We construct complex arbitrary surfaces by piecing together surface "patches". Each such patch is defined by four boundary curves, in principle, although it is harmless for one of the boundary curves to be degenerate, and to appear as a point instead of a curve segment. In the design of a surface, it is intended that the designer begin with a single surface patch, or a very small number of patches, and then subdivide these regions with additional design curves defining boundaries of smaller patches only when the internal surface needs modification. This is somewhat at variance with the customary procedure for mathematical curve fitting and surface fitting of existing curves and surfaces, in which a relatively large number of surface points already defined by some other procedure are used to obtain

mathematical expressions for a surface that best fits them. Instead, the system to be described is intended to be used by the designer at the outset, in the process of designing the surface, rather than later on as a means for making it mathematical. When the design process is completed, the surface will be completely mathematically defined, since this definition occurs automatically and concurrently with the design.

2. Notation

We shall in what follows relate the x, y, and z coordinates of points on a surface to two independent variables u and w, so that we could write

$$x = f(u, w)$$
$$y = g(u, w)$$
$$z = h(u, w).$$

If the functions f, g, and h were specified, then for a pair of values of u and w, a point in space would be defined. If we held one of the independent variables fixed, say w, then by allowing u to vary, the point in space would trace out a curve. If subsequently we set w to a new fixed value and again allowed u to vary, we would trace out another curve, and so on. Clearly by stepping the values of w by small increments and allowing u to vary after each such step, we could produce a family of space curves that would lie on the surface and define it. All that is needed is some convenient and systematic way of arriving at the functions f, g, and h.

It will turn out that the form of all of these three functions is the same; only certain internal numerical values are different. In vector notation we can write

$$[x \; y \; z] = [f(u, w) \quad g(u, w) \quad h(u, w)]$$

Since $V = [x \; y \; z]$ is a suitable conventional abbreviation for the vector on the left, we introduce a similar abbreviation for the right-hand side:

$$(uw) = [f(u, w) \quad g(u, w) \quad h(u, w)]$$

Here, in the abbreviated symbol on the left, we shall omit the comma between the two letters. Later on, when no ambiguity can arise, we

shall omit the parentheses as well, and write simply uw to stand for the vector. It is to be remembered that uw does not stand for the ordinary product of the two quantities, but is merely a bi-literal symbol standing for a vector whose components are functions of the two variables.

We plan to build up surfaces by adjoining surface "patches", in an analogy of the piecewise fitting of complicated curves by curve segments suitably joined together. Accordingly, we shall at the beginning focus our attention on one such surface patch. To simplify arithmetic, we shall stipulate that the independent parameters, u and w, can take on only values between 0 and 1. Then a surface patch can be considered to be a surface segment bounded by four space curves, (0w), (1w), (u0) and (u1).

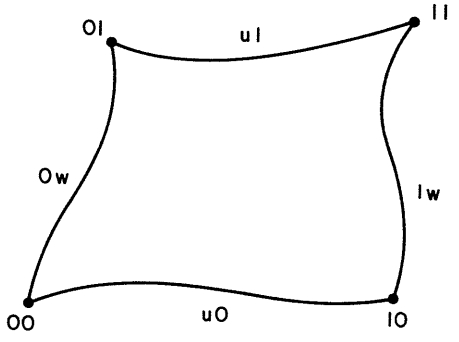


Fig. 29 Simple Surface Patch

Here, typically, the symbol (0w) stands for the vector describing the x, y, and z coordinates of points along the curve generated by allowing w to vary continuously from 0 to 1, while u is held fixed and equal to 0.

We shall introduce two scalar functions, F_0 and F_1 each a function of a single variable. These will be referred to as "blending functions" for reasons that will become clear.

In order to compress the surface equation, and the proofs that we wish to demonstrate, we shall use a kind of indicial notation; we introduce the indices i and j, which can assume only the values 0 and 1, and we invoke the customary summation convention for terms with repeated indices. This convention in our case simply means that when

an index is repeated in a term, we write out all the possible terms that the actual indicial values generate, and then add them.

3. The Surface Equation

With these conventions and notational peculiarities in mind, we write

$$(uw) = (iw)F_i(u) + (uj)F_j(w) - (ij)F_i(u)F_j(w).$$

(Typically, the first term on the right expands as follows:

$$(iw)F_i(u) = (0w)F_0(u) + (1w)F_1(u).$$

Thus the complete expansion would consist of eight terms, if carried out.) We shall proceed to demonstrate that this surface equation represents a surface that contains the four boundary curves, and is thus defined by them.

We must make a stipulation, a weak one, on the nature of the blending functions F_0 and F_1:

$$F_0(0) = 1 \qquad F_0(1) = 0$$

$$F_1(1) = 1 \qquad F_1(0) = 0$$

A further stipulation is that F_0 and F_1 be continuous and monotonic over the interval.

Now set u = a, where a can only be either 0 or 1. Then, substituting in the surface equation,

$$(aw) = (iw)F_i(a) + (aj)F_j(w) - (ij)F_i(a)F_j(w).$$

Consider $F_i(a)$ which occurs twice in the equation. By the stipulation, if i = a, $F_i(a) = 1$. Otherwise, if $i \neq a$, $F_i(a) = 0$. Hence all terms in the expansion that contain $i \neq a$ vanish; we can set i = a, and what remains is

$$\begin{aligned}(aw) &= (aw)F_a(a) + (aj)F_j(w) - (aj)F_a(a)F_j(w) \\ &= (aw) \qquad + (aj)F_j(w) - (aj)F_j(w) \\ &= (aw).\end{aligned}$$

This shows that for a = 0 or 1, and hence (aw) = (0w) or (1w), the surface equation reduces to an identity. An entirely parallel argument would show that the equation also reduces to an identity for the other two boundaries (u0) and (u1). This implies that the surface contains its boundaries.

Provided a pair of functions F_0 and F_1 that satisfy the stipulations are chosen once and for all, the surface equation may be constructed immediately and uniquely for any set of boundary curves (u0) (u1) (0w) and (1w). It is to be observed that the only restrictions on the form of the boundary curves is that they form a closed boundary and they should be continuous functions, but apart from these rather obvious restrictions, they can be of any shape whatever, including curves that can only be represented by tables of values.

We can gain intuitive insight into the nature of such a surface if we look at one of the terms, say $(uj)F_j(w)$.

We have the expansion

$$(uj)F_j(w) = (u0)F_0(w) + (u1)F_1(w).$$

This represents a weighted average of the quantities (u0) and (u1). When w = 0, $F_0(0) = 1$ and $F_1(0) = 0$, and the expression becomes simply (u0). As w increases, the weight of $F_0(w)$ decreases, while that of $F_1(w)$ increases, so that the surface partakes of the nature of both boundary curves. As w approaches the value 1, the influence of (u0) on the shape of the surface gradually disappears while the influence of (u1) gradually becomes dominant. Finally, at w = 1, the curve (u1) represents the shape of the surface. We can say that the surface is generated by a gradual transition from (u0) to (u1), and that these two curve shapes are "blended" together by virtue of the blending functions F_0 and F_1. This discussion is somewhat oversimplified, since we have omitted the term $(iw)F_j(u)$ and it too plays a part in determining the shape of the internal surface, as does of course the term involving the corner coordinates, $(ij)F_i(u)F_j(w)$. The entire surface equation also is seen to be symmetric in u and w.

4. Boundary Slope Continuity

It is our aim to design and delineate complicated surfaces by adjoining surface patches, in a piecewise fashion. Consider two such

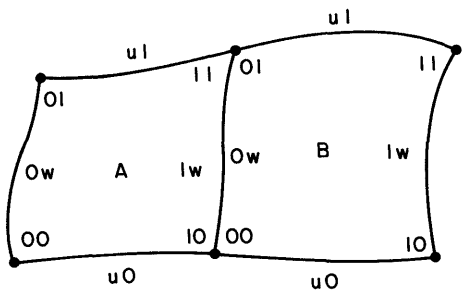


Fig. 30 Connected Surface Patches

patches A and B, with a common boundary. For patch A the boundary is (1w); for patch B it is (0w), and the vectors of coordinates are equal,

$$A(1w) = B(0w).$$

Then the two patches will be continuous across their common boundary. They will however in general be discontinuous in slope across the boundary, and we wish to investigate this and make some amendments that will correct this discontinuity of slope.

We take the partial derivative with respect to u: Our symbolism for this partial derivative is $(uw)_u = \frac{\partial(uw)}{\partial u}$, and when we substitute, say, u = 0, we can write $(0w)_u$ to mean the value of the partial derivative so obtained. Then

$$(uw)_u = (iw)F_i'(u) + (uj)_u F_j(w) - (ij)F_i'(u)F_j(w).$$

Now substitute u = a = 0 or 1, as before.

$$(aw)_u = (iw)F_i'(a) + (aj)_u F_j(w) - (ij)F_i'(a)F_j(w).$$

If we now place additional constraints on the blending functions, that their first derivatives

$$F_i'(a) = 0 \qquad (a = \text{either } 0 \text{ or } 1)$$

we obtain the result

$$(aw)_u = (aj)_u F_j(w),$$

all other terms vanishing.

This implies, for example, that when a = 0,

$$(0w)_u = (00)_u F_0(w) + (01)_u F_1(w),$$

i.e., the derivative anywhere along the boundary in the u direction (across the boundary) depends only upon the derivatives at the end-points of the boundary; it is entirely independent of the shapes of the four boundary curves, including the boundary (0w) itself.

Thus for the two patches A and B, if

$$A(10)_u = B(00)_u$$

and

$$A(11)_u = B(01)_u$$

i.e., if the boundary curves are continuous in slope in the u direction at the ends of the contiguous boundary between patches, we are guaranteed to have $A(1w)_u = B(0w)_u$ everywhere along the boundary regardless of the shapes of the boundary curves of A and B. This is a remarkably powerful and useful property, achieved at the slight exposure of extending the stipulations on the F_i.

Similarly, the second derivative with respect to u is

$$(uw)_{uu} = (iw)F_i''(u) + (uj)_{uu}F_j(w) - (ij)F_i''(u)F_j(w)$$

and if we further stipulate that $F_i''(a) = 0$ we obtain

$$(aw)_{uu} = (aj)_{uu}F_j(w).$$

This establishes second derivative (or curvature) continuity as an automatic and inherent property of adjacent patches, provided their boundary curves have this kind of continuity at the end-points of the boundary. It is easy to see that we may escalate in this way to any level of derivative continuity we wish along contiguous boundaries.

5. Correction Surfaces

The surface equation already described is very general, in the sense that it can contain virtually any boundary curve we wish, and it has certain benign properties of derivative matching along boundaries; nevertheless it is not a universal formula for all surfaces, and there are many that do not belong to its family. We have already seen that surfaces generated by the surface equation have a definite intrinsic slope along boundaries, whose variation is rigidly prescribed by a single formula. Obviously surfaces exist whose boundary slopes do not match this intrinsic slope, except at the end-points of boundaries. Nevertheless, we wish to be able to patch together such other surfaces with our special surfaces, so as to have slope continuity (or continuity of any level of derivative).

To do so, we introduce a new surface equation, describing a slope-correction surface, which when added to the first surface equation has the property of leaving the boundaries unchanged, but causing the derivatives across boundaries to vary in any arbitrary way we wish, as we move along the boundary.

The equation resembles the first form very strongly. It is

$$(uw) = (iw)_u G_i(u) + (uj)_w G_j(w) - (ij)_{uw} G_i(u) G_j(w).$$

Here, typically, $(iw)_u$ is a function of w only, and describes the arbitrary variation of the derivative with respect to u as w varies along the curve (iw), and similarly for the other boundaries. The vector $(ij)_{uw}$ represents the cross derivatives of the four corners. Typically,

$$(00)_{uw} = \frac{\partial^2 (uw)}{\partial u \, \partial w} \qquad \begin{matrix} u = 0 \\ w = 0 \end{matrix}$$

The functions G_0 and G_1 are again blending functions or weighting functions, but they have properties different from the functions F_0 and F_1. We stipulate

$$G_i(a) = 0, \; a \text{ and } i = 0 \text{ or } 1.$$

$$G_i'(a) = 0, \; a \neq i.$$

$$G_i'(a) = 1, \; a = i.$$

The proof that the vectors describing the boundaries vanish identically, and that the vectors describing the slope variation along boundaries are indeed given by the equation proceeds along precisely the same lines we used before.

Analogous forms may be obtained for correction of higher derivatives along boundaries. For second derivative correction, the surface equation is

$$(uw) = (iw)_{uu}H_i(u) + (uj)_{ww}H_j(w) - (ij)_{uuww}H_i(u)H_j(w).$$

In this equation, the blending functions H_i have the stipulations that, for a = 0 or 1 as before,

$$H_i(a) = 0$$

$$H_i'(a) = 0$$

$$H_i''(a) = 0, \quad i \neq a$$

$$H_i''(i) = 1, \quad i = a.$$

With these constraints on the H_i, it is easy to arrange matters so that this second-order correction surface is zero everywhere on the boundary, has zero slopes across boundaries, and has second derivatives across boundaries specified by $(iw)_{uu}$ and $(uj)_{ww}$ whatever these functions may be. The addition of this surface vector to a given surface vector will then provide a means for boundary second-derivative correction without disturbing either the boundary shapes or boundary slopes.

6. Matrix Form

The surface equation

$$(uw) = (iw)F_i(u) + (uj)F_j(w) - (ij)F_i(u)F_j(w)$$

by virtue of the summation convention may be expanded directly into matrices, to yield:

$$(uw) = [u0 \quad u1] \begin{bmatrix} F_0{}^w \\ F_1{}^w \end{bmatrix} + [F_0{}^u \quad F_1{}^u] \begin{bmatrix} 0w \\ 1w \end{bmatrix}$$

$$- [F_0{}^u \quad F_1{}^u] \begin{bmatrix} 00 & 01 \\ 10 & 11 \end{bmatrix} \begin{bmatrix} F_0{}^w \\ F_1{}^w \end{bmatrix}$$

(We omit parentheses, since no misunderstanding can arise. Thus typically F_0u is written in place of $F_0(u)$ as a matter of convenience and economy); which in turn is equivalent to

$$(uw) = [1 \quad F_0{}^u \quad F_1{}^u] \begin{bmatrix} 0 & u0 & u1 \\ 0w & -00 & -01 \\ 1w & -10 & -11 \end{bmatrix} \begin{bmatrix} 1 \\ F_0{}^w \\ F_1{}^w \end{bmatrix}$$

It is slightly more convenient to rewrite this in the equivalent form

$$(uw) = - [-1 \quad F_0{}^u \quad F_1{}^u] \begin{bmatrix} 0 & u0 & u1 \\ 0w & 00 & 01 \\ 1w & 10 & 11 \end{bmatrix} \begin{bmatrix} -1 \\ F_0{}^w \\ F_1{}^w \end{bmatrix}$$

so as to avoid the awkward minus signs in the 3 x 3 matrix.

Two facts should be noted. The leading row vector in front of the matrix and the trailing column vector following the matrix are transposes of one another, but with different arguments; the matrix represents the boundary conditions of a patch. The partition $\begin{bmatrix} 00 & 01 \\ 10 & 11 \end{bmatrix}$ is redundant, since its elements must agree with uj and iw for u and w equal to 0 or 1.

We have already suggested that we can maintain slope continuity across boundaries by suitable stipulations on F_i, and we have also already suggested that when desired we can adjust slopes across boundaries by a second additive vector with suitable stipulations on its G_i. We shall now investigate the combined form of the surface equation. To do so we shall prefix a symbol to the vector uw to indicate whether we are talking about the first surface equation, or the correction surface

equation, and we shall omit the prefix symbol when we are talking about the combined form. Thus

$$\begin{aligned} uw &= suw + cuw, \text{ with} \\ suw &= \text{the primary surface} \\ cuw &= \text{the correction surface} \\ uw &= \text{the combination.} \end{aligned}$$

Accordingly, using this notational convention, we will take derivatives, with respect to u, of the surface equation suw in order to determine its slope vector in the u direction.

We obtain the complete matric expression for the correction surface:

$$cuw = -\left[-1\ F_0u\ F_1u\ G_0u\ G_1u\right] \left[\begin{array}{c|c|c|c|c} 0 & 0 & 0 & u0_w & u1_w \\ 0 & 0 & 0 & 00_w & 01_w \\ 0 & 0 & 0 & 10_w & 11_w \\ \hline 0w_u & 00_u & 01_u & 00_{uw} & 01_{uw} \\ 1w_u & 10_u & 11_u & 10_{uw} & 11_{uw} \end{array}\right] \left[\begin{array}{c} -1 \\ F_0w \\ F_1w \\ G_0w \\ G_1w \end{array}\right]$$

If now we border the original surface equation matrix, it can be written,

$$suw = -\left[-1\ F_0u\ F_1u\ G_0u\ G_1u\right] \left[\begin{array}{c|c|c|c|c} 0 & u0 & u1 & 0 & 0 \\ 0w & 00 & 01 & 0 & 0 \\ 1w & 10 & 11 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} -1 \\ F_0w \\ F_1w \\ G_0w \\ G_1w \end{array}\right]$$

In this bordering process, the value of the matric product is unchanged.

Since the pre- and post-multiplicative matrices in this equation are the same as those of the correction surface equation, we can add the two 5 x 5 matrices and pre- and post-multiply by the two vectors. We shall perform, in fact,

uw = suw + cuw, and obtain

$$uw = -\left[-1\ F_0u\ F_1u\ G_0u\ G_1u\right]\left[\begin{array}{c|cc|cc} 0 & u0 & u1 & u0_w & u1_w \\ \hline 0w & 00 & 01 & 00_w & 01_w \\ 1w & 10 & 11 & 10_w & 11_w \\ \hline 0w_u & 00_u & 01_u & 00_{uw} & 01_{uw} \\ 1w_u & 10_u & 11_u & 10_{uw} & 11_{uw} \end{array}\right]\left[\begin{array}{c} -1 \\ F_0w \\ F_1w \\ G_0w \\ G_1w \end{array}\right]$$

This is a general expression for a slope-matching, slope continuous surface patch with entirely arbitrary boundaries and entirely arbitrary slopes across these boundaries. There are no stipulations whatever on the nature of the boundary slope functions. The stipulation on the F and G functions have already been discussed.

Similarly the surface equation

$$uw = -\left[-1\ F_0u\ F_1u\ G_0u\ G_1u\ H_0u\ H_1u\right]$$

$$X\left[\begin{array}{c|cc|cc|cc} 0 & u0 & u1 & u0_w & u1_w & u0_{ww} & u1_{ww} \\ \hline 0w & 00 & 01 & 00_w & 01_w & 00_{uww} & 01_{ww} \\ 1w & 10 & 11 & 10_w & 11_w & 10_{ww} & 11_{ww} \\ \hline 0w_u & 00_u & 01_u & 00_{uw} & 01_{uw} & 00_{uww} & 01_{uww} \\ 1w_u & 10_u & 11_u & 10_{uw} & 11_{uw} & 10_{uww} & 11_{uww} \\ \hline 0w_{uu} & 00_{uu} & 01_{uu} & 00_{uuw} & 01_{uuw} & 00_{uuww} & 01_{uuww} \\ 1w_{uu} & 10_{uu} & 11_{uu} & 10_{uuw} & 11_{uuw} & 10_{uuww} & 11_{uuww} \end{array}\right]\left[\begin{array}{c} -1 \\ F_0w \\ F_1w \\ G_0w \\ G_1w \\ H_0w \\ H_1w \end{array}\right]$$

represents a surface patch whose vectors of coordinates, slope, and curvature are everywhere arbitrary along its boundaries. The first column and first row of the 7 x 7 matrix represent these boundary conditions; the remainder of the matrix is redundant, since the quantities this partition contains must all come from the column and row by differentiation.

CHAPTER X

COMMUNICATION WITH INDUSTRY AND OTHERS

A. INTRODUCTION

As the preceding chapters have indicated, the work of the MIT Computer-Aided Design Project is not pure research nor pure development, nor merely applied research. It can perhaps be best characterized as goal-oriented research and development. The Project mandate and the very nature of the work itself has required that much of the effort be directed toward abstract theoretical areas with much of the flavor of pure research. At the same time, however, even the most abstract considerations are attacked with the intent that the ultimate resolution will be sound, practical, and economical for direct application in the rigors of the industrial and scientific environment. Frequently, this balancing of generality with practicality forces intermediate layers of complexity to appear in the work of the Project. It is almost a truism that suitable idealization will allow abstract and general things to be done simply. Similarly, sufficient specialization and elimination of flexibility will allow practical things to be done simply. But for the kinds of problems with which we are concerned, the simultaneous achievement of generality and practicality without complexity is impossible.

In order for the work of the Project to be useful to the industrial and scientific community, it is necessary that our approach to general man-machine problem-solving be understood in some detail. Since many Project activities break new ground and require approaches quite different from those commonly used, a large and increasing amount of Project effort must go into the preparation of technical presentations and documents describing the approach and the work. The large number of references cited in the Appendix to this report give testimony to the importance which we attach to this vital aspect of the Project mandate.

Through past experience in the launching of the APT Language and System for programming of numerically-controlled machine tools, and in other Laboratory efforts in which advanced research results are meaningful and worthwhile only when they are applied by others, we early became convinced that we could not rely upon ordinary documentation and reporting to promote effective transfer of results into the application environment. In addition to ordinary documentation, pilot projects and worked-out examples must be carried through to completion in order to demonstrate "reduction-to-practice" and to provide stimulus and guidelines for further applications. Various theses and specialized application system developments have been intended to initiate the fulfillment of this requirement, but much further work remains to be done.

Even documentation and examples are not sufficient, however, for the complex nature of the work of the Project makes it virtually impossible (at least at the early stages) to capture all of the essential information and ideas in concrete form. Throughout this initial phase of the Project, and for several more years as well, the "culture" which accompanies the use of the ideas and techniques plays an essential role, and this important ingredient can only be transmitted through direct experience and intimate collaboration. Therefore, another important activity of the Project in the area of communication and education has been a variety of cooperative arrangements with potential users of Project results. These cooperative programs have been arranged on a direct, informal basis, with numerous groups and projects at M. I. T., and with industry and outside organizations, through the AED Cooperative Program of the Project. Such joint ventures both serve the desired information transfer role and also provide important feedback to the Project on the efficacy of the work in various environments and applications.

