



Theses and Dissertations

1985-04-01

Flexible Engineering Software: An Integrated Workstation Approach to Finite Element Analysis

Brant Arnold Ross
Brigham Young University - Provo

Follow this and additional works at: <https://scholarsarchive.byu.edu/etd>



Part of the [Mechanical Engineering Commons](#)

BYU ScholarsArchive Citation

Ross, Brant Arnold, "Flexible Engineering Software: An Integrated Workstation Approach to Finite Element Analysis" (1985). *Theses and Dissertations*. 3460.
<https://scholarsarchive.byu.edu/etd/3460>

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

FLEXIBLE ENGINEERING SOFTWARE: AN INTEGRATED WORKSTATION
APPROACH TO FINITE ELEMENT ANALYSIS

BRANT ARNOLD ROSS

FLEXIBLE ENGINEERING SOFTWARE: AN INTEGRATED WORKSTATION
APPROACH TO FINITE ELEMENT ANALYSIS

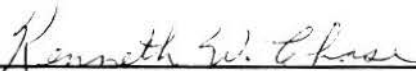
A Dissertation
Presented to the
Department of Mechanical Engineering
Brigham Young University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy


by
Brant Arnold Ross

April 1985


This Dissertation, by Brant Arnold Ross, is accepted in its present format by the Department of Mechanical Engineering of Brigham Young University as satisfying the dissertation requirement for the degree of Doctor of Philosophy.



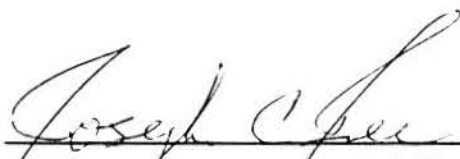
Kenneth W. Chase, Committee Chairman



Steven E. Benzley, Committee Member



Alan R. Parkinson, Committee Member



Joseph C. Free, Department Chairman

JANUARY 4, 1985
Date

ACKNOWLEDGEMENTS

When the author was laid-off from the Dubuque Works of John Deere on December 8, 1982, the choice was between finding another job, and going back to school. The support of his wife, Janna, who was willing to sacrifice for the third time for his education is greatly appreciated.

Thanks is given Dr. Kenneth Chase for arranging for a return to Brigham Young with three weeks notice, and for his support throughout this degree. Thanks are also due for the funding and direction provided by Dr. Steven Benzley, and encouragement from Dr. Alan Parkinson.

Funding was received from the Brigham Young University College of Engineering and Technology Research Chair in System Integration, held by Dr. Steven E. Benzley of the Civil Engineering Department. Funding provided by the Naval Weapons Center at China Lake under Contract N60530-84-M-ER77 was also helpful.

Hewlett-Packard provided various computer systems that have supported this research. More importantly, they have supported the graduate work of Terril N. Hurst, who has been a constant source of insight in the engineering design process. Wrestling over new concepts in engineering software with Terril has been a crucial part of this research.

Thanks are given to the Engineering Computer Services group at the Dubuque Works of John Deere. The experience and knowledge they shared was a helpful preparation for this research.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
LIST OF ILLUSTRATIONS	vi
Chapter	
1. INTRODUCTION	1
2. HISTORY OF ENGINEERING COMPUTING	10
Batch Processing	10
Interactive Systems	12
Computer Graphics	13
Distributed Systems	16
The Call for Integration	17
An Integrated Design Process	18
Databases - The Centralized Solution	21
Current Progress in Integration	23
3. TRANSFER OF FINITE ELEMENT DATA	25
Opening the Data Path	25
Neutral File Development	26
Producing the Neutral File	29
Reformatter Development	30
4. AN ENGINEERING WORKSTATION ENVIRONMENT	33
Workstation Environment	34
Data File Management	36
Industry Application	38

5.	DEVELOPING ENGINEERING SOFTWARE	41
6.	A SAMPLE PROBLEM	48
	Model Generation	50
	Neutral File Generation and Transfer	51
	The SAP IV Analysis	53
7.	DISCUSSION AND CONCLUSIONS	56
	FEA Data Transfer	56
	Engineering Workstation Environment	57
	Engineering Software Development	58
	Future Work	58
	REFERENCES	60

Appendix

A.	ROSETTA DOCUMENTATION	65
B.	DAVINCI DOCUMENTATION	94
C.	SQUIRE PROGRAMMING GUIDE	137
D.	CV FEM COMMAND SUMMARY	166

LIST OF ILLUSTRATIONS

Figure

1.1	Path of Finite Element Analysis Data.	5
2.1	Development of Engineering Computing	11
2.2	Robot Work Cell	15
2.3	A Typical Product Development Cycle	18
2.4	Engineering Organizational Grouping	20
2.5	Effect of Location on Communication.	20
3.1	Finite Element Relationships	26
3.2	Sample Portion of Neutral File from CV.	29
4.1	Davinci Software for Workstations	33
4.2	Davinci Commands	35
4.3	On-Line Documentation.	37
4.4	Software Demonstration.	37
4.5	Functional Diagram of Davinci Software.	39
5.1	Sample Multi-Level Prompts	46
6.1	Disc Drive Head.	49
6.2	Disc Drive Head Finite Element Mesh (Exploded View)	51
6.3	Oblique View of Torsional Mode.	54
6.4	Front View of Torsional Mode.	55

Table

3.1	FEA Neutral File Entities	27
3.2	Main Options of Rosetta	31
5.1	Software Development Process.	42
5.2	Sections of Software Documentation.	43

CHAPTER 1

INTRODUCTION

The computer has taken an increasing role in design, analysis, and manufacturing activities over the last fifteen years. Its ability to quickly calculate, store, and recall information may be used to improve the efficiency of engineering tasks. Computer methods have been successfully used in finite element modeling (FEM) of structures, simulation of dynamic systems, automated drafting, and automatic process planning, to name a few areas. Organizations now realize that they must efficiently use the computer in engineering product development to remain competitive [1].