The participation in the work of the Project by the Design Division of the MIT Department of Mechanical Engineering has been reported in Chapter IX. The most intimate current association of the Project is with Project MAC at M. I. T. Project MAC is a large Institute-wide project sponsored by the Advanced Research Projects Agency through the Office of Naval Research. This work, under

Contract NOnr-4102(01), is intended to study, develop, and foster man-machine problem-solving. The acronym MAC can be considered to come from Machine-Aided Cognition, indicating the broad goal of the Project, or from Multiple Access Computer, referring to the time-shared use of computer as a means toward achieving the broader goals. Project MAC has a very large IBM 7094-based time-shared computer system in successful operation on a 24-hour-per-day, 7-days-per-week-basis, with short down times for maintenance and collection of backup records. Over 200 typewriter-like consoles scattered around the Institute community provide remote access to the central computer through a telephone switching system. As has been reported in Chapter VII, the ESL Console of the Computer-Aided Design Project is attached to the central computer through a PDP-7 buffer computer. Project MAC forms an excellent example of the type of intimate working relationship which is appropriate for the Computer-Aided Design Project at this stage. The long-term goals of the two Projects are very closely related, and the differences in approach to common problems form fruitful ground for much valuable cross fertilization.

Other associations with various MIT groups have been in the nature of helping them use AED systems in techniques in their own work. These applications are listed in Appendix I.

Remaining sections of this chapter discuss the AED Cooperative Program, Special Meetings and Symposia, and Documentation by Movies.

B. AED COOPERATIVE PROGRAM

The goal of the Computer-Aided Design Project and our Air Force sponsors is to transfer completed portions of research results to industry as rapidly as possible, and to assist in field-trial application to practical and important problems. To provide a vehicle for this transfer of results, the Project maintains the AED Cooperative Program in which visiting staff members from outside organizations are accepted into the Project for a one-year term, to learn about and participate in the on-going activities. Such cooperative liaison is of value to both the individuals and organizations who accept the invitation, and

also helps to insure that the subsequent evolution of AED will be pertinent to the problems of industry. By active participation in the work itself, the industry representative at M. I. T. becomes skilled in the needed techniques being developed for the AED System, and learns many aspects which are impossible to document at present. Furthermore, working programs developed in the Project are released immediately so that additional programmers in the company plants also can gain experience with the new techniques. Continual liaison between the M. I. T. Project and company personnel by direct communication, progress reports, and technical documentation, as well as meetings and visits, allows efficient transfer of Project results to industry with important two-way benefits.

1. The Cooperative AED-1 Project

Cooperative activity with selected industrial organizations was begun following an invitation extended by the Project in December, 1963. In March, 1964, the regular staff of the Computer-Aided Design Project was augmented by experienced system programmers nominated for a one-year tour of duty by seven U.S. companies who accepted the invitation to participate in what was then named the "Cooperative AED-1 Project".

When the Cooperative AED-1 Project was set up early in 1964, it was emphasized that it would be a research rather than a development effort. Nonetheless, the plans at that time were essentially to use the then-existing AED-0 capabilities to reduce to practice the various proposed AED-1 features. It was hoped that an initial version of an AED-1 System embodying those features could be put together in approximately one year's time. As the work progressed, however, it became apparent that the modest additions to AED-0 which had been envisioned originally should be augmented. As a result, many of the proposed AED-1 features were transformed into features of the expanded AED-0 System.

At an Open House, held at M. I. T. in late August, 1964, for representatives from the cooperating companies, the changed complexion of the Project was gone over in some detail, with a primary emphasis on the reasons behind the decision to delay work on AED-1 proper

in order to strengthen the AED-0 base. The major consideration was the fact that AED-1 would not evolve from AED-0 in any direct way, but instead AED-0 was to be used as the tool to construct AED-1 from the beginning. Thus, the more ambitious the expectations became for the AED-1 System, the more stringent the requirement for the AED-0 tool.

An additional consideration was the fact that extreme difficulties were being encountered in the task of preparing a version of the original AED-0 System for distribution to the cooperating companies as a batch-processing system for the IBM 709-7090-7094 computers. Many of these unexpected problems were primarily triggered by the fact that the high modularity of the AED-0 compiler involved larger numbers of distinct subroutines than were usually encountered. In order to handle these, it was necessary to carry out modifications to the master tape generating programs of the standard Fortran Monitor System, which was being used as the batch-processing vehicle, and many of these features of the FMS System were incompletely documented. Although preliminary success was achieved by October, 1964, and an old version of the AED-0 compiler was distributed to those companies who had requested copies, the many changes made to AED-0 in the meantime, combined with the shortcomings of the distributed batch processor, indicated that efforts should be continued to obtain a cleaned-up and more-powerful version of AED-0 for use by the companies.

In December, 1964, it was decided to schedule the preparation of the first definitive version of an AED-0 batch-processing system by March, 1965. The March batch-processing system encompassed, insofar as possible, all of the features of the AED-0 System used in the MIT Time-Sharing environment, and all of the most recent advances in AED-0. Thus the dissemination of AED techniques through a nucleus of technical people in each company, using and experimenting with the system, could begin in earnest.

At the conclusion of this first cooperative effort, in March, 1965, the following results had been achieved:

1. A greatly expanded AED-0 System, embodying the most powerful high-level system programming language available, was completed in both time-sharing and batch-processing versions, and distributed to interested companies.

2. Working versions of the AEDJR System, which allows specialized problem-oriented programming languages to be defined and put to use, were completed and distributed.

3. The design of the still more powerful machine-independent AED-1 System was blocked out, and many major building blocks were completed.

4. Results of the Project began to be applied in company projects, both through direct application of the distributed systems, and through the use of the underlying programming techniques in other forms.

5. Perhaps most important, a growing number of talented individuals acquired new insights and techniques for attacking the pressing problems of computer-aided design.

The demonstrated workability of this first cooperative program led to incorporation of this type of industry cooperation as an integral part of the Air Force contract supporting the activities of the MIT Computer-Aided Design Project.

2. The AED Cooperative Program

A second invitation was extended to selected organizations to participate in the continuing but renamed AED Cooperative Program in mid-1965. Companies invited were believed to be well advanced in their own appreciation of the potentialities and problems of computer-aided design. Many already were taking the necessary steps to prepare for the new technology to ensure they would benefit from the new developments. During this second period, the prime focus was on the new AED-1 Compiler System, in preparation for bootstrapping AED to "third-generation" computers and subsequent design studies. The second group of five visitors started in July, 1965, and stayed through July, 1966.

The 1966 AED Cooperative Program was initiated by inviting companies to send an experienced system programmer to M. I. T. to

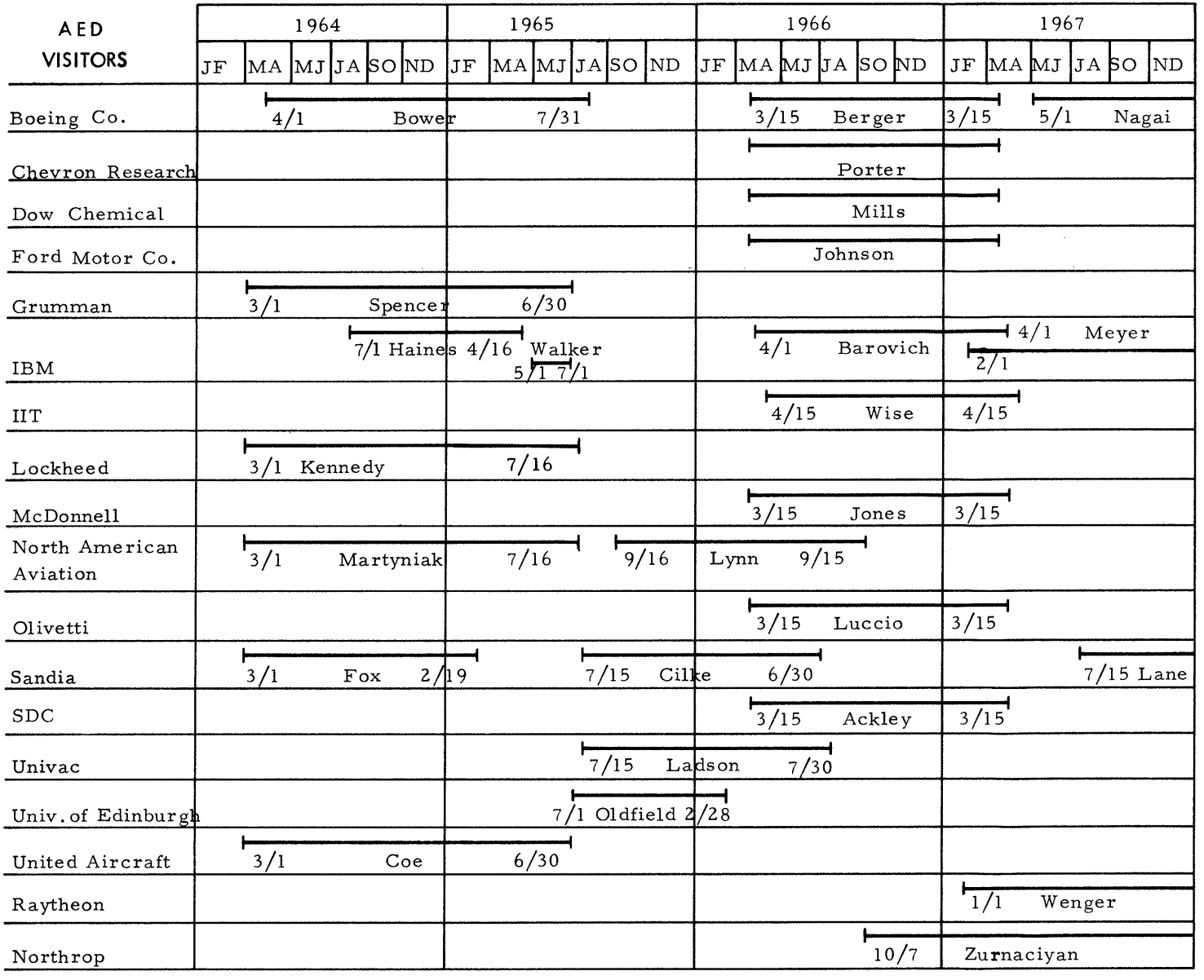


Fig. 31 AED Visitors

work with the regular Project staff for a nominal one-year period beginning in March, 1966. Nine companies accepted the invitation, and during that year of joint activity the first pass of AED-1 received major emphasis, resulting in a new RWORD and AEDJR capability. Many integrated packages also were improved, and new packages were developed in preparation for bootstrapping all AED results to third-generation computers by means of the machine-independent AED-1 Compiler.

Following the 1966 AED Cooperative Program we have shifted to an open-door policy regarding nominations for new visiting staff. In a letter dated June 19, 1967, the following paragraph described the new plan: "With this mailing, we are discontinuing the 'Invitation Mailing List' for the AED Cooperative Program. The Program will remain active, but we do not plan to issue further formal invitations. However, any company that would like to send a visiting staff member in the future is invited to write me at any time to determine our current status and to work out a mutually satisfactory arrangement. The reason for this change is that the various company activities have evidently now matured to the point where coordinated schedules for groups of visiting staff are inappropriate. We hope that many of the companies who have expressed future interest will take advantage of our continuing 'open-door' policy."

Companies which do not participate directly in the AED Cooperative Program can nonetheless obtain copies of system releases and documents so they may experiment with AED-0 and AEDJR on their own problems. Copies of all system releases are available for use on the IBM 709, 7090, 7094 and System 360 computers, as well as the Univac 1108 computer. We hope that growing expressions of interest on the part of users will enable AED to similarly be bootstrapped and made available on all major computers in the next few years. We have extended a great deal of effort in making the bootstrapping process as systematic as possible to provide a basis for these expected developments.

Figure 31 shows the pattern of participation in the AED Cooperative Program to date. Full names and company affiliation for these Visiting Staff Members are listed in the Personnel List, page v.

C. SPECIAL MEETINGS AND SYMPOSIA

Several special meetings have been organized to present information about the Project, its work, and the potential of computer-aided design as a powerful new component of advanced industrial and problem-solving technology. The following sections describe these meetings in chronological order. The AED Technical Meetings are held at M. I. T. as part of the AED Cooperative Program but with open invitation.

1. MIT/ILO Symposium on Computer-Aided Design

In May, 1963, a symposium on "Computer-Aided Design" was held at the Kresge Auditorium, Little Theater, M. I. T. The symposium was sponsored by the Industrial Liaison Office of M. I. T. for attendance by those companies who are members of the Industrial Liaison Program. The symposium was attended by 201 people representing 45 companies from the United States. According to the Industrial Liaison Office, this was the largest attendance of any meeting in the history of the MIT Industrial Liaison Program up to that time. Considerable interest was expressed by the attendees and it was agreed that the symposium was very successful and worthwhile. The program included the following presentations:

WELCOME	Professor Peter Elias, Head Department of Electrical Engineering
BACKGROUND OF THE PROJECT	Professor J. Francis Reintjes, Director Electronic Systems Laboratory Department of Electrical Engineering
CRITERIA FOR THE COMPUTER-AIDED DESIGN SYSTEM	Professor Robert W. Mann, In Charge Engineering Design Division Department of Mechanical Engineering
THEORETICAL FOUNDATIONS	Mr. Douglas T. Ross, Head Computer Applications Group Electronic Systems Laboratory
GRAPHICAL COMMUNICATIONS AND PROBLEM SOLVING	Professor Steven A. Coons Engineering Design Division Department of Mechanical Engineering

STRESS ANALYSIS TECHNIQUES	Professor Frank A. McClintock Materials Division Department of Mechanical Engineering
STRUCTURE AND OPERATION	Mr. Douglas T. Ross
MAN-MACHINE CONSOLE FACILITIES	Mr. John E. Ward, Ass't. Director Electronic Systems Laboratory Department of Electrical Engineering
SUMMARY AND GENERAL DISCUSSION	Mr. Douglas T. Ross Professor Steven A. Coons

2. 1963 Spring Joint Computer Conference Session

Also in May, 1963, a special session on "Computer-Aided Design" was included in the program of the 1963 Spring Joint Computer Conference, sponsored by the American Federation of Information Processing Societies at Cobo Hall in Detroit, Michigan. Attendance at the special session was high and considerable interest was generated in the material presented. At the Conference Luncheon the paper by Ross and Rodriguez received the American Federation of Information Processing Societies Prize Paper Award, "Awarded in recognition of an outstanding and significant contribution to the Information Processing Field through presentation of the best paper at the 1963 Spring Joint Computer Conference". The program included the following presentations:

An Outline of the Requirements for a Computer-Aided Design System, S. A. Coons, Mechanical Engineering Department, M. I. T.

Theoretical Foundations for the Computer-Aided Design System, D. T. Ross and J. E. Rodriguez, Electronic Systems Laboratory, M. I. T.

Man-Machine Console Facilities for Computer-Aided Design, R. Stotz, Electronic Systems Laboratory, M. I. T.

Sketchpad: A Man-Machine Graphical Communication System, I. E. Sutherland, Consultant, Lincoln Laboratory, M. I. T.

Sketchpad III: A Computer Program for Drawing in Three Dimensions, T. E. Johnson, Mechanical Engineering Department, M. I. T.

3. MIT/ILO Symposium on Project MAC

In May, 1964, the MIT Industrial Liaison Office offered another symposium in which the Project was represented. As part of the presentation by large users of the Project MAC facilities, D. T. Ross and C. G. Feldmann presented a paper and demonstration entitled "Verbal and Graphical Language for the AED System: A Progress Report" (see Ref. 27) in which the initial AEDJR System was described for the first time, including the definition of a simple language and a description of the role played by parsing in the AED compiling scheme.

D. T. Ross, C. A. Lang, and R. B. Polansky also presented the "May 6 Demonstration Package", the first demonstration of processing graphical language with the Algorithmic Theory of Language, set up using the AEDJR System. The demonstration used large-screen video tape of action at the ESL Display Console.

J. E. Ward and R. H. Stotz also made a presentation on the ESL Display Console and how it was used in the time-sharing environment.

4. The First AED Technical Meeting

The First AED Technical Meeting was held at M. I. T. June 22-23, 1966, with 105 people (54 from 24 organizations plus 51 from various MIT groups) in attendance. The first morning was devoted to the structure and general features of the AED-1 Processor. The first afternoon covered the use of the AEDJR System for constructing problem-oriented systems, discussions concerning distribution of AED results, and the possibility of forming an AED User Group. On the second day, the morning was devoted to a discussion of computer graphics, and in the afternoon various demonstrations and general discussions were held. From the comments of those in attendance, the meeting fulfilled its aims quite satisfactorily. The few companies who had made extensive use of AED gave highly complementary reports.

5. The Second AED Technical Meeting

The Second AED Technical Meeting was held at M. I. T. in Kresge Auditorium, January 25-27, 1967, with over 349 people (312 from 104 organizations plus over 37 people from various MIT groups)

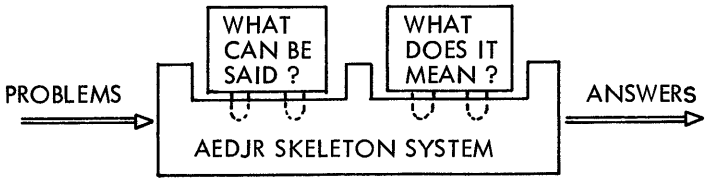
in attendance. The program included brief summaries of technical progress on the AED-1 System and prognoses for future developments, as well as another attempt to determine whether an AED User Group should yet be organized. The main emphasis of the meeting, however, was a workshop on the use of the AEDJR System and AED-0 language for making specialized user-oriented systems efficiently. The workshop session and demonstrations were carried out using the on-line time-sharing facilities of Project MAC with closed circuit television coverage of actions at the teletypewriter console and the first use of the ARDS remote storage display unit. The meeting is best described by the meeting notice and the informal meeting report, which are reproduced below.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Electronic Systems Laboratory
COMPUTER-AIDED DESIGN PROJECT
Cambridge, Massachusetts 02139

NOTICE

THE SECOND AED TECHNICAL MEETING

Little Theater
M. I. T., Kresge Auditorium
9AM-5PM, Wednesday - Friday
January 25-27, 1967



The M. I. T. Computer-Aided Design Project is pleased to announce the Second AED Technical Meeting to be held at M. I. T. Wednesday through Friday, January 25-27, 1967. The meeting will be of particular value to system programmers, computer-oriented designers and engineers of various disciplines, and managers of same. The primary purpose of the meeting will be to demonstrate that specialized problem-oriented computer-aided design systems can be created NOW at a fraction of the customary investment, by using the facilities of the AEDJR System and the AED-0 Language. Most of the meeting will be an open Workshop in which a complete, functioning, problem-oriented design system will be designed, programmed, debugged and used, live and in real time, with audience participation. Attendees will be able to propose system features and to discuss fully the techniques being used. Each facet of the Workshop design problem will be compared with similar features of industrial

applications of computers in design in order to emphasize the immediate applicability of this new system-building technology. The Workshop will use the on-line time-sharing facilities of Project MAC, with closed circuit television coverage of actions at typewriter consoles. Further details are on the attached sheets.

A progress report on the AED-1 System (with demonstrations), and discussion of time-tables for the AED Cooperative Program in 1967 will be included in the agenda for the first day.

Registration will be limited to the capacity of the Little Theater, and preference will be given to personnel from present, past, and future participants in the AED Cooperative Program for pre-registrations received before January 13. Others will receive preference in the order in which their registrations are received. Notes and documents for the Workshop will be sent to accepted registrants January 16-18, 1967.

Second AED Technical Meeting
January 25-27, 1967 Little Theater, M.I.T.

ELABORATION

1. Purpose of the AEDJR Workshop

In the Second AED Technical Meeting we are de-emphasizing computer graphics and sophisticated machine-independent programming language techniques in order to give compelling emphasis to the man-machine problem-solving process, which is the sole reason for their interest and importance. We are concerned this time with increasing the utility and reliability of existing applications of computers in the design process by improving the man-machine interface through specialized user-oriented languages in which the man communicates with the machine in "shop-talk" language uniquely appropriate to his design area. There are innumerable existing computer applications in industry which fall into this category, and provide a handsome opportunity for an evolutionary rather than revolutionary introduction of true computer-aided design into the industrial scene. This approach offers significantly greater economic advantages than the more glamorous ad hoc crash programs in new and untried gaudy-gadget computer-graphics applications. Furthermore, since the AEDJR System used in the workshop is the same as is used in the CADET System, further future advances to incorporate computer graphics will be a natural and evolutionary step.

The AEDJR Workshop of the Second AED Technical Meeting represents a challenge to industry. We are in effect, throwing down the gauntlet and saying to industry, "If we can in a matter of days create a working problem-oriented computer-aided design system for a simple application area, why cannot industry do the same in a matter of months with real design problems which they have in abundance?" The application area on which the workshop will be based, and the sophistication of the resulting system, will not be the important message of the workshop. Instead, the methods by which the system is created as a direct and orderly application of the AED approach are the focal point of the workshop. We hope that many workshop attendees will see the strong analogy between the workshop example and design areas in their own companies in which they already are using computers in the design process, and

that they will be stimulated to ask, "Why not indeed?", and will begin in earnest on the learning process to acquire the necessary skills with the new system-building technology which AEDJR provides.

The AEDJR Workshop will not be a sterile series of lectures on the details of the AEDJR System. It is not expected that attendees will be able to leave the workshop and set about immediately constructing problem-oriented systems in the manner demonstrated; for even with the powerful assistance of AEDJR, system-building is a complicated and exacting task. The objective of the workshop is to present a complete picture in depth of the AED approach to system-building for computer-aided design, opening all the doors and peeking into every nook and cranny of the various black boxes that constitute the AEDJR System in order to demonstrate that the beginnings of a new-organized, dependable technology for software development are at hand. The experience of the workshop will provide valuable background for subsequent study of the working tools and methods of the AED approach. Our objective is not teach a skill, but to impart a new level of understanding and a deeper and technically better-justified vision of the implications and imminence of meaningful computer-aided design.

2. The AED Approach to User-Oriented Systems

The general scheme for systematically deriving a sophisticated and smoothly functioning user-oriented system from an existing application of computers, using the AEDJR System, may be outlined as follows:

We begin with some class of problem which we will consider to be the area of discourse for the man-machine system to be constructed. We first prepare a set of functions or procedures which constitute an integrated package, called the semantic package, containing all of the necessary building blocks for constructing the solution to any problem within the area of discourse. These are not just ordinary procedures, for they are specifically designed to give the finest necessary sub-division of the various features which compose the area of discourse. (If we begin with already-existing programs, some modification may be needed to achieve the necessary modularity.) Each procedure is an atomic procedure in the sense that it treats some smallest indivisible piece of the problem area. The atomic procedures are written such that the value given by one procedure may be used as an input argument to another procedure. In this way, large molecular procedures may be constructed from the atomic procedures by assembling the proper nested calls. Some of the procedures in the package are concerned with modelling or representing the detailed structure and constituents of any particular problem in the area of discourse. Other atomic procedures have to do with analyzing or computing values for various aspects of a particular problem.

In general, in order to solve a particular problem in the area it is necessary to call upon the proper collection of model-building procedures until the problem statement is complete, and then to assemble the appropriate analysis procedures to apply to that model to arrive at a solution. Many problems iterate back and forth between these steps, using the results of some sub-analysis to modify or augment the model for a succeeding phase of analysis. The important point about the semantic package

is that it effectively delimits and bounds the area of application, so that by definition any problem within the area of discourse can in fact be stated and solved.

The problem then becomes that of matching the user to the problem-solving system, i.e., we next must design one or more user-oriented languages which can be translated into the appropriate calls on the procedures in the semantic package. It is at this point that the AEDJR skeleton system provides the necessary framework for defining the language and integrating its translation with the integrated package of procedures which define the area of discourse.

In general, the atomic routines of the package must be applied in a very rigid sequence in order to accomplish given tasks. Differing sequences of assembling atomic procedures into molecular compound actions will in general give differing results. The sequence imposed by the structure of the semantic package may be quite different from the sequence which is natural for the human problem-solver to think of the specification of a problem. A key feature of the powerful parsing algorithm which is incorporated in the AEDJR System is that there is a direct way to transform the natural sequence of the human thought process into the required sequence imposed by the semantic package. Thus problem-oriented languages to control the use of the semantic package to solve problems in the area of discourse can be made to seem very natural and easy to use. (The First-Pass Algorithm performs a semantic as well as a syntactic parse, and it is possible to control the behavior of the "precedence string" which represents the semantic parsing structure needed for the transformation.)

The AEDJR System incorporates a special command language by means of which the vocabulary and linguistic rules for a new language may be defined. The First-Pass Algorithm which performs parsing permits almost all desired linguistic features to be carried out elegantly and efficiently as a straight-forward application of the features of the Algorithmic Theory of Language. Wherever the basic algorithm will not perform the desired function, however, "execute programs" may be incorporated to momentarily override the standard behavior to achieve the desired result. Once facility with the language-definition language is achieved, new languages may be defined rapidly and in an orderly fashion. The AEDJR System also provides special facilities for debugging and modifying language definitions, and once a suitable definition is achieved, automatic provisions allow dumping of the definition tables for subsequent reloading with the basic First-Pass Algorithm core to create a system for production use.

One final feature which is necessary to ensure that a language is natural and user-oriented, is control over the physical format of expressions in the language. The AED RWORD Package provides an elegant solution to the problem of permitting the user to have complete control over the format of a new language and performing with great efficiency the intricate lexicographic analysis which must precede parsing. A special language (a subset of the language of regular expressions) permits natural description of the rules whereby sequences of characters cluster together to form the items (syllables, words, or punctuation) of the language input form. The regular expressions describing the various item types are processed by an automatic system

to create a specialized reading program which may be incorporated directly into the production system to provide input to the parsing algorithm controlled by the language definition tables.

3. Specific Topics to be Covered in the Workshop

1. How to structure a collection of subroutines so that they form a suitable "semantic package" of atomic routines for an area of discourse.

2. Discussion of the sequence of atomic actions imposed by the structure of the semantic package.

3. Discussion of various sequences which seem natural for problem statement, showing how all necessary information is supplied, but in a different sequence than that required by the semantic package.

4. How to use left-to-right, right-to-left, and "modifier" parsing to control the precedence string sequence of a parsed statement to map the desired sequence into the required sequence.

5. How to use the "dun" bits to control back-up and error detection in parsing.

6. How to use "execute" programs to control construction and activation of symbol-table structures and to perform unusual parsing.

7. How to use the AEDJR "mouse" to translate parsed statements into semantic package actions.

8. How to construct a complete "modeling plex" from sequences of statements to represent the total problem in the computer.

9. How to write "regular expressions" which describe the desired item-structure of an input string for presentation to the parsing algorithm.

10. How to use the "Assemble" Package to provide free-format output of answers in user-oriented terms.

11. How to use the generalized Alarm Package to efficiently report errors in user-oriented terms.

12. How to design a language which is rich in possibilities, in which definition, analysis, and control features may be used in natural combinations.

The AED Cooperative Program is a major activity of the M. I. T. Computer-Aided Design Project, sponsored by the Manufacturing Technology Division of the U. S. Air Force through Contract AF-33(657)-10954 with the M. I. T. Electronic Systems Laboratory. The Project also uses facilities of Project MAC, sponsored by the Advanced Research Projects Agency of the Department of Defense under Contract NOnr-4102(01). The "AED Cooperative Program" refers to those aspects of the overall M. I. T. Computer-Aided Design Project which are sufficiently developed to merit industry participation.

The results of the Second AED Technical Meeting were reported in AED Technical Progress Report No. 17. The following is an extract from this report which was issued on April 3, 1967.

AED PROGRESS REPORT NO. 17
November 1, 1966 - March 31, 1967

Second AED Technical Meeting

"It is a safe bet to say there will never be another quite like it. The Second AED Technical Meeting was held at M. I. T., January 25-27, 1967, but not quite as scheduled. For those who were unable to attend, the following synopsis is presented.

"Based upon attendance at the First AED Technical Meeting, we had scheduled the use of the Little Theater at Kresge Auditorium with a seating capacity of 214, and the meeting notice was worded such that registration for the meeting would be restricted to that number. Toward the middle of January, however, when pre-registrations already exceeded that capacity and still many companies which would have priority for seats had not yet been heard from, we had to make our first decision. Rather than disappoint large numbers of interested people we decided to move the proceedings from the Little Theater to the Main Auditorium upstairs, and impose no restrictions on registration. Although we are sure this was the proper decision, once the meeting went from fewer than 200 to over 400 registrants, any semblance of "workshop" became impossible. The problem was compounded because a majority of the unexpected registrants evidently lacked the requisite background familiarity with the aims and objectives of AED. (Some acknowledged that they had not even read the meeting announcement, but had been dispatched by a supervisor.) Since the material that had been prepared for the meeting was geared toward an intensive pace even for those well prepared, we were faced with a rather ticklish situation.

"The sharp division in backgrounds, preparation, and interests in the attendees was not immediately apparent, but gradually became quite obvious by the end of the "progress report" segments of the agenda of the first day. The progress report segments on the RWORD System, dumping and restoring first-pass structures, and the status of the AED-1 Compiler were intended merely to be brief demonstrations to show that progress was being made as a preamble to the AEDJR Workshop. Evidently this intended objective was not expressed clearly enough to be understood, so that considerable confusion resulted when people could not follow what was going on. The demonstrations themselves probably were not concise enough, as well. As a result, questions about the over-all objectives and relevance of our whole program were raised repeatedly, and only incompletely answered.

"In order to attempt to establish a mechanism for satisfying the needs of the technical and non-technical parts of the audience, we split up into several groups on the second day. Technical working groups on AEDJR, Language Design, Second Pass, AED for the 1108, and AED-0 Distributions met under the leadership of senior Project staff members (who performed admirably in these unscheduled roles) while the non-technical group continued discussions in the main auditorium. Although things were still rather hectic, this separation of conflicting interests vastly improved the ability

to communicate. From the remarks received, almost everyone agreed that we had done the best we could possibly do with a difficult situation, and it appeared that a considerable majority were well satisfied with the results. In the middle of the afternoon of the second day, we purposefully shifted the meeting into technical high gear with apologies to the non-technical segment in order to proceed with the Workshop. Although attendance did drop off on the third day, a gratifyingly high percentage of the non-technical attendees did remain and were glad they did, for even though technical details eluded them, they were able to sense and appreciate the unexpected and significant results of the workshop portion of the meeting which were achieved in spite of the chaotic circumstances.

"As originally planned, the AEDJR Workshop portion of the Second AED Technical Meeting was intended to demonstrate how existing applications of computers in design, constituting a problem-oriented system, could be made over into a user-oriented system using the AEDJR facilities. The example which we selected as a demonstration vehicle was that of composing large structures out of smaller structures by joining polygonal faces along their edges. This example was chosen because it was simple to describe in loose terms, and because the appropriate modeling aspect of the problem consisted almost entirely of pure structure, the only universal feature of all problem formulation. In early December when preparations for the meeting began, we intended merely to define a few string types using the Generalized String Package to form the basis for the semantic package of programs which would represent the "existing use of computers" for purposes of the meeting. Work on the graphical input to the CADET System was tabled to prepare these programs, and as the "Polyface Package" (as the polygonal face example became known) progressed, we found that instead of merely being a collection of quick-and-dirty programs for use in the meeting, we were developing a generalized approach to the universal modeling feature of the CADET System. Our plans for the Polyface Package thereupon became much more ambitious as we sought to incorporate rigor and generality for subsequent use in CADET. By the time of the meeting at the end of January, an extremely complex and intricate set of programs was only partially debugged. In fact debugging of the Polyface Package continued through the third day of the meeting.

"Another reason that the Polyface Package was chosen as an example was that the polygonal structures could be displayed on our remote storage scope located in the theater connected by phone lines to the Project MAC time-shared computer. The display programs themselves, and the operation of the storage scope remotely, were also firsts for the meeting, and many of the programs and arrangements were operated for the first time only hours before they were used in the meeting. Again debugging continued (moving the equipment off-stage) during the meeting.

"Although the Polyface and Display programs were only partially debugged, those features which did work properly were demonstrated the afternoon of the first day, and vastly improved results were shown on the third day so that people in the audience could see first-hand the progress that had been made. Final debugging of these programs, carried out at a normal working pace, was completed by mid-March but the portions which were workable during the meeting did serve their role of

showing how "existing computer applications" could be controlled and driven by system-structure established using AEDJR.

"Due to the limited functions which the Polyface Package could perform at the meeting, and also due to the disrupted schedule of the Workshop, it became clear on the second day of the meeting that the actual user-oriented language for the Polyface Package would have to be greatly simplified over that which had been anticipated. Therefore, an eager sub-group of attendees proposed that a more elaborate language, preferably an already-known language, be run through the process as well, so that we would actually make two user-oriented languages at the meeting, one much more ambitious than the other. Our staff accepted the challenge, and a visit was made to the bookstore at the Tech Coop next door to the theater to obtain a copy of Iverson's A Programming Language. One of the workshop sub-groups then selected a subset of Iverson's programming language to be set up using AEDJR.

"Starting in the afternoon and working into the evening using the on-line consoles at Project MAC, this sub-group not only set up the parsing for the subset of Iverson language, but also wrote AED-0 programs to constitute an interpretive system to result in a line-by-line compile-and-go programming system. On the third day, this system, which had not even been conceived of 24 hours before, was demonstrated to the audience and worked properly except for a coding error in the second pass interpretor in which multiplication and addition codes were interchanged so that the numerical answer given by the system was in error. A member of the audience stated that he and several others had attempted a similar system and had accomplished an equivalent result in a matter of months.

"Combined with the demonstration of the Polyface language system composing simple pictures on the display screen before it too ran into second pass bugs, we had created not one but two programming systems in the manner specified in the meeting notice. We had achieved our stated objectives technically, even though we had not been able to go through as many of the intermediate steps in detail as originally planned.

"Another impressive result which was demonstrated on the third day was the completely unscheduled generation of workable machine code for the Univac 1108 computer using the new AED-1 Second Pass. Only a few weeks before the meeting we had begun a new method for setting up the tables which drive the code-generation portions of the AED-0 Second Pass as modified for AED-1. With great diligence, this new scheme had been set up to generate FAP assembly code for the 7094, and this fresh result was reported as one of the progress report segments of the first day of the meeting. In the afternoon of the second day, Bob Coe and the programmers from United Aircraft Corporation who are carrying out the bootstrapping of AED-1 onto the Univac 1108 asked whether this new scheme could not be applied to the generation of 1108 machine code as well. Once again, the challenge was accepted, and in the evening work commenced using the time-sharing system. Within four hours workable 1108 machine code was being generated, and within seven hours good 1108 machine code was being generated for algebraic, boolean, and conditional statements of AED-0, (i.e., omitting procedure definitions and complex FOR loops, etc).

Samples of various AED-0 programs and their 1108 equivalents were shown, and copies were made available to those interested on the last day of the meeting.

"There appeared to be universal agreement that the Second AED Technical Meeting was a resounding and most impressive success technically, in spite of the confusions and hectic pace described above. The entire staff was very gratified by remarks such as "This has been the most exciting meeting I have ever attended" which were a comforting balance to the disgruntled remarks which were received in approximately equal number during the first day of the meeting. The majority of the attendees between these two poles, showed by their participation as well as their more moderate encouragement, that the efforts of the Project members were well appreciated."

D. DOCUMENTATION BY MOVIES

In addition to printed reports and technical papers, motion pictures have been used to convey the flavor of certain aspects of the Project work. A number of these movies are available for loan or purchase, as listed in Appendix III.

The first movies, prepared by Dr. Ivan E. Sutherland as part of his thesis activity, showed the Sketchpad System in operation on the TX-2 Computer at Lincoln Laboratory. A short, four-minute, edited version of this film (Film No. 4) with sound track was made available by the Lincoln Laboratory and has been loaned on request to qualified organizations by the Library of the Electronic Systems Laboratory. A longer version of this film, showing the three-dimensional Sketchpad III, and also some curve-generation simulation, was used in many presentations. Since this longer film lacks a sound track, it was not made available outside the Project, but was used in many local presentations. A combination of these films was used in a 30-minute TV program in 1964 for Station WGBH (Educational Network), and a film of this program is available (Film No. 3).

In an attempt to convey the dynamic behavior of the processing algorithms of the Algorithmic Theory of Language, a color animinated motion picture was made (in 1963) by the Project members showing the step-by-step growth of the first-pass structure as a statement is scanned from left to right. The script for this film was prepared by a program operating in conjunction with the First-Pass Algorithm on the computer, and the thousands of frames of animination were carried out by hand

using special equipment for large-scale animation which we designed and assembled. The movie suffers from defects in timing, in that the complexities of the processing take place somewhat too rapidly to be fully comprehended, and a narrator is required, as there is no sound track. However, the film was used at the 1963 SJCC presentation (Ref. 80), and has been of assistance in conveying to selected audiences the dynamic growth of complicated structures.

Another widely distributed movie on Project work is Film No. 1, showing three-dimensional "graphics in motion" on the ESL Console (or "Kludge"). Finally, Film No. 2 shows an application of AED and the ESL Console in an on-line network design system. This system, CIRCAL-0, was developed in the Electronic Systems Laboratory under a NASA Grant.

APPENDIX I

AED APPLICATIONS

The following is a summary of applications of project-developed software and the ESL Display Console, as reported by users at M. I. T. and elsewhere. (These summaries do not include the uses by the Computer-Aided Design Project and industry participants in the AED Cooperative Program, described in the body of this report.)

Three studies have been reported in which the ESL Display Console was used in studies of protein molecule models. These studies were conducted under the supervision of Professor Cyrus Levinthal, Biology Department, M. I. T. A biomedical image processing project under the supervision of Professor M. Eden, Research Laboratory of Electronics, M. I. T., examined images generated by pattern recognition programs.

The Chemical Process Control Project supported by a National Science Foundation Grant, supervised by Professor L. A. Gould, Electrical Engineering Department, Electronic Systems Laboratory, M. I. T., reports extensive use of the AED-0 compiler in design of the data structure for input to a chemical engineering plant design system.

The DISCOURSE Project of the M. I. T. Department of City Planning, under the supervision of Professor Aaron Fleisher, reports using the AED-0 compiler to provide a city designer with a data structure and a set of manipulations on these data by which he creates designs.

Several studies are reported under the supervision of Professor M. L. Dertouzos, Department of Electrical Engineering, Electronic Systems Laboratory, M. I. T., in developing CIRCAL-0, CIRCAL-1, and CIRCAL-2 for on-line computer-aided electronic circuit design, and LOTUS for on-line computer-aided system design. AEDNET, a circuit analysis program developed by Dr. Jacob Katzenelson for simulating nonlinear components and networks makes extensive use of the AED-0 compiler language.

Computer-aided design of threshold element networks, under the supervision of Professor A. K. Susskind, Department of Electrical

Engineering, Electronic Systems Laboratory, M. I. T., makes extensive use of the AED-0 compiler.

The Econometrics Project, under the supervision of Professor Edwin Kuh, Department of Economics, M. I. T., is developing a complete system to perform econometric research by using AED-0, AEDJR, and the AED integrated packages. The system will simulate models of the economy, as well as offering a variety of estimation routines.

Project Intrex, the M. I. T. information storage and retrieval program directed by Professors C. F. J. Overhage and J. F. Reintjes, uses the AED-0 compiler for programming library storage and retrieval systems. AEDJR is being considered for implementing the user language.

Project Dynamo, under the supervision of Alexander L. Pugh, III, of the Sloan School of Management, M. I. T., is using the AED-0 compiler for continuous system simulation, specifically for industrial dynamic simulations.

Six studies are reported under the sponsorship of Project MAC. A project in bootstrap compiling under the direction of Professor J. M. Wozencraft used AED-0 as the initial coding language because of its capability in dealing with recursion. The MULTICS Project, under the supervision of F. J. Corbato, has recoded parts of the CTSS supervisor in AED-0. Professors M. M. Jones and M. Greenberger report that the OPS System, a subsystem of CTSS, employs a complex dynamic storage allocation mechanism which uses the AED Free Storage system extensively. A project studying instabilities under the supervision of Professor R. J. Briggs, reports using the ESL Display Console to study linear stability theory. Professor C. L. Miller, Head of the Department of Civil Engineering, reports using the ESL Display Console in developing a graphical system for the definition of two-dimensional geometric inputs to the civil engineering analysis programs, COGO and STRUDL. Dr. Thomas C. Stockham, Jr., Lincoln Laboratory, used the ESL Display Console in debugging and manipulating waveforms in his research on high-speed convolution and also developed a graph-plotting utility package for general use.

Two studies under the supervision of Professor Roy Kaplow, Department of Metallurgy, M. I. T., used the ARDS and the ESL Display Consoles for the display of mathematical functions as part of the MAP system for on-line mathematical analysis. Also used was the ESL Console for the display of three-dimensional atomic arrangements in noncrystalline materials. Professor J. F. Elliott, Department of Metallurgy, M. I. T., reports using the AED integrated packages in studying temperature distributions in the arc-furnace electrode and in the hearth of the blast furnace.

Project DISHPAN, under the supervision of Professor E. N. Lorenz, Department of Meterology, M. I. T., reports investigating the ARDS display as a means for graphic display of contour maps and world maps.

Professor A. Bers, Department of Electrical Engineering, M. I. T., is directing the study of instabilities in plasmas. The ESL Display Console has been used in stability analysis of dispersion equations relevant to a theoretical analysis of beam plasma discharge devices, and plasma instabilities in solids, through the mapping of roots of the dispersion equations in the complex frequency and wave number planes. An automated system for plotting natural frequencies of instabilities versus physical parameters was achieved.

A study of two-dimensional stress analysis under the supervision of Professor C. A. Berg, Department of Mechanical Engineering, M. I. T., reports using the AED-0 compiler for the complex-variable formulation of plane-elasticity problems using conformal mapping. Also in the Department of Mechanical Engineering, a study of a linear flow resistance element under the supervision of Professor S. Y. Lee, is using the AED-0 compiler and the ESL Display Console in a numerical implementation of a conformal mapping technique to obtain solutions to a potential flow problem with free streamlines.

Professor William Henke, Department of Electrical Engineering, M. I. T., reports extensive use of the AED-0 compiler, and the ESL Display Console in the modeling of speech production and in the production of educational movies.

Professor A. G. Oettinger, Division of Engineering and Applied Physics, Harvard University, reports using the ESL Display Console for text and curves in a system for computer aids to mathematical analysis. Mr. T. H. Van Vleck, under the supervision of Professor I. D. Pool, used AED-0 in developing a social science data bank.

APPENDIX II

SUMMARY OF PROJECT REPORTS AND PUBLICATIONS

During the period December, 1959 to May 1967, documentation of project activity was performed in a number of ways. Section A briefly describes each type of project documentation and its purpose.

Section B (references 1 through 39) contains bibliographic references and abstracts of all formal documentation issued by the project, plus a few documents which were considered to be of an informal nature when issued but in retrospect are important for the historical record. All formal documentation was automatically sent to the distribution list approved by the sponsor (about 200 addresses on the average), and numerous additional copies were sent out in response to specific requests.

Thesis activity, both graduate and undergraduate, has played a significant role in project technical accomplishments, as well as contributing to the spread of ideas and techniques as many of these students have moved on to responsible positions in government and industry. Section C (references 40 through 73), contains titles and abstracts for 35 theses supported by or otherwise connected with the project. (In cases where a thesis was also issued as a report, reference is made to the abstract in Section B.)

Technical paper presentations and publications listed in Section C (references 74 through 114), form another important means of communicating project results. In addition, a substantial number of talks and lectures were given on Computer-Aided Design which did not result in specific publications, and these are listed in Section E (references 115 through 192).