Computer hardware has significantly improved in recent years. Engineering software remains as the bottleneck preventing better use of the computer in engineering. Past software development has demonstrated the computer's ability to efficiently perform specific engineering tasks. The current challenge is to properly integrate these capabilities such that engineering organizations may reduce their product development time [2].

A brief overview of the history of engineering software is given in Chapter 2. Past experience in software development and software use in engineering organizations suggest that the following points are important in the successful integration of software:

1. Data standards are needed for the smooth flow of data between tasks.
2. Computer users will tend to change from specialists to generalists as software becomes easier to use.
3. Better (more consistent) user interfaces are needed as the casual user works with a wider variety of software.
4. Flexibility must be built into software to allow each organization to adapt it to their product development process.

The objective of this research is to apply the above rules of integration to Finite Element Analysis (FEA) in a general way, such that this work may be extended to other areas. FEA is an ideal starting point for this research because: 1) it has existing data standards, 2) it is used by a wide variety of engineers, and 3) it is the focal point of engineering analysis in many organizations[3].

As software was developed to manipulate FEA data, a new software concepts for the engineering workstation environment evolved. Consider two creative men in engineering design, Edison and DaVinci. Edison was a specialist and conducted research with a laboratory fully staffed with specialists. DaVinci was a generalist and did most of his work by himself. Much of the existing engineering software requires a specialist because it is difficult to use. It is important for the user to be technically competent, but today many have become specialists because they have learned the "tricks" of using a particular piece of software on a particular computer. As software is improved it will become easier to use. The technically competent user will be able to use a wide set of

software tools for analysis and design. Users will tend to change from software specialists to software generalists. This work describes a new approach to establishing an engineering workstation environment for the engineering user of the future, who will use many software tools.

This report will describe the development of three major pieces of software. Rosetta.BYU accepts FEA data in a neutral format and produces input data files for various FEA programs (see Chapter 3). It allows the FE analyst to use data from mesh generators (CAD system or stand alone) more effectively. Currently, most mesh generators generate input data only for popular FEA codes, such as Nastran and Ansys. No pathway existed to programs developed in-house or less popular programs such as Nisa or Abaqus. Rosetta opens that path using two neutral formats: 1) the Navy neutral file format developed jointly by the author, Dr. Steven Benzley of Brigham Young University and representatives from various United States Naval Weapons Laboratories[4] and 2) finite element entities that are a subset of the IGES Version 2.0 standard[5].

The use of data standard standards reduces the number of paths between CAD systems and analysis programs (see Figure 1.1). Rosetta has the capability to input and output FEA data using the Navy neutral format and the IGES format. Input data sets may be created for the following FEA programs: Abaqus[6] and SAP IV[7], and graphics program Movie.BYU[8]. Support of additional programs requires an added subroutine to unload the data structure in the format of that program. The dynamically-allocated data structure provides the following:

1. Model checking to detect missing nodes and elements, and elements without material properties.
2. Complete editing for each of the 14 FEA data entities. Geometry may be input from a file and the rest of the model entered interactively.
3. Renumbering of entities to eliminate gaps in ID numbers, which are unacceptable for some FEM programs. Gaps are sometimes left in the model by the mesh generator.
4. Reordering of the data set to make it more understandable.

A library of finite elements used by Nastran, Abaqus and SAP IV is used by Rosetta to convert finite element models from one data standard to another. The library indicates the IGES topology type of each element and the relationship between the local node numbers of the element and the IGES element.

Davinci.BYU helps the analyst classify software and data files (see Chapter 4). The analyst can easily organize a library of programs into menus interactively. For each program, Davinci will store:

1. The program execution instruction.
2. A brief description of the program.
3. Location of manual and support people.
4. Location of the source file for the program.

Data files may also be classified using up to six user-defined categories, as well as by size, time of creation and name. Davinci