Finally, Section F lists a group of material of various forms which has become known as the "AED Documentation", i.e., specific details and instructions on the design and use of the programs and systems developed by the project. Selected AED Documents have been distributed to a specially-maintained mailing list of directly

interested groups and individuals, but have not been sent to the formal mailing list of the project. Since this material is highly volatile and subject to frequent change, anyone using this list for reference would be interested in the latest version. Thus, the listing in Section E is as of the date of publication of this report, not the contract end date of May 30, 1967.

A. DESCRIPTION OF DOCUMENTATION TYPES

1. Project Interim Engineering Reports

This group of 11 documents in the IR-series reports across-the-board progress of the project at six-month intervals (except in two cases where two six-month intervals were combined in a single document). All work is either reported in full for each period, or summarized in cases where other separate full documentation was available (see 2 below). These reports all carry Air Force report numbers in addition to the MIT-assigned numbers.

2. Project Technical Reports and Technical Memoranda

In addition to the reporting in the IR-series, 17 documents on specific topics were issued in the report (R) and technical memorandum (TM) series. These represent phases of the work which merit separate and complete reporting for easy reference. They are considered part of the formal documentation and were distributed to the same address list as used for the IR-series.

3. Informal Documentation

A large number of informal memoranda (about 190) have been issued over the life of the project, primarily for internal use. These M-series documents were not given general distribution, although many have gone to outside organizations in answer to specific requests, or where they formed part of the "AED Documentation".

Because of the close working relationship between the Computer-Aided Design Project and Project MAC at MIT, many of these memoranda were also jointly issued in the Project MAC documentation system and thus carry two sets of identifying numbers -- one for this project and one for Project MAC.

B. ABSTRACTS OF PROJECT TECHNICAL DOCUMENTATION

1. Final Report 6873-FR-3, Automatic Programming of Numerically Controlled Machine Tools, John E. Ward, January 15, 1960, 120 pp.

This report covers the three-year period of development work on the APT System which preceded the period covered by the present report. It documents the first discussions of "computer aids to design", carried out as one task during the final year of the APT work. It also contains reprints of Final Reports 6873-FR-1 and 6873-FR-2 on the development of numerical control, thus serving as a reference for all N/C MIT work for the Air Force under Contract AF-33(038)-24007 from February, 1951 through November 30, 1959.

2. Interim Engineering Report 8436-IR-1, "Investigations in Computer-Aided Design", for the period 1 December 1959 to 30 May 1960; Project Staff; published January 1961; 156 pp; DDC No. AD252062.

This Interim Report covers the first six months of the Project under Contract AF-33(600)-40604. Topics covered include a generalized technique for utilizing computer storage; a general description of the new calculating program for the APT System for numerical control programming; techniques for the evaluation of simultaneous logical functions; a summary of a computer routine for constructing a three-dimensional description of an object from orthographic projections; descriptions of the light pen and light cannon (photosensitive devices for transmitting human-modulated signals from the output scope back into the computer) and associated picture-language forms; conclusions on the study of automatic feedrate regulation in metal cutting; studies of stress calculations and other mathematical techniques for design; standard part selection by computer; and mechanical devices for graphical input to a computer.

3. Interim Engineering Progress Report 8436-IR-2, "Investigations in Computer-Aided Design", for the period 1 June 1960 to 28 February 1961; Douglas T. Ross and Steven A. Coons; published November 1961; 81 pp; DDC No. AD269573.

This Interim Report covers the seventh through the fourteenth months of the Project under Contract AF-33(600)-40604. Topics covered include a description of initial work on a bootstrap compiler for programming with "plex" structures,

including the handling of free storage and symbol tables; modifications to the new calculating program (ARELEM) for the APT System; initial experiments with graphic languages; some proposals for improved computer design for manipulating plex structures; summaries of theses on graphical languages, stress analysis and selection of standard parts; and mathematical techniques useful in design.

4. Interim Technical Progress Report Nos. 3 and 4, ASD-TR-7-820 (IR 3 and 4), "Investigations in Computer-Aided Design for Numerically Controlled Production", for the period 1 March 1961 to 8 February 1962 (MIT Report ESL-IR-138); Douglas T. Ross and Steven A. Coons; published May 1962; 86 pp; DDC No. AD282679.

This combined Interim Report covers the fifteenth through the twenty-sixth months of the Project, now under Contract AF-33(600)-42859. Topics covered include a description of current status on the basic bootstrap compiler, the available programs of the bootstrap plateau system, and the multipass compiler; discussion of a new first-pass algorithm which is believed to have wide applicability to all forms of problem statement; descriptions of three manual-intervention console designs -- a rudimentary version now operating on the MIT IBM 709 Computer, a proposed version for the 709, and a study of a remote console for a large-scale central computer; computer studies in three-dimensional shape description and stress analysis; and plans for pilot studies in pin-jointed trusses and sculptured parts.

5. Interim Technical Progress Report No. 5, ASD-TR-7-820 (IR 5), "Investigations in Computer-Aided Design for Numerically Controlled Production", for the period 1 March 1962 to 31 August 1962 (MIT Report ESL-IR-164); Douglas T. Ross and Steven A. Coons; published February 1963; 51 pp; DDC No. AD403685.

This Interim Report covers the twenty-seventh through thirty-second months of the Project under Contract AF-33 (600)-42859. Topics covered include: a restatement of project goals, including a description of the proposed system in use; a summary of theoretical developments, including a new Algorithmic Theory of Language, and a Theory of Symbolic Computation; a summary of program developments, including features of the Multi-Pass Compiler, two- and three-dimensional graphic languages, and programs for stress analysis and pilot studies; and a description of new equipment for display of three-dimensional figures.

6. Interim Technical Progress Report No. 6, ASD-TR-7-820 (IR 6), "Investigations in Computer-Aided Design for Numerically Controlled Production", for the period 1 September 1962 to 31 May 1963 (MIT Report ESL-IR-180); Douglas T. Ross and Steven A. Coons; published August 1963; 57 pp; DDC No. AD418183.

   This Interim Report covers the thirty-third through forty-second month of the Project under Contract AF-33 (600)-42859. Topics covered include: the features of the AED-0 (Automated Engineering Design) Compiler, two- and three-dimensional graphical languages and other programming studies; and a new display console for attaching to the IBM 7094 Computer.

7. Interim Technical Progress Report No. 7, ASD-TR-7-820 (IR 7), "Investigations in Computer-Aided Design for Numerically Controlled Production", for the period 1 June to 30 November 1963 (MIT Report ESL-IR-202); Douglas T. Ross and Steven A. Coons; published June 1964; 41 pp; DDC No. 442880.

   This Interim Report covers the forty-third through forty-eighth month of the Project, now under Contract AF-33(600)-10954. Topics are covered in the theoretical and hardware areas, including an outline of a machine-independent scheme for generation of efficient computer programs, analog curve generation for visual displays, and progress in several areas of stress analysis and other applications, type theory, and the AED-0 Compiler and ESL Console in Time-Sharing.

8. Interim Technical Progress Report IR 8-236-I, "Investigations in Computer-Aided Design for Numerically Controlled Production", for the period 1 December 1963 to 30 May 1964 (MIT Report ESL-IR-221); Douglas T. Ross, Steven A. Coons, and John E. Ward; published December 1964; 93 pp; DDC No. 604678.

   This Interim Report, although the second one under Contract AF-33(657)-10954, is the first one under a new MMP Project Nr. 8-236, and starts a new AF number series for these Interim Reports. For the Computer-Aided Design Project, it is the eighth in a series and covers the forty-ninth through fifty-fourth month of the Project.

Technical topics include innovations in compiling and language processing for the AED-1 System, additions to AED-0 Compiler capabilities, processing both graphical and verbal language by a single algorithm, generalized parametric surfaces, three-dimensional display in time-sharing, stress analysis and other design topics. To assist in the dissemination of research results the Project is establishing various cooperative contacts with other groups. Work with Project MAC and Ship Design at MIT, and the AED-1 Project, in which programmers from industry join in the research effort at MIT, are outlined.

9. Interim Engineering Progress Report IR 8-236-II, "Investigations in Computer-Aided Design for Numerically Controlled Production", for the period 1 June to 30 November 1964 (MIT Report ESL-IR-241); Douglas T. Ross, Steven A. Coons, and John E. Ward; published June 1965; 74 pp; DDC No. 467764.

This Interim Report under Contract AF-33(657)-10954, although the second one under MMP Project Nr. 8-236, is the ninth in a series and covers the fifty-fifth through the sixtieth month of the Project. Technical topics include innovations in compiling and language processing for the AED-1 System, additions to AED-0 Compiler capabilities, processing both graphical and verbal language by a single algorithm, generalized parametric surfaces, three-dimensional display in time-sharing, stress analysis and other design topics. This report begins with a five-year summary of the organization and progress of the Project. Later chapters describe modifications to the AED-0 Compiler System, plans for its distribution to industry, and its use in preparations for AED-1. Graphic language in time-sharing and for shape description is summarized, along with various mathematical techniques for mechanical design problems. Experiments in improved display console techniques are also described.

To assist in the dissemination of research results the Project is establishing various cooperative contacts with other groups. Work with Project MAC and Ship Design at MIT, and the AED-1 Project, in which programmers from industry join in the research effort at MIT, are outlined.

10. Interim Engineering Progress Report IR 8-236-III, "Investigations in Computer-Aided Design for Numerically Controlled Production", for the period 1 December 1964 to 31 May 1965 (MIT Report ESL-IR-262); Douglas T. Ross, Steven A. Coons,

and John E. Ward; published March 1966; 66 pp; DDC No. 482837.

This Interim Report under Contract AF-33(657)-10954, although the third one under MMP Project Nr. 8-236, is the tenth in a series and covers the sixty-first through the sixty-sixth month of the Project.

This report announces the distribution of the AED-0 System to companies participating in the AED (Automated Engineering Design) Project, and the progress in developing the AED-1 System. Technical topics include innovations in compiling and language processing for the AED-1 System, additions to AED-0 Compiler capabilities, processing both graphical and verbal language by a single algorithm, generalized parametric surfaces, three-dimensional display in time-sharing, stress analysis, and other design topics. Chapter 3 describes the application of computer-aided design to practical mechanical design problems, such as surface description and display, as in the design of ship hulls and aircraft fuselages. Chapter 4 reports the study of display requirements for future time-shared computer systems.

11. Combined Interim Engineering Progress Report IR 8-236-IV and V, "Investigations in Computer-Aided Design for Numerically Controlled Production", for the period 1 June 1965 to 31 May 1966 (MIT Report ESL-IR-278); Douglas T. Ross, Steven A. Coons, and John E. Ward; published August 1966; 125 pp; DDC No. 802213.

This Combined Interim Report, under Contract AF-33(657)-10954, although the fourth and fifth under MMP Project Nr. 8-236, is the eleventh and twelfth in a series and covers the sixty-seventh through the seventy-eighth months of the Project. Technical topics include: (1) innovations in compiling and language processing for the AED (Automated Engineering Design) System to permit processing both graphical and verbal language by a single algorithm; (2) the successful beginning of the 1966 AED Cooperative Program with industry; (3) progress in developing the AED-1, CADET, AEDJR, and AEDNET Systems; (4) the application of computer-aided design to mechnical design problems, such as surface description, display of surfaces having discontinuous slopes, and stress analysis; and (5) progress in display hardware for three-dimensional display in time-sharing, including a DDA (Digital Differential Analyzer) rotation matrix, analog curve generation, and installation of a display buffer computer.

12. Interim Engineering Progress Report IR 8-236-VI, "Investigations in Computer-Aided Design for Numerically Controlled Production", for the period 1 June to 30 November 1966 (MIT Report ESL-IR-320); Douglas T. Ross and John E. Ward; published August 1967; 67 pp; DDC No. AD821385.

This Interim Report under Contract AF-33(657)-10954, although the sixth under MMP Project 8-236, is the thirteenth in a series and covers the seventy-ninth through the eighty-fourth months of the Project.

The major emphasis continued to be focused on development of the AED (Automated Engineering Design) family of programming systems. Technical topics reported include: (1) a major revision of the RWORD Package, which builds items from the input character stream, (2) introduction of a "Features Feature", which permits selecting only those facilities relevant to a particular problem situation, (3) incorporation of a revised AEDJR into the general problem-solving scheme, which enlarges the present realm of possible AED applications, (4) major improvements in preprocessing, (5) improved integrated packages, and (6) completion of AEDNET, which simulates nonlinear circuits. In the display area, work reported includes: (1) installation of a PDP-7 buffer computer for the ESL Display Console, (2) improved display hardware, and (3) design of low-cost remote storage-tube displays. Increased activity and interest is reported in the cooperative phases of Project work, including the AED Cooperative Program, in which programmers from industry join in the research effort at MIT. A report on the First AED Technical Meeting is included, and plans for the Second AED Technical Meeting are presented.

13. Ross, D. T., and Feldmann, C. G., Papers on the APT Language, Technical Memorandum 8436-TM-1, June 1960, 46 pp, DDC No. AD243156.

This Technical Memorandum is a reproduction of two papers written by Douglas T. Ross and Clarence G. Feldmann of the Computer Applications Group. The first paper (by D. T. Ross) describes the motivations and viewpoint which influenced the design of the English-like part programming language of the APT System for automatic programming of numerically-controlled machine tools. An example of the (then) current APT System language is given, and future developments are discussed.

The second paper (by C. G. Feldmann) essentially takes up where the first leaves off, and describes additional features of the APT Language which were being added to extend its usefulness and generality. Together these papers

provide, in a condensed form, a case study description of the evolution of a specially-designed language.

14. Smith, A. F., Method for Computer Visualization, Technical Memorandum 8436-TM-2, October 1960, 49 pp, DDC No. AD248436. (SM Thesis in Department of Electrical Engineering, MIT)

This computer program produces a three-dimensional description of an object from a three-view orthographic projection of that object. A number of restrictions have been placed on the types of objects the program will handle. The most important of these is that the object must be bounded by surfaces which are simply connected plane regions.

The three views of the given orthographic projection are viewed as "point-line structures". From the given two-dimensional structures, a three-dimensional point-line structure is built up in which all the lines and points which could possibly exist are represented. After "fixing" as many points and lines of this structure as possible, the program sets up a structure listing all the plane surface regions which could possibly exist. Additional operations enable the program to fix or eliminate some of these planes, along with points and lines. If it is possible to fix or eliminate all the planes, a solution is reached and the program determines which side of each of the fixed planes is "solid", and is finished. If this is not possible, an attempt is made to present several alternative solutions to the problem.

Detailed flow diagrams of most of the program are given. The program was not coded for computer testing but hand calculation indicated that it should operate successfully in most cases.

15. Meyer, C. S., A Digital Computer Representation of the Linear Constant-Parameter Electric Network, Technical Memorandum 8436-TM-3, August 1960, 97 pp, DDC No. AD24837. (SM Thesis in Department of Electrical Engineering, MIT)

A digital computer routine resulting in a set of equations that can be solved for the branch currents or branch voltages of a linear, constant-parameter electric network is described. Topological relations are defined; and a matrix equilibrium equation, based on these relations, forms the desired representation. Advantages are obtained from the unique branch numbering system which affords the circuit analyst the opportunity of specifying tree or link branches. Other existing computer routines are studied, and a comparison is made with the method of this research. Detailed flow charts are presented, and a sample circuit is analyzed.

This work, done as a thesis, is reported as part of a study program in computer-aided design. Although the major emphasis of the study is on design of mechanical parts to be made by numerically-controlled manufacturing, the study of computer methods for design of electrical networks is also germane.

16. Ross, D. T., Computer-Aided Design: A Statement of Objectives, Technical Memorandum 8436-TM-4, September 1960, 22 pp, DDC No. AD252060.

The MIT Computer-Aided Design Project is engaged in a program of research into the application of the concepts and techniques of modern data processing to the design of mechanical parts, and the further development of automatic programming (APT) systems for numerically-controlled machine tools. The Project is a cooperative venture between the Computer Applications Group of the Electronic Systems Laboratory and the Design Division of the Mechanical Engineering Department. This document states the philosophy of approach being used by the Computer Applications Group. (A companion document, 8436-TM-5, states the philosophy of the Design Division.)

From the computer applications point of view, the primary problem is not how to solve problems, but how to state them. It is proposed that outside-in problem statement, in which a problem is described first in general terms and then refined and made precise by further elaborative statements, is required, rather than the inside-out problem statement form which characterizes present computer programming. General problems are viewed as internally structured by means of interconnected "objets". An objet is an abstract entity of meaning, and the computer's "understanding" of a problem is represented by the structure connecting the objets of the problem. The human's understanding is in terms of a language which is isomorphic to the structure of objets. This language for problem statement will consist of pictorial as well as alphabetic representations, and can be molded to suit particular problem areas. The various project activities required to establish a proper research environment also are outlined.

17. Coons, S. A., and Mann, R. W., Computer-Aided Design Related to the Engineering Design Process, Technical Memorandum 8436-TM-5, October 1960, 13 pp, DDC No. AD252061.

The MIT Computer-Aided Design Project is engaged in a program of research into the application of the concepts and techniques of modern data processing to the design of mechanical parts, and the further development of automatic

programming (APT) systems for numerically-controlled machine tools. The Project is a cooperative venture between the Computer Applications Group of the Electronic Systems Laboratory and the Design Division of the Mechanical Engineering Department. This document states the philosophy of approach of the Design Division. (A companion document 8436-TM-4 states the philosophy of the Computer Applications Group.)

The engineering design process is viewed as a stochastic iterative process in which a recognized human need leads to a preliminary tentative concept of a means for its achievement; subsequent analysis, evolution, and judgement leads to modification of the concept, and even possibly to a modification of the original goal, until certain standards are met and the need is satisfied.

The manpower-time requirements for proceeding from the original design concept to its realization in a manufactured part, device, or system is investigated, with a view to determining where best to begin consideration of computer aids in the sequence. In general, the conclusion is drawn that for the present the computer can be most effective in replacing manpower in routine drafting, minor design decisions, engineering computation occurring in analyses (particularly stress computations), and as an aid in the selection of standard parts.

18. Welch, J. D., Automatic Feedrate Regulation in Numerically Controlled Contour Milling, Report 8436-R-1, December 1960, 64 pp, DDC No. AD253676.

The application of continuous-path numerical control and tracer control to machine tools has increased the need for a more automatic and accurate method of feedrate selection. A method of using a feedback signal from the cutting process to determine the desired feedrate is presented as a solution to this problem.

The choice of the proper feedback signal is first considered. It is shown that whereas a computer regulation system is best suited to controlling the metal removal rate, a feedback regulation system is best suited to controlling the tool wear rate. The tool temperature is the output of the cutting process which is most closely related to the tool wear rate. The method chosen for detecting the tool temperature is the tool-work thermocouple technique.

An experimental system for regulating the feedrate of the MIT numerically-controlled milling machine is described. The results show that subject to certain restrictions the tool-work thermal emf can be used successfully as a feedback signal to control the machine feedrate so as to keep the cutting temperature essentially invariant to changes in the cross section of the cut.

19. Randa, G. C., Design of a Remote Display Console, MIT Report ESL-R-132, February 1962, 126 pp, DDC No. AD274985. (EE Thesis in Department of Electrical Engineering, MIT)

This report is a design study of a console which may be remotely connected to a high-speed digital computer. The console is designed to present visual information generated during the course of problem solution on a cathode-ray-tube display. The console is provided with an integral drum memory for display maintenance, and allows the printing of points, characters and vectors. Light pens also are provided for graphic communication and program control. Primary control of the console is exercised by the central data processor, in the selection of modes of operation; however, local control over the chaining of blocks of data to be displayed is provided by the use of control words stored in the display memory.

These features provide flexible operation of the remote console. Since several consoles of this type may be connected to a computer, low cost is a prime consideration in the design.

This report discusses equipment specifications derived from a consideration of several problems of varying complexity, and concludes that use of the central data processor itself to perform computations associated with the display is the most economical system. The general system organization of the console equipment and its interconnection with the central processor are outlined, and the detailed logical design of the console and associated storage unit are considered.

20. Randa, G. C., and Grondstra, J. W., Design for a Manual Intervention Console, Memorandum No. 8753-M-55, March 1962.

The work reported here covers the design study of a manual-intervention console to be used with an IBM 709 Data Processing System. The console will provide operator intervention, display, and control functions which will permit him to function efficiently while utilizing the computer for computer-aided design studies, or while monitoring the progress of a difficult, complex problem. Facilities are available on the console for use of two cathode ray oscilloscopes for display of points, characters, or vectors (using incremental-digital rather than analog display generation techniques); two electric typewriters for operator communication with the machine; a high-speed photoelectric tape reader for loading of large amounts of data; light pens and light cannon for use with the oscilloscopes; indicator lights to indicate the status of the problem; an alarm clock as an external time-keeping device; activate pushbuttons for inserting one-shot data; and mode toggle switches for use in indicating static inputs to the machine.

Work which has been completed and which is reported here includes the systems design for the console, using characteristics of assumed console equipments, and the preliminary logic design required to interconnect the equipments and the computer input-output channel. The equipments used on the console and their functions are described, as well as the methods used to control them in conjunction with the signals available from the 709 direct data connection. Programming also is considered as it affects the design of the console and its operation with the computer. Detailed logic diagrams also are given to describe the circuits necessary to make the console operational. The various modes of operation are described and the instrumentation is defined.

21. Ross, D. T., An Algorithmic Theory of Language, MIT Report ESL-TM-156, November 1962, 68 pp, DDC No. AD296998.

The Algorithmic Theory of Language takes the view that processing algorithms define classes of language. A language belongs to a class depending upon whether or not it is properly processed by the corresponding algorithm. Following n-component element and plex definitions, several General Principles concerning the step-by-step growth of large, complex structures are introduced. The words and symbols of language are then considered to be elements with attractive and repulsive properties which cause them to link together to form linguistic structures. The General Principles are applied to suitable element definitions to yield derivations of successively more elaborate algorithms defining the behavior of these elements, and generating in one left-to-right pass the First-Pass Structure, which explicitly exhibits the syntactic and semantic structure of a statement by showing syntactic context by a tree structure and semantic context by the "precedence string".