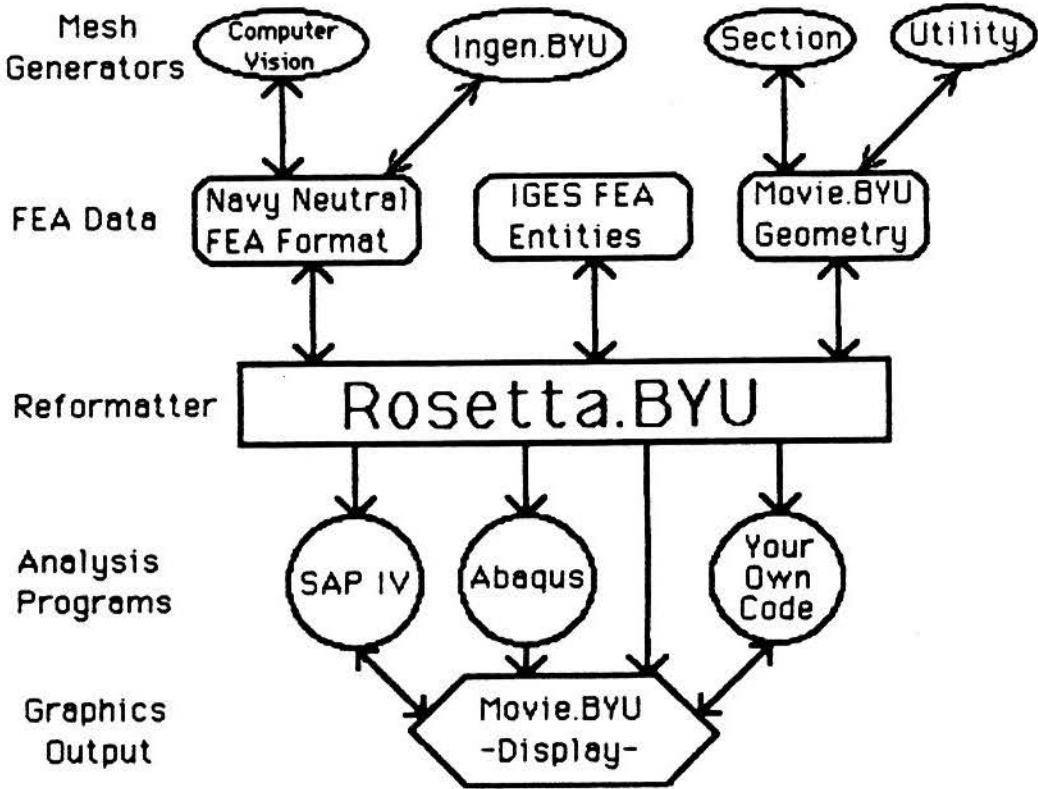


Figure 1.1 Path of Finite Element Analysis Data

automatically prompts for information on new data files as they are created. It can store information on backed-up files. A few suggestions for categories that might be used to organize data files follow:

1. By program name - linking each file to the program it is used with: Nastran, Visicalc, etc.
2. By project - linking each file to the appropriate project number, name, or charge code.
3. By product - using a product number or name.
4. By customer - for users who provide analysis services.

Two subsets of the Davinci program were created for use in multi-user environments (mainframes, shared minicomputers, etc.). In this role Davinci would manage software shared by a group of computer users. Data file management features are removed from both versions, and edit features appear in the system manager's version (Sentinel) but are removed from the user's version (Gateway). Davinci helps users share software and aids communication of software status.

Squire.BYU is a library of routines that provides a consistent, high-quality interface for interactive Fortran programs (see Chapter 5). Input, editing, option selection, file handling, and various utility functions are included. Users benefit when this library is used with a set of software, because input and editing procedures are consistent from one program to another.

Squire routines adapt input to both new and experienced users, by providing three levels of prompts for each input. The first level is terse for the expert user. The second level is a fairly complete description and the third level suggests a response. If the user can't answer a question from the first level prompt, the second and third levels can be seen by hitting the return key without giving a response. Multiple responses may be given at one time, by placing them separated on one line with a space or comma. This allows the expert user to answer ahead and avoid prompts.

A set of general subroutines may be used to edit real, integer and text items. The Squire routines take care of the editing details while allowing the programmer the flexibility of formatting the data for each specific application. The Squire.BYU library is used by Rosetta.BYU and Davinci.BYU.

One unique aspect of the three above pieces of software is their portability [9]. As they were developed they were periodically compiled and run on a variety of computers: an HP9000/HP-UX, VAX/VMS, VAX/Berkeley 4.2 Unix and IBM 4341/VM. One version of each program runs on all computers. The few system dependent statements are clearly marked, with statements included for each machine. Unused statements are commented out. Moving the program to a different computer then requires only that a different set of statements be used.

This software was used to perform a modal analysis of a disk drive component (see Chapter 6). The sample problem required the following:

1. The basic geometry of the component was created on a ComputerVision CADDS4 system. A mesh was generated.
2. The FEA data was put into a neutral format using the PUTFEM software developed at China Lake Naval Weapons Center.
3. The neutral file was transferred by tape from the Computer Vision to a VAX/VMS system, then to an HP9000/520.
4. Using the HP9000/520, the neutral file was processed by Rosetta and a SAP IV input file was produced of the FEA model.
5. Analysis specification cards were added to the SAP IV input file and a mode shape analysis was performed on the HP9000/520. SAP IV has been modified to directly produce output files compatible to the Movie.BYU graphics package.

6. Movie.BYU data files were transferred to a VAX/VMS system. Color continuous pictures were generated and photographed for each mode shape with Movie.BYU.
7. A 16mm movie was made of several animated mode shapes. Some 360 frames were shot for each 30-second animation. The animation feature of Movie was used to produce the frames.

The 9000 software was organized and managed using the Davinci program. Davinci also contained information to help the user understand the software, in the form of on-line documentation and automatic demonstration of software capabilities.

In summary, the original contributions of this work are all related to development of a unique engineering workstation environment. This environment is designed to take advantage of anticipated advances in computer hardware and data standards. The following original contributions are implemented (see also Chapter 7):

1. Development of an easy-to-understand FEA data standard in conjunction with the Navy.
2. Development of a library of finite element information used to convert data from one standard to another.
3. Use of FEA data standards to link FEA analysis with pre- and post-processing.
4. Solution of the problem of linking custom in-house codes to standard CAD/CAM mesh generators.

5. Creation of a Fortran library of routines to provide a consistent user environment and reduce programming efforts.
6. Use of classification techniques to organize programs and data files as the engineer pursues a wider range of techniques.

Efforts to improve the engineering design process with the computer must focus on interaction between existing software. It is not enough to perfect each step in the process, rather the process as a whole must be improved. This work has produced software tools to improve the finite element modeling and analysis process. More work is needed to extend these concepts over a broader portion of the design process. Data standards must continue to improve and spread to more technical areas.

CHAPTER 2

HISTORY OF ENGINEERING COMPUTING

This chapter provides an overview of the history of engineering computing, starting in the late nineteen sixties. Understanding technical computing development is important because it explains some current practices. Change is always met with some inertia. Understanding past techniques makes it easier to develop compatible new methods.

Batch Processing

In the early nineteen sixties, most engineering companies used computers in a batch mode. Batch processing requires that input data (programs, data, instructions) be stored on some physical medium (keypunch cards, paper tape, etc.) to be processed by the computer in one stream. A unique set of output resulted from any set of input data. All instructions were fixed before processing, no intermediate adjustments were possible. The engineer's only contact with the computer was a card reader and printer. A typical computer job was to run a set of cards through the card reader in the morning and return in the afternoon to search through a pile of computer output. The problems with batch processing may be categorized into four major areas (see also Figure 2.1):

1. Data stored on keypunch cards or paper tape was hard to review; consequently it was difficult to find errors. Once errors were detected, a new keypunch card or paper tape was needed. Care was needed to correct existing errors without adding new ones.
2. The delay in receiving output that was characteristic of batch processing made it difficult for the engineer to maintain a train of thought in his computer work. The engineer could only submit a few sets of input each day because of the excessive delays.
3. Communication with other users was poor. The only way to share input data or programs was to physically exchange the input media. Those who worked in different locations found this very difficult. Poor communication discouraged the development of standards to facilitate data transfer.
4. The user had no control over response time. Operating procedures were handed down to the engineer.

BATCH

Problems:

- Hard to Edit
- Delay in Feedback
- Poor Communication with other users
- Little local control

GRAPHICS

Improvements:

- Analysis output is clear. User can do "what if"

INTERACTIVE

Improvements:

- Full-Screen Edit
- Quick Feedback
- Access to files from other users

DISTRIBUTED NETWORKS

Improvements:

- Local control
- Problems:
- Poor communication with other users

Figure 2.1 Development of Engineering Computing.

Interactive Systems

In the middle nineteen seventies, interactive computer systems were introduced. Key punch card and paper tape machines were replaced with terminals with cathode-ray tube (CRT) screens. An entire screen of data could be modified by moving a cursor about using arrow keys, and typing over errors. This made program and data modification easy. It was easier to check for errors and harder to inadvertently tamper with good data.

Computer response was better in interactive systems. The user could make corrections during program execution and run as many processes as he liked. Response was quick enough to allow the engineer to modify data and retry an analysis to test the sensitivity of his design parameters. Also, programs could be written to adaptively prompt for information based on earlier input [10].

With interactive systems, each user controlled a portion of the disk space. It was easy to share information with other users and was a good environment for user interaction. However, little effort was made to set standards and pursue integration. Why? Because it was a time of specialization, and the specialists were not interested in interacting with others. The attitude of the specialist is described by Toffler [11]:

... specialization was accompanied by a rising tide of professionalization. Whenever the opportunity arose for some group of specialists to monopolize esoteric knowledge and keep newcomers out of their field, professions emerged.

The sophisticated analysis and modeling that became possible with the computer produced many different specialities. New capabilities

to solve a large number of equations were used by some to analyze structures using finite element techniques [3], by others to study the kinematics and dynamics of mechanisms[12], and still others to analyze vibrations [13]. Seireg [14] comments on the development of specialities within the product design and manufacturing cycle:

It is only natural that in the formative stage a technology develops along specialized lines to meet specific needs. Factory automation is no exception. Manufacturing encompasses business as well as several branches of engineering. These long established disciplines have articulated differing philosophies, methods, and means of communication. Marketing, finance, design, analysis, drafting, production planning, value engineering, fabrication, materials handling, process control, assembly, quality assurance, purchasing, inventory control, warehousing, human resource management- the professionals in each of these tasks have begun to use computer hardware, languages, and methodologies that are designed for their particular needs but that may not interface ...

The pioneers who developed new computer tools for engineering devoted their resources to their own specialty. In general, they could not justify the effort needed to interface their work with others. Some attempts were made to combine engineering software from different specialities, but resulted in very specific tools that were not widely accepted or used [15].

Computer Graphics

Soon after interactive systems became popular, computer graphics found wide use in engineering. First, two-dimensional plots were

used. Later, graphics were used to actually depict parts and assemblies. Added capabilities such as hidden-line/surface removal, shading and continuous color-tone output produced realistic images [7].

Computer graphics helped users comprehend and interpret output quickly. The engineer was able to generate plots, make conclusions on product performance, adjust variables, and generate new plots for an improved product, all in one session with the computer. Graphical output was better able to communicate the results of engineering analysis to those who were not specialists [16] (management, specialists in other areas, less-skilled personnel). Figure 2.2 shows a sample graphical output with hidden line removal.

After graphical output became accepted, special computer hardware was developed: graphics tablets, light pens, mouses (mice?), and track balls [17]. Graphical input simplified the creation of geometric data and allowed user commands to be specified by pointing. The combination of good interactive programming techniques and computer graphics resulted in computer software that could be used by non-specialists [3], because it was easy to input data and easy to understand the results.

The use of computer graphics motivated integration efforts on a case-by-case basis. Analysis programs were modified to output data in a compatible format with general-purpose graphics programs [18]. There was no standard for graphics data, rather each graphics program had its own data format.