The present development stops with the concepts of major and minor modifiers and leaves ambiguity resolution and other topics to future papers. (This document is a preprint of a paper submitted to the Journal of the Association for Computing Machinery for publication in 1963.)

22. Sutherland, I. E., Sketchpad: A Man-Machine Graphical Communication System, M. I. T. Lincoln Laboratory Technical Report No. 296, 30 January 1963, 91 pp, DDC No. AD404549. (This report, based on a Doctoral thesis, was not directly supported by the CAD Project but was distributed by it because of the close technical coupling in matters of goals and thesis supervision.)

The Sketchpad system uses drawing as a novel means of communicating with a computer. The system contains input, output, and computation programs that enable it to interpret information drawn directly on a computer display. It has been used to draw electrical, mechanical, scientific, mathematical and animated drawings; it is a general-purpose system. Sketchpad has shown the most usefulness as an aid to the understanding of processes, such as the motion of linkages, which can be described with pictures. Sketchpad also makes it easy to draw highly repetitive or highly accurate drawings and to change drawings previously drawn with it. The many drawings in this report, including legends and labels, were all made with Sketchpad.

A Sketchpad user sketches directly on a computer display with a "light pen." The light pen is used both to position parts of the drawing on the display and to point to them to change them. A set of push buttons controls the changes to be made, such as "erase" or "move." Except for legends, no written language is used.

Information sketched can include straight line segments and circle arcs. Arbitrary symbols may be defined from any collection of line segments, circle arcs, and previously defined symbols. A user may define and use as many symbols as he wishes. Any change in the definition of a symbol is at once seen wherever that symbol appears.

Sketchpad stores explicit information about the topology of a drawing. If the user moves one vertex of a polygon, both adjacent sides will be moved. If the user moves a symbol, all lines attached to that symbol will automatically move to stay attached to it. The topological connections of the drawing are automatically indicated by the user as he sketches. Since Sketchpad is able to accept topological information from a human being in a picture language perfectly natural to the human, it can be used as an input program for computation programs which require topological data, e.g., circuit simulators.

Sketchpad itself is able to move parts of the drawing around to meet new conditions which the user may apply to them. The user indicates conditions with the light pen and push buttons. For example, to make two lines parallel, he successively points to the lines with the light pen and presses a button. The conditions themselves are displayed on the drawing so that they may be erased or changed with the light pen language. Any combination of conditions can be defined as a composite condition and applied in one step.

It is easy to add entirely new types of conditions to Sketchpad's vocabulary. Since the conditions can involve anything computable, Sketchpad can be used for a very wide range of problems. For example, Sketchpad has been used

to find the distribution of forces in the members of truss bridges drawn with it.

Sketchpad drawings are stored in the computer in a specially designed "ring" structure. The ring structure features rapid processing of topological information with no searching at all. The basic operations used in Sketchpad for manipulating the ring structure are described.

23. Stotz, R. H., Specialized Computer Equipment for Generation and Display of Three-Dimensional Curvilinear Figures, M. I. T. Technical Memorandum ESL-TM-167, March 1963, 154 pp, DDC No. AD406608. (Also SM Thesis in Department of Electrical Engineering, M. I. T.; also published in condensed form in the Proceedings of the 1963 Spring Joint Computer Conference.)

Studies being conducted of Computer-Aided Design of three-dimensional shaped objects have shown the need for improved graphical man-computer communications, particularly faster displays. A straight-line-and-curve-drawing display system is proposed which is capable of drawing two-dimensional, axonometric projections of curvilinear three-dimensional figures at up to 100 times the speed of point-plotting display scopes. The system, based on digital incremental computing techniques, consists of a Line Generator to produce time-varying x, y, and z pulse-train signals proportional to the numerical input information; a Rotation Matrix to transform these signals into the h and v coordinate axes of the scope; and Accumulating Registers (bi-directional counters) to hold the resultant data for the scope deflection amplifiers. The Line Generator is capable of producing straight lines and second-order curves of variable length.

Two basic elements are compared as building blocks for the Line Generator and Rotation Matrix: the Binary Rate Multiplier (BRM) and the Digital Differential Analyzer (DDA). The operating principles of these units are described and their differences as computing elements for this system are analyzed. The entire system was simulated on a PDP-1 computer which has a standard display scope and the results of comparative tests between DDA- and BRM-drawn figures are shown. Although the BRM has larger errors than the DDA for equivalent register lengths, its simplicity makes it attractive. BRM errors are studied in detail, and theoretical and simulation results for improved BRM's are given.

Additions to the display system permitting generation of stereoscopic and perspective projections also are described, and figures resulting from simulations of these

systems are shown. It is concluded that a display system with an incremental computing capability will provide a sound basis for future work in Computer-Aided Design.

24. Coons, S. A., An Outline of the Requirements for a Computer-Aided Design System, M.I.T. Technical Memorandum ESL-TM-169, March 1963, 13 pp, DDC No. 404832. (Also published in the Proceedings of the 1963 Spring Joint Computer Conference in Detroit.)

Computer-Aided Design, as distinct from Automatic Design, involves the creation of a man-machine system in which the designer and computer can work smoothly, as a team, on original design problems requiring creative solutions. This paper reviews the iterative and unpredictable nature of the design process, establishes broad requirements of evolutionary flexibility which a Computer-Aided Design System must meet, and describes several applications indicating that such a broad general system is practicable.

25. Ross, D. T., and Rodriguez, J. E., Theoretical Foundations for the Computer-Aided Design System, M.I.T. Technical Memorandum ESL-TM-170, March 1963, 43 pp, DDC No. 404832. (Also published in the Proceedings of the 1963 Spring Joint Computer Conference in Detroit.)

For an evolutionary Computer-Aided Design System to become a reality, it must be built on a cohesive, rigorous foundation. This paper describes the application of The Algorithmic Theory of Language and its companion Theory of Operators to the problem. It is shown how algorithms of the Language Theory transform statements made in graphical or verbal language into the corresponding First-Pass Structure, which explicitly exhibits the syntactic and semantic structure of the statement. Operators then transform the meaning into the Modeling Plex, which represents the problem to which the statement referred. Successive operators then generate further models of statements so that a common approach to all aspects of the Computer-Aided Design System is achieved. The concepts are illustrated by an example drawn from servomechanism design.

26. Johnson, T. E., Sketchpad III, Three-Dimensional Graphical Communication with a Digital Computer, M.I.T. Technical Memorandum ESL-TM-173, May 1963, 51 pp, DDC No. 406855.

(Also SM Thesis in Department of Mechanical Engineering, M. I. T.; also published in condensed form in the Proceedings of the 1963 Spring Joint Computer Conference.)

In Computer-Aided Design of mechanical parts, effective graphical man-machine communication is required. In particular, it is desired that the human designer at the computer console be able to quickly construct and manipulate three-dimensional figures in as natural a manner as possible, at the same time making use of the logical and computational power of the computer to assist him. At the present time, the most appropriate input-output device for this purpose appears to be an output oscilloscope screen, coupled with a light-pen input to the computer and various other manual input devices such as control knobs, switches and keyboards.

This paper describes a programming system written for the M. I. T. Lincoln Laboratory TX-2 Computer which permits scope, light-pen and control knobs to be used in a flexible manner to draw three-dimensional, straight line, "wire frame" figures. Three orthogonal views complement a perspective view to permit simultaneous observation from several vantage points to increase depth perception. Drawing can be accomplished in any view.

27. Ross, D. T., and Feldmann, C. G., Verbal and Graphical Language for the AED System: A Progress Report, M. I. T. Project MAC Report MAC-TR-4, May 6, 1964, 26 pp, DDC No. AD604678. (Also included in ESL Interim Technical Progress Report No. 8, IR-8-236-I, Ref. 8.) This paper was presented at the M. I. T. Industrial Liaison Symposium on Project MAC, May 6, 1964.

For Computer-Aided Design use of time-sharing a single language which can take either verbal or graphical form is required. This paper describes how a single language processing technique, which is in turn a special application of more general concepts concerning the step-by-step growth and processing of large structures of interrelated elements, can efficiently process both language forms in the same manner. Illustrations of the concepts involved are also drawn from the methods used in the AED-0 Compiler, an efficient ALGOL-60-based compiler used in Computer-Aided Design work, which is available as a public command in the Project MAC CTSS.

28. Ross, D. T., AEDJR: An Experimental Language Processor, M. I. T. Technical Memorandum ESL-TM-211, published September 1964, 53 pp, DDC No. 453881.

The AEDJR System is a miniaturized version of the kind of evolutionary problem-solving system which is the goal of the M. I. T. Computer-Aided Design Project. The core of the system is a simplified but more general version of the First-Pass Algorithm which automatically transforms an input string of statements in a language into a structure which shows both the syntactic and semantic structure of the statements. The behavior of this algorithm is controlled by meta-properties associated with words of the language. AEDJR allows new words to be defined and their meta-properties to be assigned in a convenient manner, so that essentially arbitrary artificial languages can be processed by the system. It also has provision for testing new vocabularies thus established and for using AEDJR processing to drive arbitrary working programs. Thus problem-oriented systems with powerful control languages can be easily constructed. Although the present version of AEDJR is preliminary and experimental, it demonstrates tha techniques being used for development of a full-scale, generalized Computer-Aided Design System.

29. Ross, D. T., Implications of Computer-Aided Design for Numerically Controlled Production, M. I. T. Report ESL-TM-212, September 1964, 31 pp, DDC No. AD 453880. (Reprint of a paper presented at the First Annual Meeting and Technical Conference of the Numerical Control Society in New York on March 20, 1964.)

Although the potential impact of Computer-Aided Design techniques on the application of numerical control in a production environment is very great, changes in present practice will be evolutionary rather than revolutionary. There are three main portions of the Computer-Aided Design concept: 1) combined verbal and graphical language, for making statements about problems; 2) the modeling plex, for storing in a consistent fashion all of the necessary information about a problem; and 3) generalized operators, for manipulating statements and models to perform useful mathematical and data processing functions. Man-machine console facilities, including graphical and verbal input-output facilities, are needed and must be made economically feasible on a large scale. The combined hardware-software system can be applied to various new problem areas by bootstrapping new techniques and processes into the system using available

facilities. It appears that the steps now associated with the application of numerical control proper will play a crucial but subordinate role in the overall design-to-manufacturing process, and will take on new forms and procedures. These longer range implications are a natural evolution from the integration of "master dimensions" and similar techniques now being pursued in industry. Portions of this overall new approach can be made available to the small business, and the techniques can be specialized for particular purposes, such as drafting, for small and large business alike.

30. Hamilton, M. L., and Weiss, A. D., An Approach to Computer-Aided Preliminary Ship Design, M. I. T. Technical Memorandum ESL-TM-228, January 1965, 65 pp, DDC No. AD 461412.

This report describes an application of Computer-Aided Design concepts to the general preliminary design of ships, in which shape description plays an important part. Although the present study is preliminary in nature and will require considerable elaboration for practical use, it does indicate the feasibility of the approach. The design and evaluation of hull forms was accomplished "on line" using the Project MAC time-shared digital computer, the display console developed by the M. I. T. Electronic Systems Laboratory, and a very general, parametric surface description technique developed by Prof. S. A. Coons. Three-dimensional hull surfaces displayed on the CRT screen could be altered in a few seconds by typed-in changes in parameters, and could be rotated to any desired viewing angle for study. Using these techniques, the lines of the US DD 692 were simulated such that the routines for calculating midships coefficient, prismatic coefficient, displacement, wetted surface area, and centers of buoyancy yielded values closely resembling those of the actual ship. A brief economic analysis shows the great saving in time and cost and the corresponding increase in study of alternative designs which would be possible with such a design system.

31. Stotz, R. H., and Ward, J. E., Operating Manual for the ESL Display Console, M. I. T. ESL Memorandum 9442-M-129 (also Project MAC Memorandum MAC-M-217), March 9, 1965, 47 pp.

This document, which describes the hardware functions and detailed command structure of the ESL Display console, supersedes an earlier memorandum written in August 1963 during the construction of the console. The ESL Console is designed to operate from the direct data channel of an IBM 7094 Computer, and is a specialized

computer which automatically converts three-dimensional drawing commands from the 7094 memory into arbitrary two-dimensional projections. Real-time rotation, translation, and scale changes are possible even in time-sharing (the console operates under the Project MAC 7094 Compatible Time-Sharing System). Light-pen tracking for graphic input is an automatic hardware function. Other features include character and vector generators, and convenient manual input devices for controlling the real-time display manipulations.

32. Lang, C. A., New B-Core System for Programming the ESL Display Console, M. I. T. ESL Memorandum 9442-M-122 (also Project MAC Memorandum MAC-M-217), April 30, 1965, 84 pp.

Graphical man-machine communication is made possible by devices such as the ESL Display Console, but as with all elaborate input-output devices, the programming required to obtain desired performance is an intricate and exacting task. Programming problems are greatly compounded when real-time response is required in a time-sharing environment. To resolve these difficulties and provide to the user a convenient software interface, the Computer-Aided Design Project has prepared two integrated packages of system programs known as the A-Core/B-Core System. The A-Core System is incorporated into the Project MAC Time-Sharing System Supervisor and provides elemental real-time control of the ESL Display Console. The B-Core System described in this memorandum in turn controls the A-core functions in response to a large number of convenient functions which constitute the building blocks of graphical man-machine communication. Not only is the user freed of the intricacies of console programming, but also future changes in system hardware and software will not affect his programs.

Flexible means are provided by the B-Core System for creating and editing a display file. Procedures for adding, removing, and replacing console commands, as well as a "copy" function whereby identical sets of commands may be reproduced within the file are included in the package. An important feature is the ability to define subroutine pictures or "subpictures" (analogous to the way AED-0 procedures are defined) and each time one is called only a single command is added to the display file. Also, a set of procedures for adding "standard picture parts" to the display file is provided. These include characters, points, lines, arcs of circles, rotation matrix commands, and console mode-change commands.

The user may identify items added to the display file by assigning each one a name. All communication between the user and the system about them then takes place in

terms of these names, the system automatically performing the required transformations for communication with A-core. The names remain invariant as the positions of the commands which they represent are moved in the display file. Thus B-core provides a form of automatic storage allocation for the display file.

33. Ross, D. T., The M. I. T. Computer-Aided Design Project -- A Half-Decade Summary, M. I. T. ESL Memorandum 9442-M-140, July 1965, 35 pp. (Reprinted in ESL-IR-241, Ref. 9, also pp. 774-797 of Symposium Proceedings.)

This paper, presented at the Air Force Materials Symposium, Miami Beach, Florida, June, 1965, does not describe the Computer-Aided Design concept from the user's point of view, but instead covers the methods of attack being used in creating the system itself. The work of the Project includes theoretical studies of problem structuring, language processing, compiling, and shape description; program developments for constructing working design systems and studying applications; hardware developments for man-machine communication; and cooperatire ventures with industry to ensure practical transmission of research results into the industrial environment. Progress in these areas during the first five years of effort is described, and plans and prognoses for future developments are outlined. Annotated illustrations summarize the technical discussion.

34. Johnson, T. E., Analog Generator for Real-Time Display of Curves, M. I. T. Lincoln Laboratory Technical Report 398, July 28, 1965, 28 pp, DDC No. AD623945 (an abstract of this report is given in IR-236-IV and V, Ref. 11).

Real-time interaction between man and digital computers often requires a computer-directed graphical display. The cathode-ray tube (CRT) has so far been the only candidate for applications requiring fast write and erase times. However, in addition to an adequate display surface, there is need for more advanced generators which are capable of drawing straight lines and curves without demanding constant attention from the computer. Digital techniques for drawing curves and straight lines without performing point-by-point calculation in the computer are well known but are quite expensive and complex. Analog methods for generating curves are simpler and more direct, but have found little use because of speed and accuracy limitation and the high price of analog switching components.

Two recent advances in solid-state analog devices that have changed this situation are the high-current field-effect transistor (FET) and the low-cost, high-bandwidth, high-power operational amplifier. The advent of the high-current FET provides, for the first time, analog switching speeds in the microsecond region with no current or voltage offset.

This report discusses the design and performance of a low-cost analog generator capable of forming rotated cubics and conics for display. The generator, presently in operation on the TX-2 computer at Lincoln Laboratory, has an accuracy of 0.1 percent and is capable of producing over 3000 arbitrary curves per second. It should be regarded as an extremely fast general-purpose analog computer that has been adapted for the particular application of real-time display. The analog computer is driven by a general-purpose digital computer that supplies the parameters, selects the computing paths, and furnishes the length of time the analog generator must run to form any one solution (the display of a curve). While the analog generator is computing the locus of the curve, the driving digital computer is disconnected from the generator and is free for general computation. Thus, the computational load on the digital computer necessary to initiate and maintain a complicated display of cubics and conics is greatly reduced.

35. Ross, D. T., Lang, C. A., Polansky, R. B., Some Experiments with an Algorithmic Graphical Language, M.I.T. Technical Memorandum ESL-TM-220, August 1965, 55 pp, DDC No. 472147.

This report describes the inner workings of the "May 6th Demonstration Program", a graphical language system using the ESL Display Console attached to the Project MAC time-shared computer system. The program was written to demonstrate that picture languages, such as Sketchpad, do not require special techniques, but are processed by the same First-Pass Algorithm of the Algorithmic Theory of Language as is used for programming languages in the AED family of compilers. The system is constructed around the AEDJR System for defining and parsing languages, the Pseudo Pen Program for precise light-pen action, the A-Core/B-Core System for controlling real-time display console action in the time-sharing environment, plus a number of routines to define the meaning of the graphical words. Although refinements of these techniques are being incorporated in the newer CADET Systems (Computer-Aided Design Experimental Translator), this report gives a more complete description of how graphical language systems are constructed than has previously been available.

36. Wolman, B. L., Operators for Manipulating Language Structures, M. I. T. ESL Memorandum 9442-M-160, (also Project MAC Memorandum MAC-M-304), March 18, 1966, 28 pp. (Presented at the Symposium on Symbolic and Algebraic Manipulation, Association for Computing Machinery, Washington D. C., March 29-31, 1966.)(Based on an SM Thesis, June 1965, Ref. 55a.)

The manipulation of symbolic expressions, optimization of computer programs by a compiler, and the use of graphical or pictorial input-output have heretofore been considered to be unrelated problems. The Algorithmic Theory of Language provides a language structure capable of representing both the syntactic and semantic structure of statements in algebraic, procedural, or graphical languages. Utilizing the semantic sequencing information in the structure, "operators" defined for atomic forms may be applied to arbitrarily complex structures to provide a uniform and powerful manipulation capability for these and other areas of application. This paper describes an on-line, experimental version of a system constructed in this manner.

37. Ross, D. T., The AED Approach to Generalized Computer-Aided Design, M. I. T. Report ESL-R-305, April 1967, 56 pp, DDC No. AD814912.

Computer-aided design requires a multitude of specialized man-machine problem-solving systems, each with its own graphical-verbal jargon suited to the field of application. Many popular misconceptions about computer-aided design hinge on a lack of appreciation of the underlying processes which are involved in any successful system. An experiment introduces to nonsystem-programmers the thesis that four principal phases are involved: the lexical, parsing, modeling, and analysis phases, which receive, recognize, understand, and solve a problem. The AED (Automated Engineering Design) Approach of the M. I. T. Computer-Aided Design Project provides generalized table-driven processors for each of these phases so that powerful specialized systems can be created, modified, and maintained with a fraction of the customary effort. The finite-state machines of the RWORD System, and the powerful parsing of the AEDJR System, coupled with system-programming techniques of the AED Compiler for generating Semantic Packages provide an orderly and efficient methodology which soon will be available on a variety of computers.

38. Lapin, R. B., Translation Between Artificial Programming Languages, M.I.T. Report ESL-R-306, April 1967, 119 pp, DDC No. AD815395, (also SM thesis in Department of Electrical Engineering, M.I.T.).

Automatic translation of computer programs from one artificial language to another is often a suitable solution to the reprogramming problem. The report presents a general method for translation between artificial programming languages and its application to the translation between MAD (Michigan Algorithmic Decoder) and AED-0 (Algol Extended for Design). The method can be applied to any language, as long as the scope of the target language effectively includes that of the source language. Use of the AEDJR language processor and associated manipulation operators assures generality while maintaining simplicity. The translation is accomplished by first defining the source language in terms of the AEDJR grammar. This definition is then used to produce a parsed tree structure for the source program. Next, the application of a set of translation rules produces an equivalent tree structure for the target program. Finally, from this structure the equivalent program is generated as a character string in the target language

39. Coons, S. A., Surfaces for Computer-Aided Design of Space Forms, M.I.T. Project MAC Technical Report MAC-TR-41, June 1967, 111 pp, DDC No. AD663504.

The design of airplanes, ships, automobiles, and so-called "sculptured parts" involves the design, delineation, and mathematical description of bounding surfaces. A method is described which makes possible the description of free-form doubly curved surfaces of a very general kind. An extension of these ideas to hyper-surfaces in higher dimensional spaces is also indicated.

The surface technique has been specifically devised for use in the Computer-Aided Design Project at M.I.T., and has already been successfully implemented here and elsewhere.

C. ABSTRACTS OF THESES ASSOCIATED WITH PROJECT WORK

40. Meyer, C. S., A Digital Computer Representation of the Linear Constant Parameter Electric Network, Master of Science thesis in Electrical Engineering, June 1960.

(Also published as Technical Memorandum 8436-TM-3, for Abstract see Ref. 15.)

41. Smith, A. F., Method for Computer Visualization, Master of Science thesis in Electrical Engineering, June 1960.

(Also published as Technical Memorandum 8436-TM-2, for Abstract see Ref. 14.)

42. Welch, J. D., Automatic Feedrate Regulation in Numerically Controlled Contour Milling, Master of Science thesis in Electrical Engineering, June 1960.

(Also published as Report 8436-R-1, for Abstract see Ref. 18.)

43. Johnson, L. E., Graphical Communication with a Digital Computer, Master of Science thesis in Mechanical Engineering, June 1961.

This thesis reports the results of an investigation of the problem of conveying graphical information about mechanical parts from the designer to a digital computer. This study stems from the need for new techniques for the utilization of computers in the design process. The particular technique which is discussed in some detail would enable a designer to sketch projections of a part using an electromechanical drawing device, push buttons to indicate numerical quantities, and thus convey to the computer sufficient information for a complete shape description of the part.