The proliferation of graphics software was matched by a proliferation of graphics hardware. Initially, vector storage tubes dominated the graphics terminal market. Later, as computer memory chips

became faster and less expensive, raster displays became the accepted standard. As terminals became more sophisticated, functions that were originally performed in software became hardware functions. Examples include: drawing of graphics primitives such as circles and arcs; rotation and translation of entities; and solid fill of polygons.

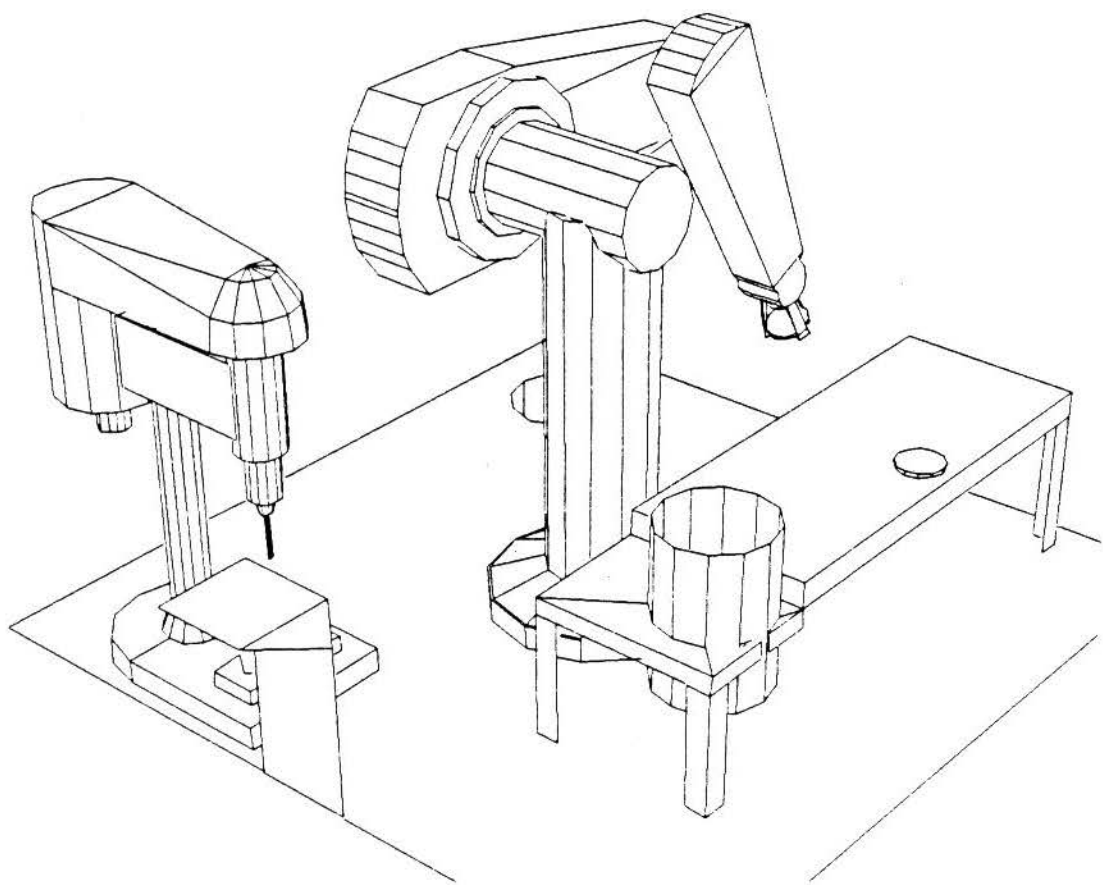


Figure 2.2 Robot Work Cell

Each vendor developed his own protocol for using these functions. Hardware functions reduced the input needed to produce a picture, but made it difficult to write a program to interface with terminals from

different vendors. A standard was needed to simplify the use of different terminals. One standard has been adopted and four others are under consideration by the American National Standards Institute (ANSI) [19]. These standards are mostly used by graphics programmers to write device drivers, and do not receive much publicity. However, the graphics standard is one of the first standards to be accepted in computer software. The development of Fortran-77 to standardize file manipulation and the use of character strings is a similar process. These standards primarily effected programmers. The computer user was not yet influenced by standards for input or output.

Distributed Systems

The early nineteen eighties saw a declining interest in large, centralized computers as smaller, local computers became popular. Super-minicomputers such as Digital Equipment Corporation's VAX became widely used in engineering analysis. The VAX generally provided a better environment for interactive programs than a large mainframe. A group of ten to fifty professionals was needed to economically justify a VAX. More recently 16- and 32-bit microprocessor based workstations offer nearly the computing power of a VAX for a single user [20,21] at a lower cost. They are small enough to sit on a desk and can be justified by the needs of two or three professionals. Within a few years, workstations costs are projected to drop to the point where each engineer will have one.

At the same time, small computer systems that specialized in design functions were developed using the latest graphics hardware. These CAD systems were sold as a combination of hardware and software.

They were referred to as "turnkey" systems, since their software made them ready to use as delivered (just "turn the key"). The first systems were based on 16-bit microprocessors and were used for two-dimensional drafting. Later systems could manipulate full three-dimensional models and used super-minicomputer hardware to improve response. Finite element mesh and numerical control (NC) tape generation were added to the basic drafting capabilities.

The Call for Integration

The progression from batch processing to current workstations and CAD system technology caused an abrupt change in thinking upon the introduction of CAD systems. Until then, only technical specialists had been involved with computers in engineering. With CAD systems, designers and draftspersons with forty to eighty hours of training became heavy computer users. Supervisors who had been bewildered by computers in the past now operated their own CAD systems. These non-specialists soon became aware of the capabilities of the computer and desired to share resources with the isolationists, the specialists. The new users found that it was difficult to interact with the specialists' resources.

Most different pieces of software would not work together without custom interfaces, data was never in the "right" format (no standards), and nobody could see the big picture of computer utilization in the design and manufacturing process. That is why Schaeffer [22] said:

The current state of computer-aided engineering can be represented as a set of unconnected processes. By themselves, the individual areas such as finite element analysis, geometric modeling, and drafting adequately

satisfy the local needs of engineering departments. But these hardware and software bits and pieces communicate data poorly between one another. In effect, the areas exist merely as "Islands of Automation." As a result, many efforts in a CAE system are redundant, information transfer is slow and often done on paper instead of electronically, and productivity is not as high as it could be.

An Integrated Design Process

Many people could understand how the sharing of product definition data throughout a company could improve productivity. Far fewer were prepared to take the steps needed to integrate. A review of the typical design process (see Figure 2.3) will illustrate some of the activities that need to be integrated. The three boxes in the top half of the figure represent design engineering activities. In the investigation stage various concepts are explored, a concept is selected, and technical specifications for the product are set. In the next stage an initial prototype is made that hopefully meets performance specifications with the selected concept. A

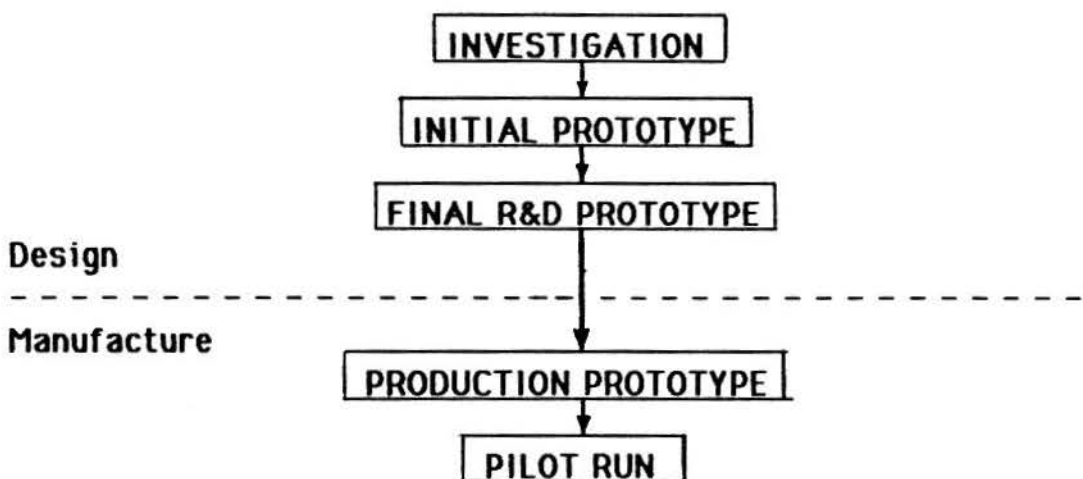


Figure 2.3 - A Typical Product Development Sequence

final research and development (R&D) prototype should demonstrate product feasibility, meeting economic requirements and using current technology.

The two boxes in the bottom half of the figure represent manufacturing engineering activities. A production prototype is built as manufacturing processes are selected and refined. The pilot run tests the manufacturing process in a high volume environment.

In the optimal situation, data would easily move by computer, not only from one step to the next, but freely from one end of the design cycle to the other. For example, Potter[23] expresses a manufacturing viewpoint.

While most CAE systems have produced a savings in two or three years, the most important benefit lies in improving productivity by bringing the engineering process closer to the factory floor. . . .

Manufacturing is coming full circle. When the first manufacturing was done in a cottage industry, the designer was also the manufacturer, conceiving and then fabricating products one at a time. Eventually interchangeability of parts was conceived, production was separated into specialized functions, and thousands of identical parts were produced at one time.

Today, production is much more batch oriented, and although the designer and manufacturer will probably not become one again, the functions are being drawn closer in the movement towards an integrated manufacturing system. Ironically, while the market demands a high degree of product diversification, the need for increased productivity and reduced costs is driving us towards more coherent systems.

Figure 2.4 depicts a typical organizational grouping for design and manufacturing functions in a large (over 1000 employee) company. The dotted line indicates the separation between design and manufacturing. Each group is composed of specialists and has limited interaction with the

other groups. Design and manufacturing groups seldom spend much time together. In many companies their offices are in different buildings. Thomas Allen of MIT (as cited in the book, *In Search of Excellence*) studied the effect of physical separation. His results [24] are shown in Figure 2.5. The physical distance that separates design and manufacturing activities may cause problems. However, appropriate use of the computer may help. A Hewlett-Packard experience provides a good example.