A shape description interpretation program is described which can be used by the computer to concisely store the orthographic view information given by a designer as he draws at a console. A proposed input device for transmitting shape description information to the computer is also described in some detail. The interpretation program can be used with this device, however the major part of it is sufficiently general to be used with other graphical input devices.

44. Parmelee, R. P., A Study of a Stress Analysis Facility for Computer-Aided Design, Master of Science thesis in Mechanical Engineering, June 1961.

This thesis reports a study into various aspects of stress analysis in computer-aided design, with such features as internal computer storage of the three-dimensionality of the part, and the ability to communicate graphical information to the computer.

Several elementary stress analysis techniques are considered (beam stresses, torsion stresses, Mohr's circle combination of stresses, and plate stresses) and these techniques are outlined briefly in the introductory pages. The necessary programs for obtaining cross section from the internally stored three-dimensionality information of the part and for automatic programming of the specific stress analysis problem are discussed and outlined.

In addition, the problems associated with the automatic programming and solution of two partial differential equations are considered. The Poisson or nonhomogeneous Laplace equation and the nonhomogeneous biharmonic equation, describing torsion stresses and plate stresses, respectively, are considered from the standpoint of automatic programming requiring only that the designer, while viewing an oscilloscope-produced representation of the boundaries of the region (a cross section for the case of torsion stresses) align suitably an array of mesh points (also appearing on the scope face) on the region.

These separate subfacilities are then organized and placed under the control of a central processor or compiler which interprets the designer's statement of the specific problem, assembles the appropriate subfacilities, initiates computation, and organizes the outputs for display to the designer or for transfer to some other facility in the overall computer-aided design system.

45. Purvis, J. D., Jr., An Investigation of a Standard Parts Selection Facility for Computer-Aided Design, Master of Science thesis in Mechanical Engineering, September 1961.

This thesis reports work done in an investigation of a standard parts selection facility for a computer-aided design system. The work that was done involved the analysis of the procedure followed by an engineering designer when selecting a standard threaded fastener. The information requirements of the designer were ascertained by this analysis. A method of data storage was formulated and used for the storage of the information requirements. The final step was the formulation of data retrieval routines by which a threaded fastener would be selected to meet either a specified external loading or a specified special design requirement.

46. Verderber, J. A., Graphical Input-Output Devices for Computer-Aided Design, Master of Science thesis in Mechanical Engineering, June 1961.

In the process of engineering design the use of graphical data of the symbolic, indicial, and ikonal types is necessitated by the stages of conceptualization and analysis prerequisite to the formulation of a hierarchy of design decisions. The factors affecting design decisions are weighed with an eye toward development of graphical input-output devices for designer-computer matching. Three basic manual intervention techniques for the processing of graphical data are discussed. In the first, a scheme for computer control of a designer's sketching is analyzed for the logic, error, power, control, and utilization information necessary and precedent to the formulation of rigorous design criteria for a prototype system. The second and third approaches are based on remote manipulation techniques for the electromechanical or electronic display of graphical data. Stages in the design, construction, and testing of "bread-board hardware" of wide flexibility are described in detail, and it is found feasible to employ remote manipulation as a means of activating a graphical input-output device for digital computers. A sample sketch made with the help of the test control system is included.

47. Randa, G. C., Design of a Remote Display Console, Master of Science thesis in Electrical Engineering, February 1962.

(Also published as Report ESL-R-132, for Abstract see Ref. 19.)

48. Bennett, P. T., Automatic Strength of Materials Routines for Computer-Aided Design, Master of Science thesis in Mechanical Engineering, February 1963.

This thesis describes an investigation in the field of computer-aided stress analysis. It reviews the underlying philosophy of the forms this aid should take to blend in the best way the capabilities of the designer and the computer.

The several related steps in designing a part for strength are examined and discussed. The development of a working computer system for beam stress analysis is described. The underlying theory is presented, and a delineation of the graphical input and control techniques is included.

49. Stotz, R. H., Specialized Computer Equipment for Generation and Display of Three-Dimensional Curvilinear Figures, Master of Science thesis in Electrical Engineering, February 1963.

(Also published as Technical Memorandum ESL-TM-167, for Abstract see Ref. 23. Also published in condensed form in the Proceedings of the 1963 Spring Joint Computer Conference, see Ref. 81.)

50. Sutherland, I. E., Sketchpad, A Man-Machine Communication System, Doctor of Philosophy thesis in Electrical Engineering, February 1963.

(Also published as Lincoln Laboratory Report TR-396, for Abstract see Ref. 22. Also published in condensed form in the Proceedings of the 1963 Spring Joint Computer Conference, see Ref. 82.)

51. Johnson, T. E., Sketchpad III, Three-Dimensional Graphical Communication with a Digital Computer, Master of Science thesis in Mechanical Engineering, June 1963.

(Also published as Technical Memorandum ESL-TM-173, for Abstract see Ref. 26. Also published in condensed form in Proceedings of the 1963 Spring Joint Computer Conference, see Ref. 83.)

52. Tillman, C., Jr., On an Adaptive Technique for Solving Linear Algebraic Equations, Master of Science thesis in Mechanical Engineering, June 1963.

Pattern-search techniques may be employed in solving linear algebraic equations by minimization of the extremum functions with which linear equations are associated. Digital computer programs were written for determining the efficiency of pattern-search techniques relative to other common methods by which linear equations are solved. Functional relationships between computation time T, number of unknowns n, and condition number P were obtained for equations will full coefficient matrices. It was found that at fixed $P = 100$, solutions of 5-decimal accuracy could be obtained in computation times proportional to $n^{1.2}$-$n^{2.2}$, the exact proportionality depending on the eigenvalue distributions of the coefficient matrices being considered. With constant $n = 12$, T was found to vary as $P^{0.2}$-$P^{0.5}$, with eigenvalue distribution again determining the exact exponent. It was concluded that for many types of linear problems pattern searching may offer considerable time savings over conventional solution techniques.

53. Best, A. R., The Examination of Pen Tracking Schemes for Man-Computer Systems, Bachelor of Science thesis in Electrical Engineering, June 1964.

This thesis examined easily implementable tracking patterns for visual man-machine communication systems. The method of investigation was simulation on the PDP-1 computer of various hardware logics and parameters. Essentially different programs were written for each logic, and parameters were changed in each program to determine the most effective combinations. The conclusion is that a square or a small tracking cross that "predicts" the next location of the pen is the best.

54. Hamilton, M. L., and Weiss, A. D., An Approach to Computer-Aided Preliminary Ship Design, Master of Science thesis in Mechanical Engineering and in Naval Architecture, September 1964. (Also published as Technical Memorandum ESL-TM-228, for Abstract see Ref. 30.)

55. Cohler, N., Three-Dimensional Computer Displays Using Thin-Film Analog Multipliers, Master of Science thesis in Electrical Engineering, June 1965.

This thesis describes the development and testing of a four-quadrant analog multiplier for use in a hybrid analog-digital computer display system. The multiplier utilizes the effect of magnetoresistance in thin ferromagnetic films. Its advantages for this application include low cost, versatility as a three-term multiplier-added, small size, and ease of replication with "standard" thin-film computer memory components. Comparison is made with a commercially available analog multiplier which also uses magnetoresistive elements. The limitations, accuracy, and input-output requirements of the multiplier, as well as thoughts for its future improvement, are discussed.

55a. Wolman, B. L., Operators for Manipulating Verbal and Graphical Language Structures, Master of Science thesis in Electrical Engineering, June 1965. (For Abstract see Ref. 36.)

56. Guttmann, E. G., Investigation of Threshold Logic in High-Speed Display Systems, Master of Science thesis in Electrical Engineering, June 1965.

A computer-driven graphical display system constructed by the Electronic Systems Laboratory utilizes binary rate multipliers (BRM's) to generate and rotate figures made up of straight lines to be visually displayed on a cathode ray tube. The displayed pictures are sometimes distorted and the lines are ragged looking, due to inherent accumulative round-off errors of the binary rate multipliers in the rotation matrix. In this thesis, a new approach to improve the picture quality by eliminating the distortion and smoothing out the lines was studied -- the use of digital differential analyzers (DDA's) to replace the binary rate multipliers. The particular problem was to design a four-input DDA capable of adding four ten-bit signed numbers in 1.5 microseconds.

The realization of such an adder by conventional AND/OR or NAND logic would be prohibitively complex and expensive. Multi-input threshold logic, on the other hand, is found to yield a relatively simple realization. Complete logic and system designs for a DDA rotation matrix using threshold logic are established, including interfaces with the other parts of the display system. One cell of the four-input adder was designed and tested, and found to have a transition time of 80 nanoseconds.

It is concluded that a workable system could be constructed with this cell. Suggestions are given for further work, particularly in obtaining increased speed.

57. Ku, J. P. W., The Use of Dual Quarternions in the Analysis of General Spatial Four-Bar Mechanisms, Master's thesis in Mechanical Engineering, June 1965.

This thesis concerns a computer program for the analysis of space linkages involving five basic joints: turn, slide, turn-slide, screw, and ball joints, a list which covers all possible elementary motions of a kinematic pair in three-dimensional space. The program calculates the successive positions of the moving members of a four-bar space linkage when one of the links acts as a driver, and is limited to position calculations; velocity and acceleration calculations have not been attempted. Details of the theory, the mathematical technique, and a description of the program itself are presented.

58. Merrill, R. M., Generalized Three-Dimensional Strain Analysis via Reticulated Framework, Master of Science thesis in Mechanical Engineering, June 1965.

This method of strain analysis is based upon the replacement of the continuous structure by a discretized mechanical structure consisting of nodes connected by elastic members (springs) in such an array that conditions of isotropy are maintained. The nodes are arrayed in a pattern of corners of small cubes. For this array, Poisson's ratio is 0.25. For other Poisson ratios, each cube of the array contains a smaller internal cube whose vertices are spring-connected to the vertices of the larger outside cube.

59. Heubeck, N., Computer-Aided Design of a Ship Hull, Master of Science thesis in Mechanical Engineering, September 1965.

This thesis describes the application of the concepts of computer-aided design for the preliminary design of ships, primarily the description and evaluation of ship hulls. This was accomplished using parametric surface patches developed by Professor S. A. Coons for the hull description, the Project MAC time-shared digital computer for "real-time" communication and the Electronic Systems Laboratory's display as a visual aid.

The hull was evaluated by calculating the displacement, center of buoyancy, wetted surface area, the midships coefficient, waterplane coefficients and prismatic coefficient. Compartments were placed in the ship and the volumes and deck areas were calculated. The display screen showed the cross-sectional area curve, ship's lines and compartment locations.

60. Cheek, T. B., Design of a Low-Cost Character Generator for Remote Computer Displays, Master of Science thesis in Electrical Engineering, February 1966. (Also published as M.I.T. Project MAC Report MAC-TR-26, DDC No. AD631269.)

A requirement exists for a low-cost remote display terminal with alphanumeric and line-drawing capabilities for use with time-shared computer systems. This thesis, conducted as part of the overall remote display design project, was undertaken to investigate novel approaches to character generation, with the goal of drastically reducing present-day costs for such devices.

A survey of existing devices and character generation techniques was carried out, and a design approach was chosen which takes advantage of mass-fabrication techniques. This includes using a five-by-seven dot matrix raster and a resistor array "read-only" character memory for the 96 printable symbols of the Revised Proposed ASCII Code. Circuits designed included a dot matrix generator, and a resistor array memory with selection logic sense amplifiers, and a shift register output buffer.

An experimental character generator with an eight-word memory was built, largely using integrated circuits and was found to work as desired. It is concluded that the design approach will yield a character generator that is of low enough cost to find wide use in remote computer terminals.

61. Evans, D. S., Man-Machine Communication for Simulation of Nonlinear Circuits, Master of Science thesis in Electrical Engineering, February 1966.

The Project MAC time-shared digital computer and the ESL Display Console provide useful facilities for man-machine communication in the design of nonlinear electrical networks. The work described herein constitutes the graphical part of a system that utilizes these facilities to allow simulation of nonlinear electrical networks.

This thesis describes the concepts, data structure, operations and paths of communication that comprise a set of programs which allow the user to specify graphically the network to be analysed, to observe the behavior of the network, and to easily modify the network's structure.

62. Ling, G. C., Investigation of a Semiconductor Laser Data Link for Remote Computer Displays, Master of Science thesis in Electrical Engineering, February 1966.

With a visual display terminal, such as the ESL Display Console, coupled to a time-sharing computer system a user can have his results, whether in character form or graphical form, displayed rapidly on a cathode-ray tube. The ESL Console must be located close to the computer, however. One of the goals of Project MAC at M. I. T. is to have a visual display equipment at every terminal, and the ESL Display Console can be used as a central generating station serving a number of remote stations, provided a multi-megapulse data link is available. If the remote station cannot be easily reached by coaxial cables, which is the most effective way to link the Console with a remote station, then other methods need be investigated. As compared to a TV cable, a UHF broadcasting system, or a microwave link, a laser link offers simple setup and relatively low cost.

Mr. E. J. Chatterton of Lincoln Laboratory has successfully transmitted a continuous train of infrared-light pulses, emitted from a Gallium-Arsenide diode, through a distance of 1.8 miles under various weather conditions and at rates up to 11 megapulses per second. For this

thesis, his experimental setup was borrowed to experiment with its usefulness as a connection between the Console and a remote station. First, an investigation was made of the reliability of the setup as a data link, and a rate of 1.65 megapulses was chosen for the tests. Next, the necessary digital logic for encoding and decoding display information was designed and constructed. Finally, the transmitting end was set up at Technology Square and the receiving end was installed in the Electronic Systems Laboratory. Pictures generated by the Console were successfully received by the receiving end. It is concluded that this is a useful technique, and improvements on the present system for effective operation are suggested.

63. Stratton, W. D., Investigation of an Analog Technique to Decrease Pen-Tracking Time in Computer Displays, Master of Science thesis in Electrical Engineering, February 1966. (Also published as M.I.T. Project MAC Report MAC-TR-25, DDC No. AD631396.)

Through the use of a display and a light-sensitive pen, graphical material can be directly inserted into a computer by using the pen to control the position of the electron beam at the face of the CRT -- a process called pen tracking. In present digital pen-tracking techniques, a tracking pattern (usually a cross) with a substantial number of points is generated on the face of the CRT and the binary response of the pen to the individual points of the pattern is employed to calculate pen position. The large number of pattern points, and the phosphor decay time associated with each, yield a typical tracking cycle of 500 to 1000 microseconds. Since the cycle must be repeated about 100 times per second, 5 to 10 percent of display time is consumed.

To reduce the time required by the tracking operation, an analog technique employing a four-point tracking pattern was investigated, in which the amplitude response of the pen to corresponding pairs of points is used to determine the position of the pen relative to the center of the pattern. One channel of the proposed two-channel analog tracking system was designed, constructed, and coupled to the horizontal channel of the ESL computer display console. To avoid the phosphor-decay limitation, an experimental "beam" pen capable of detecting the electron beam rather than the phosphor luminescence was employed. The system includes a pattern generator, sample-and-hold gates, difference amplifier, envelope detector and noise filter, and a threshold-logic analog-to-digital converter. The time required to generate the tracking pattern and develop the binary equivalent of the

horizontal distance separating pen and pattern center is only 25 microseconds. Tracking is generally satisfactory, but some anomalies were noted, apparently due to the characteristics of the experimental pen being used.

It is concluded that the analog technique is feasible for improving the speed of pen tracking, but recommended that further studies be made of the limitation inherent in the method.

64. Weinberger, M. R., Multi-Stage Random Search and Automatic Network Synthesis, Doctor of Philosophy thesis in Mechanical Engineering, February 1966.

This thesis develops a multi-stage random search strategy which enables one to search for an arbitrarily small, non-zero neighborhood of the absolute minimum (or maximum) of a many-dimensional criterion-surface. The criterion-surface is time-invariant, uni- or multimodal; it further satisfies the hypothesis that, when one considers all possible multi-stage random searches, there exists a probability-distribution for the absolute minimum within all its constant-height surroundings. The strategy is more efficient in that it requires fewer trials than other multi-stage random policies. The strategy predicts average numbers of trials and upper bounds for any chosen probability of convergence towards the minimum. The strategy is self-contained and can be either fully automated or used with human intervention in computer-aided design; it is not overly sensitive to uncertainties in data and to the presence of noise; it can be adapted to a search with limited search effort. The random search is superior to some deterministic search schemes under quantitatively known conditions. The second part of this report applies the general method to the design of state determined systems. These systems may be nonlinear and/or time-variant and their design criteria are continuous functions or functionals of the state-variables. An example is presented in detail to illustrate most of the concepts.

65. Haber, M. R., Two-Dimensional Display of Mathematical Symbols and Expressions, Master of Science thesis in Mechanical Engineering, June 1966.

Using equations stored in list structure form as input, a SLIP program converts these expressions into a visual display. The program will also allow referencing of sub-parts of the equations, with a light pen, for the purpose of making alterations in the source expressions. Necessary parentheses or brackets are inserted into the equation and display is allowed of BCD, integer and

floating point constants as well as variables, operators and functions. In addition, provision is being made to allow the later appending of routines for defining special operators, functions and variables.

66. Parmelee, R. P., Three-Dimensional Stress Analysis for Computer-Aided Design, Doctor of Philosophy thesis in Mechanical Engineering, June 1966.

Two- and three-dimensional general-purpose stress analysis programs have been developed, programmed and tested for the engineer who knows nothing about computers or computer programming. The method of analysis is based on that of Ritz in which finite difference equations are generated by extremal methods from continuous, piece-wise linear approximations to the displacements. Non-uniform placement of nodes appropriate to the boundary and stress distribution of each problem enables the solution of relatively difficult problems such as notched sheets and spheres with networks of as few as 30 to 130 nodes.

67. Starzec, R. E., A Scheme for Information Storage and Retrieval, Master of Science thesis in Mechanical Engineering, June 1966.

This thesis presents a scheme for information storage and retrieval that could be used in a "standard-parts-selection" type of computer-aided design system. It utilizes a symmetric list structure by which descriptions and associations are contained in the pointer rings rather than by repeated storage. All items with some common property lie on one pointer ring. Two sample programs, written in the CORAL language, are contained in the work. One program, BUILDIN', creates structure, and the second program, called SEARCHIN', performs a simple search of the structure.

68. Bernhardt, L. J., A Ten-MC Binary Rate Multiplier for Computer Line Generation, Bachelor of Science thesis in Electrical Engineering, June 1967.

The complexity of computer visual output on display equipment is limited by the time constraints imposed by the need to renew the picture 30 times a second to prevent flicker, and the speed of various display functions. The Electronic Systems Laboratory Display Console designed in 1963 uses a one-mc. binary rate multiplier to perform line generation. This thesis investigated the possibility of using a 10-mc. binary rate multiplier (BRM)

to increase line-generation speed. A six-bit model was built to test timing limitations and the effectiveness of techniques. A 12-bit BRM was then designed to operate at 10-mc. and minimize even further the time needed to draw short lines. It was concluded that the 10-mc. BRM could effectively be used for line generation if oscilloscope deflection techniques for displaying points at this speed could be developed.

69. Blount, F. T., Design of a Low-Cost Computer Graphical Input Device, Bachelor of Science thesis in Electrical Engineering, June 1967.

A low-cost remote computer display terminal which uses a direct-view storage tube to display data has been developed at the M. I. T. Electronic Systems Laboratory. To input graphical data to the computer, a low-cost graphical input device is needed that would allow a person to move a cursor over the storage tube screen in a natural and convenient manner. Present graphic input tablets are quite expensive.

The device built consists of a writing surface or tablet made of Teledeltos paper, which has a low, uniform resistance. The tablet is 11" by 11" with a usable writing space of 9" by 9". A voltage difference of 14.2 volts is applied alternately along opposite pairs of edges at 1 msec. intervals. The voltage at any point on the tablet is detected by a writing instrument (a ball point pen) connected to voltage measuring circuitry. The two output voltages, corresponding to the pen's x and y coordinates, control the cursor position on the display screen, and after A/D conversion, as inputs to the computer.

The accuracy of the device is quite adequate for the intended purpose. Departures from a linear resistance characteristic in the Teledeltos paper introduces a maximum error of only 0.02 volts to the output voltages. The circuitry introduces an error of between 0.1 and 0.3 volts. The accuracy is thus two percent or better. The total parts cost for the device is about $100.

70. Lapin, R. B., Translation Between Artificial Programming Languages, Master of Science thesis in Electrical Engineering, June 1967.

(Also published as Report ESL-R-306, for Abstract see Ref. 38.)

71. Levitt, B. K., High-Resolution Beam Pen, Master of Science thesis in Electrical Engineering, June 1967.

Light-pen systems are currently used to provide a rapid transfer of graphical information between a computer and its operator. For certain applications, however, light-pen response time (determined by photocell rise time and phosphor decay time) is too slow, and the need for a faster system is evident. The beam-pen system constructed by the Electronic Systems Laboratory at M. I. T. is a sufficiently fast device, but its resolution is not comparable to that of the light pen.

A theoretical analysis indicated that the resolution could be significantly improved through the use of a simulated matched-filter signal-processing technique involving synchronous detection followed by gated low-pass filtering. A new beam-pen system was developed in compliance with the above signal-processing scheme, and tests indicate it indeed possesses a satisfactorily high resolution. A final modification is suggested which will result in a fully-operable beam-pen system which will be particularly useful where response time is critical.

72. Vassar, E. R., Audio-Coupled Receiver-Demodulator for Telephone-Transmitted Digital Data, Bachelor of Science thesis in Electrical Engineering, June 1967.

The increasing use of multiple, remote input-output devices, such as display terminals, has created a demand for an inexpensive, portable data receiver which can retrieve modulated digital data from the telephone system channel. Present digital data transmission via telephone requires installation of data modems. An attractice alternate is the utilization of an inductive pickup and demodulator with an ordinary telephone handset. Such devices commercially available operate only up to about 200 bits per second. The design and test of such a device, which might be useful at bit rates up to 600 bits per second, is described in this report. Concluding suggestions are advanced for an improved version which could push up the bit rate to the desired frequency of 1200 bits per second or more.

73. Rodriguez, J. E., A Graph Model for Parallel Computations, Doctor of Science thesis in Electrical Engineering, September 1967.

This thesis presents a computational model called program graphs which makes possible a precise description of parallel computations of arbitrary complexity on non-structured data. In the model, the computation steps are represented by the nodes of a directed graph whose links represent the elements of storage and transmission of data and/or control information. The activation of the computation represented by a node depends only on the control information residing in each of the links incident into and out of the node. At any given time any number of nodes may be active, and there are no assumptions in the model regarding either the length of time required to perform the computation represented by a node or the length of time required to transmit data or control information from one node to another. Data dependent decisions are incorporated in the model in a novel way which makes a sharp distinction between the local sequencing requirements arising from the data dependency of the computation steps and the global sequencing requirements determined by the logical structure of the algorithm.

The concept of the state of a program graph is introduced and it is proved that every program graph represents a deterministic computation, i.e., that the final state of each computation started from the same initial state is unique. Computations which do not terminate properly are defined in terms of the concept of hand-up state. Methods of analysis are developed and necessary and sufficient conditions for the absence of hang-up states are obtained. These conditions are interpreted in terms of the structure of the graph and the manner in which the decision elements are imbedded in that structure. Finally, an equivalence problem for program graphs is formulated and a solution to this problem is presented.

D. TECHNICAL PAPERS AND PUBLICATIONS

74. Feldmann, C. G., "Automatic Data Processing for Numerically Controlled Machine Tools", presented at the ASME Production Engineering Conference, May 18, 1960. (Reprinted in Technical Memorandum 8436-TM-1, see Ref. 27.)

75. Ross, D. T., "A Generalized Technique for Symbol Manipulation and Numerical Calculation", presented at the ACM Conference on Symbol Manipulation, Philadelphia, Pa., May 20-21, 1960 and published in the Communication of the ACM, March, 1961.

76. Ross, D. T., "The Design and Use of the APT Language for Automatic Programming of Numerically Controlled Machine Tools," presented at the Sixth Annual Computer Applications Symposium, Armour Research Foundation of the Illinois Institute of Technology, Chicago, Illinois, October 29, 1959. (Reprinted in Technical Memorandum 8436-TM-1, see Ref. 27.)

77. Ross, D. T., "Man-Machine Aspects of Automatic Programming for Numerically-Controlled Machine Tools," presented at the AIEE Winter General Meeting held in New York City, February 2, 1961.

78. Ross, D. T., "Data Processing for Numerical Control in the USA," presented at the International Federation for Information Processing (IFIP) Congress 62, Munich, Germany, Aug. 27-Sept. 1, 1962, published in the Proceedings, pp. 259-260.

The following five papers formed a dedicated session on Computer-Aided Design at the 1963 Spring Joint Computer Conference, May 23, 1963, in Detroit, Michigan. All five papers were published in the AFIPS Conference Proceedings (American Federation of Information Processing Societies), Vol. 23.

79. Coons, S. A., "Outline of the Requirements for a Computer-Aided Design System." (Also published as Technical Memorandum ESL-TM-169, for Abstract see Ref. 24.)

80. Ross, D. T., and Rodriguez, J. E., "Theoretical Foundations for the Computer-Aided Design System." (Also published as Technical Memorandum ESL-TM-170, for Abstract see Ref. 25; also published in Simulation, Vol. 2, No. 4, March 1964, pp. R-8 to R-20.)

81. Stotz, R. H., "Man-Machine Console Facilities for Computer-Aided Design." (Material condensed from Master of Science thesis, Ref. 49, which was also published as Technical Memorandum ESL-TM-163; for Abstract see Ref. 23; also published in Simulation, Vol. 2, No. 4, March 1964, pp. R-3 to R-7.)

82. Sutherland, I. E., "Sketchpad; A Man-Machine Graphical Communication System." (Material condensed from Doctor of Science thesis, Ref. 50, which was also published as MIT Lincoln Laboratory Report TR-396; for Abstract see Ref. 22.)

83. Johnson, T. E., "Sketchpad III, Three-Dimensional Graphical Communication with a Digital Computer." (Material condensed from Master of Science thesis, Ref. 51, which was also published as Technical Memorandum ESL-TM-173; for Abstract see Ref. 26.)

84. Ross, D. T., "On Context and Ambiguity in Parsing," presented at the Working Conference on Mechanical Language Structures held at IDA, Princeton, New Jersey, August 14-16, 1963; also published in the Communications of the ACM, Vol. 7, No. 2, pp. 131-133, February 1964.

85. Ross, D. T., "Implications of Computer-Aided Design for Numerically-Controlled Production," presented at the First Annual Meeting and Technical Conference of the Numerical Control Society in New York on March 20, 1964; also published in the Conference Proceedings, pp. 100-111. (Also published as Technical Memorandum ESL-TM-212; for Abstract see Ref. 29.)

86. Ross, D. T., and Feldmann, C. G., "Verbal and Graphical Language for the AED System: A Progress Report," presented at the MIT Industrial Liaison Symposium on Project MAC, May 6, 1964; also published as a Project MAC Report MAC-TR-4. (Also included as Appendix in ESL Interim Technical Report ESL-IR-221, Ref. 8.)

87. Ward, J. E., "Display Consoles for the MAC System," presented at the MIT Industrial Liaison Symposium on Project MAC, May 6 and 7, 1964. (Published as Project MAC Report MAC-TR-5, June 1964.)

88. Knuth, D. E., Bumgarner, L. L., Ingerman, P. Z., Merner, J. N., Hamilton, D. E., Lietzke, M. P., Ross, D. T., "A Proposal for Input-Output Conventions in Algol 60," A Report of the Subcommittee on Algol of the ACM Programming Languages Committee, Communications of the ACM, Vol. 7, No. 5, May 1964, pp. 273-283.

89. Ross, D. T., "Automated Engineering Design AED-1," presented at the SHARE Design Automation Workship, Atlantic City, N.J., June 24-26, 1964, also published in the Proceedings.

90. Johnson, T. E., "Sketchpad," presented at the SHARE Design Automation Workshop, Atlantic City, N.J., June 24-26, 1964, also published in the Proceedings.

91. Lang, C. A., and Polansky, R. B., "Graphical Language -- A Step Towards Computer-Aided Design," IEEE Machine Tools Conference, Hartford, Conn., November 16-17, 1964. Conference Paper No. CP64-575.

92. Ross, D. T., "Computer-Aided Design," presented at the 29th Annual Machine Tool Electrification Forum (sponsored by Westinghouse), Buffalo, N.Y., April 27-29, 1965. Paper No. 304-11N, published in the Proceedings.

93. Feldmann, C. G., "Programming in AED-0," presented at the Spring APT Technical Meeting, Chicago, Illinois, April 27-30, 1965, published in the Proceedings.

94. Ross, D. T., "System Provisions for the Debugging of AED Language Programs," presented at the IFIPS Congress 1965, New York City, May 24-29, 1965, published in the Proceedings, Vol. 2, pp. 593-595.

95. Ross, D. T., "The MIT Computer-Aided Design Project -- A Half Decade Summary," presented at the 4th Air Force Materials Symposium, Miami Beach, Florida, June 9-11, 1965, published in the Proceedings, pp. 774-797.

96. Ross, D. T., Lecture Notes Summer Course on Computer Graphics, University of Michigan, June 14-18, 1965.

97. Ross, D. T., "Current Status of AED," presented at the 2nd Annual SHARE Design Automation Workshop, Atlantic City, N. J., June 23-25, 1965, published in the Proceedings.

98. Haring, D., "The Beam Pen: A Novel High-Speed, Input/Output Device for Cathode-Ray-Tube Display Systems," presented at Fall Joint Computer Conference, Las Vegas, Nevada, Nov. 30-Dec. 2, 1965, AFIPS Conference Proceedings, Vol. 27, Part I, pp. 647-855.

99. Mann, R. W., and Coons, S. A., "Computer-Aided Design," McGraw-Hill 1965 Yearbook of Science and Technology, 9 pp. (Also reprinted in Report of First Meeting of DOD/AOA Discussion Group on Computer-Aided Design and Documentation, March 1966, pp. D-2 to D-12.)

100. Wolman, B. L., "Operators for Manipulating Language Structures," presented at the Symposium on Symbolic and Algebraic Manipulation, Association for Computing Machinery, Washington, D.C., March 29-31, 1966. (Abstract published in ACM Communications, Vol. 9, No. 8, August 1966, pp. 553-554, see also Ref. 36.)

101. Katzenelson, J., "Simulation for Nonlinear Circuits by AEDNET," presented at the 3rd Annual SHARE Design Automation Workshop, New Orleans, La., May 16-19, 1966. (Abstract published in the Proceedings.)

102. Ward, J. E., "The MIT CAD Project," review of project work presented at the Second Meeting of the DOD/AOA Discussion Group on Computer-Aided Design and Documentation, June 23, 1966. Published in the Report of the meeting, pp. B-6-B-8.

103. Ward, J. E., "Display Hardware for Dynamic Man-Machine Interaction," presented at the 20th Annual AFCEA Convention, Washington, D.C., June 7-9, 1966; published in Conference

Proceedings, and also in SIGNAL, Journal of the Armed Forces Communication and Electronics Association, Vol. XX, No. 11, July 1966, pp. 54-61.

104. Ward, J. E., "Systems Engineering Problems in Computer-Driven CRT Displays for Man-Machine Communication," presented at the 1966 IEEE System Science and Cybernetics Conference, Washington, D. C., October 18, 1966. Published in the Conference Record, also published in the IEEE Transaction on Systems Science and Cybernetics, Vol. SSC-3, No. 1, June 1967, pp. 47-54.

105. Katzenelson, J., "AEDNET: A Simulator for Nonlinear Networks," Proceedings of the IEEE, Vol. 54, No. 11, November 1966, pp. 1536-1552.

106. Katzenelson, J., Evans, D. S., and Lee, H. B., "A Program for On-Line Analysis of Nonlinear Electronic Circuits," presented at the IEEE International Convention, New York, March 20-25, 1967. Published in the Convention Record.

107. Ross, D. T., "The Computer-Aided Design Project at MIT," presented at the Interdisciplinary Conference on Computer Graphics, Ohio State University, April 5, 1967. Published in the Proceedings.

108. Ross, D. T., "An Approach to Automated Engineering Design," presented at the AOA Annual Meeting, San Francisco, May 9-11, 1967. Abstract published in the Proceedings.

109. Stotz, R. H., and Cheek, T. B., "A Low-Cost Graphic Display for a Computer Time-Sharing Console," presented at the 8th National Symposium on Information Display, San Francisco, Calif., May 24-26 1967. Published in the Technical Session Proceedings, Society for Information Display, pp. 91-100. (Also published as Technical Memorandum ESL-TM-316, July 1967.)

110. Ross, D. T., Stotz, R. H., Thornhill, D. E., and Lang, C.A., "The Design and Programming of a Display Interface System Integrating Real-Time and Time-Shared Computers," presented at the 4th Annual ACM/SHARE Design Automation Workshop, Los Angeles, June 1967, published in the Proceedings.

111. Evans, D. S., and Katzenelson, J., "Data Structure and Man-Machine Communications for Network Problems," Proceedings of the IEEE, Vol. 55, No. 7, July 1967, pp. 1135-1144.

112. Ross, D. T., "The AED Approach to Generalized Computer-Aided Design," presented at the 1967 ACM National Conference, Washington, D. C., August 1967, pp. 367-385. (Also published as Report ESL-R-305, for Abstract see Ref. 37.)

113. Ross, D. T., "The AED Free Storage Package," Communications of the ACM, Vol. 10, No. 8, pp. 481-492, August 1967.

114. Ross, D. T., "Features Essential for a Workable Algol X," ALGOL Bulletin, No. 26, August 1967, pp. 6-12, and ACM Sigplan Notices, Vol. 2, No. 11, November 1967.

E. LECTURES AND TECHNICAL PRESENTATIONS WITHOUT PUBLICATION

115. Ross, D. T., "Logarithmic Search and the First-Pass Algorithm," talk at the MITRE Corp, Lexington, Mass., February 20, 1962.

116. Coons, S. A., Description of the Project work in a lecture at the Engineering Management Research and Development Institute of the General Electric Company, Crotonville, N. Y., May 22, 1962.

117. Coons, S. A., Description of the Project work in connection with the NSF Commission on Graphics Course Content Study, Princeton University, May 7-8, 1962.

118. Coons, S. A., "Computer-Aided Design," lecture at Worcester Polytechnic Institute, October 16, 1962.

119. Coons, S. A., Description of the Project work to a group of Engineering Managers as part of the Modern Engineering Program, General Electric Company, Saratoga Springs, N. Y., November 14 and 15, 1962.

120. Coons, S. A., "Computer-Aided Design," talk at an Engineering Seminar, IBM Corporation, New York, February 1963.

121. Coons, S. A., "Computer-Aided Design," lecture at the Mechanical Engineering Department, University of New Hampshire, March 1963.

122. Mann, R. W., talks on Computer-Aided Design at North Carolina State University and Ohio State University, March 1963.

123. Ross, D. T., "Breaking the Man-Machine Language Barrier," talk to the Middletown Scientific Society, Middletown, Conn., April 1963.

124. Johnson, T. E., "Computer-Aided Design and Sketchpad III," talk and movie presented at the Salem-Lynn Division of the AIEE-IRE, April 1963.

125. Coons, S. A., "Computer-Aided Design and Sketchpad," talk and movie presented at the General Electric Management Institute, Crotonville, N. Y., May 1963.

126. Mann, R. W., "Computer-Aided Design," talk at the University of Colorado, Denver, Colorado, May 1963.

127. The following group of seven talks were given at an MIT Industrial Liaison Symposium, Computer-Aided Design, May 9, 1963, co-chaired by D. T. Ross and Professor S. A. Coons. Registrants from government and industry totaled 201.

Reintjes, Professor J. F., "Background of the Project"

Mann, Professor R. W., "Criteria for the Computer-Aided Design System"

Coons, Professor S. A., "Graphical Communications and Problem Solving"

McClintock, Professor F. A., "Stress Analysis Techniques"

Ross, D. T., "Theoretical Foundations for Computer-Aided Design"

Ross, D. T., "Structure and Operation of the AED System"

Ward, J. E., "Man-Machine Console Facilities"

128. Coons, S. A., Description of the Project work at an ASEE Meeting, Philadelphia, Pa., June 1963.

129. Ross, D. T., and Coons, S. A., "Computer-Aided Design," talks at Bell Telephone Laboratories, Murray Hill, N. J., and International Telephone and Telegraph Company, Nutley, N. J., June 25 and 26, 1963.

130. Ross, D. T., "Computer-Aided Design," talk at an Engineering Seminar, IBM Corporation, Kingston, N. Y.

131. Coons, S. A., talks on Computer-Aided Design at the Ford Motor Company, the Chrysler Corporation, and the General Motors Corporation, August 21-23, 1963.

132. Ross, D. T., "Computer-Aided Design," talk at the IBM Mohansic Laboratories, Poughkeepsie, N. Y., September 17, 1963.

133. Coons, S. A., "Computer-Aided Design," talk at the Computer Division of the Philco Corporation, Philadelphia, Pa., October 7, 1963.

134. Coons, S. A., "Computer-Aided Design," talk at the Engineering Institute of Design and Drafting Automation, University of Wisconsin, October 10, 1963.

135. Coons, S. A., "Computer-Aided Design," talk at the Arthur D. Little Corp., Cambridge, Mass., October 17, 1963.

136. Ross, D. T., "Computer-Aided Design," talk at an MIT Electrical Engineering Colloquium, October 23, 1963.

137. Coons, S. A., "Computer-Aided Design," talk at the Ford Motor Company, Detroit, Michigan, October 23, 1963.

138. Coons, S. A., "Computer-Aided Design," talk at the General Electric Engineering Management School, Saratoga Springs, N.Y., November 15, 1963.

139. Ross, D. T., Ward, J. E., and Johnson, T. E., description of Project work at the IEEE Display Workshop, Las Vegas, Nevada, November 16, 1963.

140. Ross, D. T., "Computer-Aided Design," talks at the Sandia Corporation, Livermore, California, and Albuquerque, New Mexico, December 12 and 13, 1963.

141. Ross, D. T., member of panel on Numerically Controlled Machines, 1964 SAE Automotive Engineering Congress and Exposition, Detroit, Michigan, January 13-17, 1964.

142. Coons, S. A., "Computer-Aided Design," lecture at an Electrical Engineering Department Symposium, Brooklyn Polytechnic Institute, January 1964.

143. Coons, S. A., talks on Computer-Aided Design and mathematical methods for surface description at IBM-Poughkeepsie, IBM-Kingston, and IBM-New York, February 5-6, 1964.

144. Coons, S. A., "Computer-Aided Design," talk at Westinghouse, Pittsburgh, Pa., February 18, 1964.

145. Coons, S. A., "Computer-Aided Design," talk at Gulf Research and Development Corporation, Pittsburgh, Pa., February 19, 1964.

146. Coons, S. A., "Computer-Aided Design," lecture at the Cooper Union, New York, N. Y., March 24, 1964.

147. Ross, D. T., "Algorithmic Theory of Language," Graduate Colloquium on Systems and Communications, Carnegie Institute of Technology, May 14, 1964.

148. Coons, S. A., "Computer-Aided Design," talk at General Electric Modern Engineering Course, Saratoga Springs, N. Y., May 8, 1964.

149. Coons, S. A., talk and movies on Computer-Aided Design at the Engineering Summer Conference sponsored by the University of Detroit and the University of Michigan, August 3-4, 1964.

150. Coons, S. A., "Computer-Aided Design and Sketchpad," talk at the American Can Company, New York, N. Y., October 20, 1964.

151. Ross, D. T., panelist on session on Input and Output of Graphics, Fall Joint Computer Conference, San Francisco, October 26-29, 1964.

152. Coons, S. A., "Computer-Aided Design and Sketchpad," talk at Raytheon Corporation, Boston, Mass., October 28, 1964.

153. Ross, D. T., "Computer-Aided Design," talk at IBM Engineering Seminar, Poughkeepsie, N. Y., April 9, 1965.

154. Coons, S. A., "Computers in Design," talk at the ASTME Conference and Exposition, April 1, 1965.

155. Coons, S. A., "Computer-Aided Design," talk at the Post-College Education Program, Carnegie Institute of Technology, April 12, 1965.

156. Coons, S. A., talk on Computer-Aided Design at the IBM symposium on Man-Machine Communication, Yorktown Heights, N. Y., May 4, 1965.

157. Coons, S. A., "Computer-Aided Design," talk at the AOA Meeting in Santa Monica, California, May 7, 1965.

158. Coons, S. A., "Surface Mathematics," talk to Lockheed-California and Lockheed-Georgia personnel in California, May 8, 1965.

159. Coons, S. A., "Computer-Aided Design," lecture at University of California in Berkeley, May 15, 1965.

160. Coons, S. A., lecture on Computer-Aided Design at the AIChE Information Processes and Computers Symposium, San Francisco, May 18, 1965.

161. Coons, S. A., "Computer-Aided Design," talk presented at the DECUS (Digital Equipment Corporation Users Society) Meeting at Harvard University, May 20, 1965.

162. Ross, D. T., lectures on Computer-Aided Design at the Summer Conference on Computer Graphics, University of Michigan, June 14-16, 1965.

163. Coons, S. A., talk on Computer-Aided Design at the Fourth Air Force Materials Symposium, Miami Beach, Florida, June 9-11, 1965.

164. Ross, D. T., "A Perspective on the Language Problem," presented at the AMA Briefing Session on Software -- The Computer Software Decade, Roosevelt Hotel, New York City, July 26, 1965.

165. Coons, S. A., "Computer-Aided Design," talk at the Summer Conference on Application of Computers to Automated Design, University of Michigan, August 2-3, 1965.

166. Ross, D. T., "Recent Advances in Programming Languages," a panel discussion at the ACM National Conference, Cleveland, Ohio, August, 24-27, 1965.

167. Stotz, R. H., "Computer Displays," talk at the IEEE Display Workshop, San Mateo, California, August 27-28, 1965.

168. Coons, S. A., talks on Computer-Aided Design at General Motors and the Society of Body Engineers, Detroit, Michigan, September 16, 1965.

169. Coons, S. A., talk on Computer-Aided Design at the Drafting Automation Institute, University of Wisconsin, October 15, 1965.

170. Polansky, R. B., "Computer-Aided Design," presented at an IEEE Meeting, Worcester Polytechnic Institute, Worcester, Mass., October 19, 1965.

171. Sibley, E. H., "Computer-Aided Design," talk before the Executives of Sacony-Mobile, Philadelphia, Pa., October 21, 1965.

172. Coons, S. A., "Computer-Aided Design," talk at the National Electronics Conference, Chicago, Illinois, October 27, 1965.

173. Ross, D. T., "Computer-Aided Design," presented at an Advanced Engineering Program Seminar, MIT, November 2, 1965.

174. Coons, S. A., talks on Computer-Aided Design at the University of Nebraska and Wichita State University, December 7 and 8, 1965.

175. Ward, J. E., "Display Hardware for Graphical Man-Machine Communication," talk presented at the MIT Industrial Liaison Symposium on Sensing, Analyzing, and Processing Visual Information, December 21, 1965 (267 registrants).

176. Katzenelson, J., "On-Line Simulation of Nonlinear Electrical Networks," talk presented at the MIT Project MAC Seminar on February 1, 1966.

177. Ross, D. T., and Ward, J. E., talks on Computer-Aided Design and Computer Graphics at a special MIT Industrial Liaison Program Meeting for executives of Laboratory for Electronics, Inc., February 2, 1966.

178. Coons, S. A., "Computer-Aided Design: The Designer's Viewpoint," paper presented at the IEEE Convention in Los Angeles, California, February 3, 1966.

179. Ward, J. E., "Display Hardware for Project MAC," Engineering Seminar at IBM, Kingston, N. Y., March 9, 1966. (Description of ESL Console and related developments.)