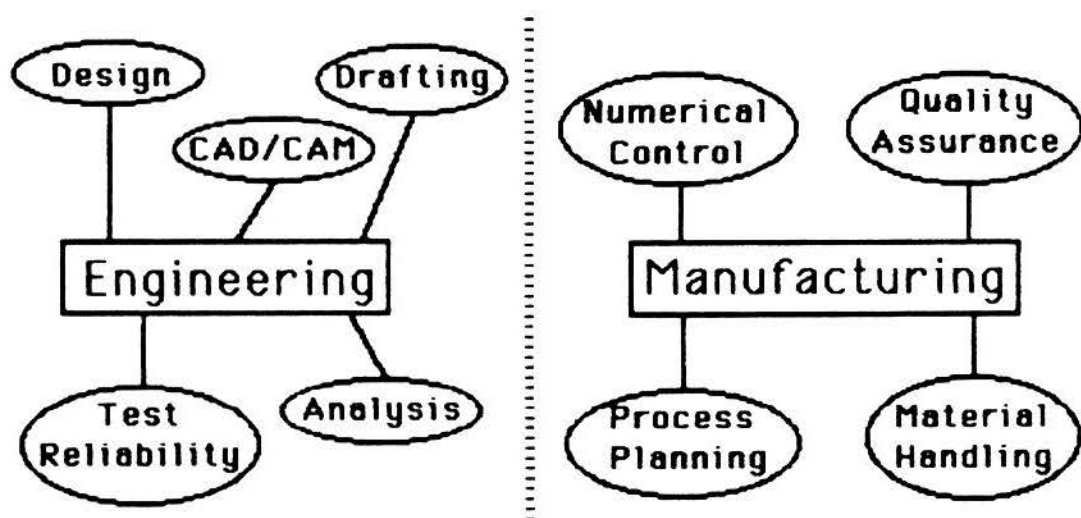


Figure 2.4 Engineering Organizational Grouping

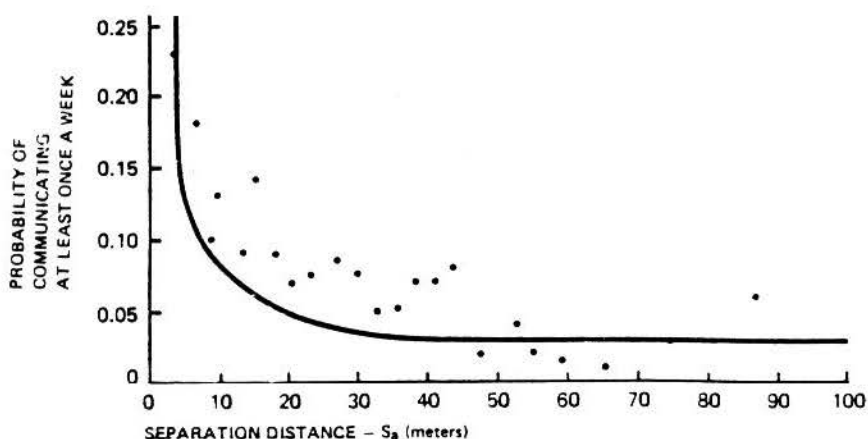


Figure 2.5 Effect of Location on Communication

Hewlett-Packard is primarily an electrical engineering company. Some felt that mechanical engineers, who make up a small minority, did not have the opportunity to interact with their peers. Seven mechanical engineers, each representing a different HP division, were given an account on a computer information network. The network promoted communication between the mechanical engineers, even though they were scattered about the Western United States. They came to a consensus on various topics and have made an impact on several corporate decisions, as a handful of people among over seventy thousand employees. Electronic mail can bring people together who are physically far apart [25]. Much more will be possible when engineering data is shared as easily as mail.

Databases – The Centralized Solution

Hardware is already available that allows computers to talk to each other. Proprietary networks allow many heterogeneous systems to share data. Local area networks are becoming popular to tie personal computers together. Relatively little has been done to develop software to move, store, and relate product data.

Dube and Johnson [26] outlined five basic points to consider in developing a computer-assisted engineering data base:

1. Data management techniques should eliminate redundant data. The geometric, process, classification data, etc. should be stored only once.
2. The data integrity of the product must be maintained. Data should be protected from theft and destruction by accidental

operator error, malicious acts, or hardware failure. Finkle [27] covers this topic in some detail. This is a difficult problem because various groups need access to product data during the product design cycle. The privilege of changing data transfers from one group to another as the cycle progresses. Felippa [28] proposes that each group should have a local data base with relatively free access. Interaction between local and global data bases would be carefully controlled.

3. Users should be able to access data on two levels: by interactive query and through a standard programming language (such as Fortran and other high-level languages).
4. Data should be separated into different subgroups (views, layers, etc.) to allow the user to selectively extract data.
5. Data should be independent of the application. This implies that a data definition interfaces the data and any application. The data definition would specify the structure and form of the data.

Navathe [29] mentions another very important point, relatability, which is the ability to show relationships between different pieces of data. The usefulness of a data base in engineering depends upon the richness of the relationships that are maintained. The database format and organization are called a schema [30]. Development of a comprehensive schema for engineering design data is very difficult. Many man-years of effort were spent developing schemas for CAD systems, even though they "only" treated graphical data.

The Initial Graphics Exchange Standard (IGES) [5] was developed to represent geometric, topological, and non-geometric product definition data. It contains a wide variety of entities, such as: lines, arcs, surfaces, planes, splines, ruled surfaces, etc. Even with its capabilities, much work is still needed before a complete product definition could be stored. For example, a tolerance specification is represented as a text entity. It needs to store the relationship between the tolerance and its associated features so that the data may be used by a tolerance analysis program. A wide variety of other relationships would be needed for use by analysis programs currently in use for product design.

CURRENT PROGRESS IN INTEGRATION

Little software is available that uses IGES other than turnkey CAD systems. Systems that claim compatibility to share data throughout design, analysis, and manufacturing functions are primarily proprietary and dependent on a specific collection of hardware [31]. The systems provide interfaces to popular analysis codes such as Nastran and Ansys, but rarely provide for data transfer to in-house codes.