180. Katzenelson, J., "Nonlinear Networks and their Simulation on Digital Computers," series of lectures at the University of California, Berkeley, March 14-18, 1966.

181. Ross, D. T., "The AED-1 Compiler for the GE-645," talk presented at the MIT Project MAC Seminar, April 5, 1966.

182. Ross, D. T., "Structure of a Time-Shared Hardware-Software System for Sophisticated Local and Remote, On-Line, Mixed Graphical and Verbal Problem-Solving," presented at the IBM Programmer's Club, Yorktown Heights, N. Y., April 15, 1966.

183. Reintjes, J. F., "The Role of Computers in Modern Design Technology," presented at the University of Wisconsin Conference on Computer-Aided Solid State Circuit Design, May 3-4, 1966.

184. Katzenelson, J., "Simulation of Nonlinear Circuits by AEDNET," paper presented at the Third Annual SHARE Design Workshop, New Orleans, Louisiana, May 16-19, 1966.

185. Ross, D. T., "The AED Approach to User-Oriented Languages for Automated Engineering Design," talk presented at the RCA Computer Seminar, Princeton, N. J., November 30, 1966.

186. Ward, J. E., presentation and discussion of movie of computer graphics produced on the ESL Console, Symposium on The Growth of Computer Graphics, Marshall Space Flight Center, Huntsville, Alabama, March 1-3, 1967 (jointly sponsored by MSFC, The Army Missile Command, and the Assoc. for Computing Machinery.)

187. Ross, D. T., "The AED Approach," talk given at the APT Interface Task Group Meeting, Cambridge, Mass., April 4, 1967. (Included demonstrations of graphics hardware and software at Project MAC by D. E. Thornhill and R. B. Polansky.)

188. Ross, D. T., "The Computer-Aided Design Project at MIT," talk presented at the Interdisciplinary Conference on Computer Graphics at Ohio University in Columbus, Ohio, April 5, 1967.

189. Ross, D. T., "The AED Compiler System: Status and Plans," talk presented at the ACM SICPLAN/PLANCOM Workshop on Compiler-Building Tools at Atlantic City, N. J., April 16-17, 1967.

190. Ward, J. E., "AED," talk given at the Northrop-MIT Technical Meeting held at MIT, April 27, 1967.

191. Ross, D. T., "The AED Cooperative Program -- A University-Industry Collaboration in Computer-Aided Design," talk given at the Boston Chapter ACM Meeting in Cambridge, Mass., April 27, 1967.

192. Ross, D. T., "An Approach to Automated Engineering Design," presented at the AOA 9th Annual Spring Meeting, San Francisco, California, May 9-11, 1967.

F. BIBLIOGRAPHY OF INFORMAL AED DOCUMENTS

The following is the recommended current list of documents describing the use of the AED language and systems, and is based on the AED Bibliography by D. T. Ross, ESL Memorandum 70429-M-153-2 (Project MAC Memorandum MAC-M-278-3), September 1, 1967. Since parts of the system and the documentation are continually being reworked, these documents in many cases supercede earlier ones, and any documents dated prior to May, 1968, that are not listed, may be considered obsolete.

These documents are all in an informal memorandum series distributed only to those expressing a specific interest in AED details. In addition to the ESL memorandum number, most also carry a number in the MIT Project MAC memorandum series, which is widely distributed at M. I. T. Many also are identified as belonging to a series of "AED Flashes" or "AED Progress Reports", distributed to the organizations participating in the AED Cooperative Program. For convenience in reference, the list has been organized into subject groupings: Basic Use of AED-0 Language, AEDJR, Batch Processing, etc.

Finally, a number of the documents (indicated by an asterisk) have been grouped into a packet called the "AED User Kit", which can be requested as an entity. The AED User Kit, which contains some 730 pages, is the basic documentation needed to use AED. (The total AED bibliography listed in this section is 730 + 662 = 1,392 pages.)

General Information and Administrative Documents

193. An Invitation to Participate in the Cooperative AED Project for Computer-Aided Design by D. T. Ross, December 14, 1965, 15 pp. (Also reprinted in ESL-IR-278, Ref. 11.)

> This memorandum extends an invitation to participate in a continuation of the AED Cooperative Program conducted as a component of the MIT Computer-Aided Design Project. Interested organizations are invited to participate by nominating an experienced system programmer to be sent to M. I. T. to join with our regular Project staff in system research and development for a nominal one-year

period. Section I of this memorandum presents information of interest to those making the decision concerning company participation. Project goals, organization, results to date, and plans for next year are described. The benefits of company participation are presented. Section II describes how to report the company decision regarding participation at M. I. T., and the mechanics of participation. The final page provides a check list for use in response to this invitation. Supporting technical information and background material is summarized in an accompanying document (Document 194 below).

194. AED Project Technical Information by D. T. Ross, December 14, 1965, 37 pp. (Also reprinted in ESL-IR-278, Ref. 11.)

This document presents technical information supporting the "Invitation to Participate" (Document 193 above). The approach of the Project to the problems of computer-aided design, current status, and planned activities are outlined.

195. *General Description of the AED-1 Processor and the Display Interface System, D. T. Ross, ESL Memorandum 9442-M-170/MAC-M-312, June 16, 1966, 36 pp.

196. The AED Approach to Generalized Computer-Aided Design, D. T. Ross, M. I. T. Electronic Systems Laboratory Report ESL-R-305, April 1967, 55 pp.

(For Abstract see Ref. 37 which is the same document.)

197. AED Invitation Kit, D. T. Ross, ESL Memorandum 70429-M-193, October 5, 1967, 122 pp.

This document has been prepared as an introduction to the AED Cooperative Program, an important on-going feature of the MIT Computer-Aided Design Project. The Invitation Kit currently includes the following items:

1. Copies of the four invitation letters, from December 1965 through June 1967.
2. A summary of participants in the AED Cooperative Program.

---

*Asterisk indicates document is included in "The AED User Kit".

3. Document 196 (The AED Approach to Generalized Computer Aided Design).
4. The Notice of the Second AED Technical Meeting, January 1967.
5. Document 193 (An Invitation to Participate ---).
6. Document 194 (AED Project Technical Information).

Basic Use of AED-0 Language

198. *AED-0 Programming Manual -- Preliminary Releases Nos. 1-4, D. T. Ross, October - December 1964, 142 pp.

199. *Modifications and Addenda to the AED-0 Programmer's Guide, C. Feldmann, C. Bower, and P. Ladd, ESL Memorandum 9442-M-143/MAC-M-261, August 18, 1965, 65 pp.

200. *AED Flash No. 34 - Available AED System Macros, C. Feldmann and R. Lynn, ESL Memorandum 9442-M-177/MAC-M-325, September 14, 1966, 14 pp.

201. *AED Flash No. 27 - New CTEST2 Command, C. Feldmann, ESL Memorandum 9442-M-158/MAC-M-291, January 7, 1966, 3 pp.

202. *AED Flash No. 2 - Machi Usage, R. Bigelow, ESL Memorandum 9442-M-116/MAC-M-207, December 8, 1964, 5 pp.

203. *New Free Storage Package, D. T. Ross, ESL Memorandum 9442-M-173/MAC-M-318, July 15, 1966, 27 pp.

204. *AED Flash No. 5 - Preset Usage, B. Wolman, ESL Memorandum 9442-M-116/MAC-M-207, December 8, 1964, 4 pp.

205. *AED Flash No. 20 - New Array Handling Language for AED-0, P. Ladd, ESL Memorandum 9442-M-145/MAC-M-271, September 24, 1965, 3 pp.

206. *AED Flash No. 33 - The FEATURES Feature, D. T. Ross and C. Feldmann, ESL Memorandum 9442-M-176/MAC-M-321, August 15, 1966, 7 pp.

207. *Converting McCracken's Algol Book to Describe AED-0, D. T. Ross and N. D. Fulton, ESL Memorandum 70429-M-200/MAC-M-366, February 14, 1968, 9 pp. (Also Errata Sheet 70429-M-200.1/MAC-M-366.1)

208. *Directions for Updating the AED-0 Programmer's Guide, plus an Index of Terms, and a Description of AED-0 Items, N. D. Fulton and D. T. Ross, ESL Memorandum 70429-M-198-1/MAC-M-363-1, February 19, 1968, 25 pp.

Frequently Used Packages which Extend AED-0 Compiler Features

209. *AED Flash No. 13-2 - Argument Checking Procedures for AED, J. Walsh, ESL Memorandum 9442-M-125-2/MAC-M-225-2, April 15, 1965, 3 pp.

210. *AED Flash No. 15 - Execution Time Procedure Calls, J. Walsh, ESL Memorandum 9442-M-131/MAC-M-236, April 22, 1965, 4 pp.

211. *AED Flash No. 23 - Use of GENCAL, J. Walsh, ESL Memorandum 9442-M-152/MAC-M-275, October 14, 1965, 3 pp.

AED-0 Programming Techniques

212. *AED Flash No. 12 - Multi-Entry Procedures, D. T. Ross, ESL Memorandum 9442-M-124/MAC-M-222, February 25, 1965, 3 pp.

Packages for System Building

213. *AED Flash No. 26 - BSS Plex Dump, R. Ladson, ESL Memorandum 9442-M-156/MAC-M-283, November 2, 1965, 12 pp.

214. The New Generalized String Package, D. T. Ross, ESL Memorandum 9442-M-164, May 2, 1966, 55 pp.

215. A Change in the New String Package, A. Mills, ESL Memorandum 9442-M-164-1, July 20, 1966, 2 pp.

216. AED Flash No. 31 - FOCL-Frame Oriented Command Language, W. D. Maurer, ESL Memorandum 9442-M-172/MAC-M-317, July 14, 1966, 5 pp.

217. *AED Flash No. 36 - The ASEMBL Package, B. Wolman and C. Feldmann, ESL Memorandum 9442-M-180/MAC-M-331, October 28, 1966, 16 pp.

218. *AED Flash No. 35 - Line-Assemble Package, P. Ladd and C. Feldmann, ESL Memorandum 9442-M-179/MAC-M-330, October 14, 1966, 10 pp.

219. *AED Flash No. 37 - The Alarm Package, A. Mills and C. Feldmann, ESL Memorandum 9442-M-185/MAC-M-343, January 31, 1967, 18 pp.

220. *AED Flash No. 38 - Delayed Merge Program, B. Wolman, ESL Memorandum 9442-M-186/MAC-M-344, February 15, 1967, 17 pp.

221. *AED Flash No. 39 - RWORD Package, S. Ackley, W. Johnson, and J. Porter, ESL Memorandum 70429-M-189/MAC-M-350, May 29, 1967, 25 pp.

222. *AED Flash 36.1 - An End-of-Line Routine for Use with the ASEMBL Package, F. Bates, ESL Memorandum 70429-M-180.1/MAC-M-331.1, November 9, 1967, 5 pp.

223. *AED Flash No. 42 - Basic Input/Output Package, C. G. Feldmann and D. T. Ross, ESL Memorandum 70429-M-199/MAC-M-365, March 7, 1968, 13 pp.

AEDJR

224. Preliminary Documentation for the 1966 AEDJR System, R. Lapin, D. T. Ross, and R. Wise, ESL Memorandum 9442-M-182, January 5, 1967, 106 pp.

225. *Operators for Manipulating Language Structures, B. Wolman, ESL Memorandum 9442-M-160/MAC-M-304, March 18, 1966, 28 pp. (For Abstract see Ref. 36.)

226. *AED Flash No. 40 - The Top-Down Mouse (APRAL), I. Wenger, ESL Memorandum 70429-M-191/MAC-M-354, July 12, 1967, 8 pp.

Batch Processing

227. *AED-0 Compiler for Batch Processing, H. W. Spencer, ESL Memorandum 9442-M-115/MAC-M-204, December 2, 1964, 10 pp.

228. *FMS Tape Distribution, C. Feldmann, J. Rodriguez, and H. Spencer, ESL Memorandum 9442-M-127, April 13, 1965, 13 pp.

229. *FMS Tape Distribution, C. Feldmann, ESL Memorandum 9442-M-127-2, April 26, 1965, 3 pp.

230. *FMS Tape Distribution, C. Feldmann and L. Walker, ESL Memorandum 9442-M-127-3/MAC-M-253, July 1, 1965, 10 pp.

231. *FMS Tape Distribution, C. Feldmann, ESL Memorandum 9442-M-127-4, January 25, 1966, 16 pp.

232. *Maintenance Release No. 1, C. Feldmann, ESL Memorandum 9442-M-127-5, July 28, 1966, 3 pp.

233. *Distribution of 1966 AEDJR and Other Maintenance Information, C. Feldmann, ESL Memorandum 9442-M-127-6, March 8, 1967, 8 pp.

234. *Maintenance Release No. 2, C. Feldmann, ESL Memorandum 9442-M-127-7, March 15, 1967, 6 pp.

235. *AED Flash No. 41 - 7094 to 360 Conversion using AED, C. G. Feldmann, ESL Memorandum 70429-M-194/MAC-M-361, October 19, 1967, 12 pp.

236. *AED-0 Compiler for Batch Processing, C. G. Feldmann, ESL Memorandum 70429-M-115-1/MAC-M-204-1, March 12, 1968, 7 pp.

237. Running AED on the MIT 360, C. G. Feldmann, ESL Memorandum 70429-M-201/MAC-M-373, May 2, 1968, 55 pp.

Graphic Processing

238. Operating Manual for the ESL Display Console, R. Stotz and J. Ward, ESL Memorandum 9442-M-129/MAC-M-217, March 9, 1965, 47 pp.

239. New B-Core System for Programming the ESL Display Console, C. Lang, ESL Memorandum 9442-M-122/MAC-M-216, April 30, 1965, 87 pp.

240. Additions to the New B-Core System, C. Lang, ESL Memorandum 9442-M-122-2/MAC-M-216-2, August 2, 1965, 2 pp.

241. ESL Display Console Operating System Manual, R. Bayles, ESL Memorandum 9442-M-118/MAC-M-201, December 10, 1964, 18 pp.

242. Three-Dimensional Pseudo Pen Subroutine for use with the ESL Display Console, R. Polansky, ESL Memorandum 9442-M-138/MAC-M-256, July 13, 1965, 25 pp.

243. Surfaces for Computer-Aided Design of Space Figures, S. Coons, ESL Memorandum 9442-M-139/MAC-M-255, July 21, 1965, 29 pp.

244. AED Flash No. 18 - A Subroutine for Generating Coons' Surfaces, C. Lang, ESL Memorandum 9442-M-137/MAC-M-252, June 30, 1965, 4 pp.

245. AED Flash No. 22 - Use of Remote Display Consoles, J. Rodriguez, ESL Memorandum 9442-M-148/MAC-M-274, October 7, 1965, 4 pp.

246. The Design and Programming of a Display Interface System Integrating Multi-Access and Satellite Computers, D. Ross, R. Stotz, D. Thornhill, and C. Lang, ESL Memorandum 9442-M-190/MAC-M-353, July 6, 1967, 18 pp.

Time-Sharing Features

247. *AED Flash No. 11 - AEDBUG Usage, B. T. Fox, ESL Memorandum 9442-M-121/MAC-M-213, December 28, 1964, 20 pp.

248. *AED Flash No. 21 - Octal Debugging Subroutine LOPAT, J. Walsh, ESL Memorandum 9442-M-146/MAC-M-272, September 24, 1965, 4 pp.

249. *AED Flash No. 21-1 - Addition to Subroutine LOPAT, B. Wolman, ESL Memorandum 9442-M-146-1/MAC-M-272-1, July 20, 1966, 2 pp.

250. AEDBUG Program Description, B. T. Fox, ESL Memorandum 9442-M-142/MAC-M-260, August 12, 1965, 21 pp.

251. *AED Flash No. 25 - Loader/Unloader, B. Wolman, ESL Memorandum 9442-M-155/MAC-M-286, November 16, 1965, 12 pp.

252. *AED Flash No. 28 - Reduction of Load Time, B. Wolman, ESL Memorandum 9442-M-168/MAC-M-309, May 31, 1966, 3 pp.

253. *AED Flash No. 29 - Description of "LAED COMMAND", B. Wolman and C. Feldmann, ESL Memorandum 9442-M-169/MAC-M-311, June 10, 1966, 10 pp.

254. *AED Flash No. 29-1 - Change in the "LAED" Command, B. Wolman, ESL Memorandum 9442-M-169-1/MAC-M-311-1, August 3, 1966, 1 p.

255. *AED Flash No. 30 - Writing "MACHINE INDEPENDENT" AED-0 Programs, J. Rodriguez, B. Wolman, C. Feldmann, and D. Ross, ESL Memorandum 9442-M-171/MAC-M-315, June 21, 1966, 6 pp.

256. *AED Flash No. 32-1 - Tracing Subroutine STOPAT, B. Wolman, ESL Memorandum 9442-M-174-1/MAC-M-319-1, August 9, 1966, 5 pp.

AED Cooperative Program Progress Reports (all by D. T. Ross)

257. Progress Report No. 1 (March 1 - May 15, 1964), ESL Memorandum 9442-M-104/MAC-A-114, dated June 1964, 26 pp.

258. Progress Report No. 2 (May 16 - June 30, 1964), ESL Memorandum 9442-M-108/MAC-A-115, dated July 15, 1964, 15 pp.

259. Progress Report No. 3 (July 1-31, 1964), ESL Memorandum 9442-M-109/MAC-A-117, dated August 5, 1964, 14 pp.

260. Progress Report No. 4 (August 1 - October 31, 1964), ESL Memorandum 9442-M-111/MAC-A-121, dated November 12, 1964, 11 pp.

261. Progress Report No. 5 (November 1 - December 30, 1964), ESL Memorandum 9442-M-123/MAC-A-125, dated January 21, 1965, 23 pp.

262. Progress Report No. 6 (January 1 - February 28, 1965), ESL Memorandum 9442-M-128/MAC-A-130, dated March 10, 1965, 12 pp.

263. Progress Report No. 7 (March 1-31, 1965), ESL Memorandum 9442-M-130, dated April 9, 1965, 11 pp.

264. Progress Report No. 8 (April 1-30, 1965), ESL Memorandum 9442-M-133, dated May 1965, 9 pp.

265. Progress Report No. 9 (May 1 - July 31, 1965), ESL Memorandum 9442-M-141, dated July 30, 1965, 13 pp.

266. Progress Report No. 10 (August 1-31, 1965), ESL Memorandum 9442-M-144, dated September 21, 1965, 7 pp.

267. Progress Report No. 11 (September 1-30, 1965), ESL Memorandum 9442-M-151, dated October 15, 1965, 7 pp.

268. Progress Report No. 12 (October 1 - December 15, 1965), ESL Memorandum 9442-M-154, dated December 27, 1965, 14 pp.

269. Progress Report No. 13 (December 16, 1965 - February 15, 1966), ESL Memorandum 9442-M-161, dated March 7, 1966, 12 pp.

270. Progress Report No. 14 (February 16 - May 15, 1966), ESL Memorandum 9442-M-166, dated May 20, 1966, 20 pp.

271. Progress Report No. 15 (May 16 - July 31, 1966), ESL Memorandum 9442-M-175, dated August 5, 1966, 14 pp.

272. Progress Report No. 16 (August 1 - October 31, 1966), ESL Memorandum 9442-M-181, dated November 9, 1966, 23 pp.

273. Progress Report No. 17 (November 1, 1966 - March 31, 1967), ESL Memorandum 9442-M-188, dated April 3, 1967, 15 pp.

274. AED Progress Report No. 18 (April 1 - December 15, 1967), ESL Memorandum 70429-M-195, dated December 15, 1967, 14 pp.

APPENDIX III

FILM LOANS - ELECTRONIC SYSTEMS LABORATORY

The following films are available for two-week loans, free of charge:

1. KLUDGE. A 7-minute, black-and-white, silent, 16-mm film, 1966. Introductory shots show the Project MAC Computer and ESL Display Console. The main body of the film shows graphical drawing in two and three dimensions with the light pen, three-dimensional real-time rotation of figures and surfaces. A separate sequence at the end shows construction and rotation of complex molecular models.

2. CIRCAL-0. A 7-minute, black-and-white, silent, 16-mm film, 1966. This film shows how an on-line digital computer utility and an electronic circuit designer "cooperate" through a program that analyzes electronic networks. Man-machine interaction is accomplished through a graphical input-output terminal and through a teletypewriter terminal.

3. COMPUTER SKETCHPAD. A 30-minute, black-and-white, sound, 16-mm film, 1964. More detailed version of the film listed immediately following, including three-dimensional drawing. (Film of TV Program.)

4. SKETCHPAD. A 7-minute, black-and-white, sound, 16-mm film, 1962. Two-dimensional graphical language with constraints.

5. APT. A 30-minute, black-and-white, sound, 16-mm film, 1959. Film of an Education TV Program describing the Automatically Programmed Tool or APT System, developed by the M. I. T. Electronic Systems Laboratory with the U. S. Air Force supporting this work, in cooperation with the Aerospace Industries Association. The program describes the preparation of control tapes for numerically controlled machine tools through the use of English-like APT language for part programming.

6. A NUMERICALLY CONTROLLED MACHINE TOOL. A 20-minute, color, sound, 16-mm film, 1953. A lucid exposition of the basic principles of numerical control of machine tools. The film describes the first numerically controlled machine tool, developed in 1952 by the M. I. T. Servomechanisms Laboratory (now Electronic Systems Laboratory), under sponsorship of the U. S. Air Force. The film shows graphically the decoding of commands punched on paper tape into control signals which are applied to servomechanisms which drive the axes of machine motion. The basic steps of manual preparation of the input-control tape are also covered.

We request that all borrowers limit the loan period to two weeks, and that they mail films via air parcel post, special handling, each film insured for $50.00, addressed as follows:

Mass. Institute of Technology
Electronic Systems Laboratory
35-315
Cambridge, Mass. 02139
Attn: Librarian

PLEASE NOTE: Since our films are often tightly scheduled months in advance, many interested parties prefer to purchase their own copies of these films. Films 3 through 6 are available from the source listed below. Films 1 and 2 are not at present available for purchase.

CINE, Inc.
51 Kondozian Street
Watertown, Mass. 02172