It is not surprising that many analysts dislike CAD/CAM integration efforts. They do not want any association with the drafting function. They are offended when a CAD/CAM system forces them to give up analysis software that they have used for years, because it is not "supported." Better results may be obtained with internal projects to integrate in-house analysis software with turnkey CAD systems, as demonstrated by Xerox [32].

However, most companies cannot afford to expend the resources

needed to create their own integration software. The Xerox success reflected a major effort on the part of fifteen staff members, yet merely demonstrated the feasibility of integrating CAD and CAM. It is much more difficult to create techniques that will solve the integration problem for general engineering. A more dramatic example is provided by the Boeing Corporation, with a 250 million dollar computer budget per year. They created their own data base and began to write translators to their CAD systems and engineering analysis programs. The effort overwhelmed them, budget notwithstanding. They decided to abandon their effort and support national standards such as IGES.

Kennicott [33], who coauthored the IGES standard and continues engineering database development full-time at General Electric, admits that a complete product definition database is not yet available.

This document describes software that big or small companies may use to move FEA data from one system to another. A computer environment is developed for the new type of user who will operate an engineering workstation in an integrated environment. The complete integration solution is not offered here, but a start.

CHAPTER 3

TRANSFER OF FINITE ELEMENT DATA

Opening the Data Path

Finite element analysis (FEA) is data intensive. Development of pre- and postprocessors to manage large amounts of data was needed before this tool could be used extensively by industry. Powerful mesh generators are available to help the engineer create finite element models of complex structures. Mesh generators may be based on solid or wire-frame geometry models, and may run on a specific CAD system or on a general-purpose computer. A need common to all mesh generators is the ability to transfer model data to both "custom" and "standard" finite element analysis codes.

Most mesh generators can create input data sets for popular FEA codes, such as Nastran, Ansys, SAP, etc. The user may also need to transfer data to specialized in-house FEA codes. Engineers could be more efficient if finite element software supported both custom and standard FEA programs.

One solution would be to write translator software to convert information from an input data set of a "standard" FEA code into a data set for a specialized code. A disadvantage of this method is that information not present in the standard input data would not be available for the specialized code. An in-house code may require much custom data.

A more general solution would be to output all finite element modeling information in a "neutral" format, a format that is not specific to any FEA program. Separate software would be developed to reformat the neutral data for specialized analysis codes. This approach requires a neutral format that could meet the needs of a variety of FEA programs. A neutral file acts as a database that contains finite element information for a particular model. FEA database relationships are shown in Figure 3.1.

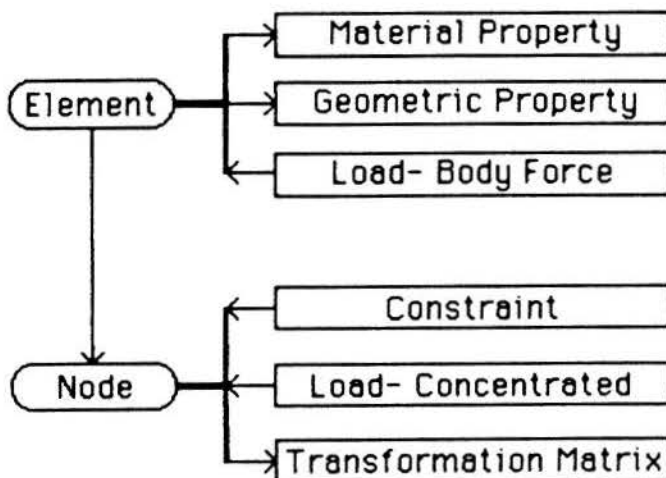


Figure 3.1 Finite Element Relationships

Neutral File Development

The Naval Weapons Center Computer-Aided Engineering Support Office (CAESO) at China Lake began development of a neutral file in January 1983 [31] to meet their need to extract data from the ComputerVision (CV) CADDs 4 system and create input data sets for Abaqus [6], an FEA program that is not supported by the CADDs 4 mesh generator. Software to output the prototype neutral file and a prototype reformatter were also developed

by CAESO. In April 1984 a meeting was held at Brigham Young University (BYU) with representatives from various Navy laboratories to define the first release of the neutral file [4]. One objective of the BYU meeting was to generalize the neutral file and reduce bias towards the CV data base. It was decided that the neutral file should be a free format text file, as explained below.

Each entity (node, element, etc.) begins on a new record (80 characters long) and is identified by a four-character keyword. An identification (ID) number follows the keyword. The entities, their keywords, and content are summarized in Table 3.1 (A detailed explanation may be found in reference 4, enclosure 4). Data fields are separated by commas, and each record is terminated with a semicolon (See Figure 3.2). This format has the following advantages:

Table 3.1 - FEA Neutral File Entities

<u>Type</u>	<u>Key</u>	<u>Content</u>
Header	HEAD	Comment or title card
Node	NODE	Transf. matrix #, coordinates, scalar
Element	ELEM	Name, mat'l #, geom prop #, node #'s
Material Property	MATL	Name, isotropic condition, reference type, reference value, property values
Geometric Property	PROP	Name, property values
Load, Element	ELOD	Element #'s
Load, Nodal	NLOD	Node #'s
Load, Harmonic	HARM	DOF, displacement value, node #'s
Constraint, Perm	PCON	DOF(s), displacement value, node #'s
Constraint, Multiple.	MCON	Dependent, independent node #'s, ratios
Constraint, Omitted DOF	ODOF	Degree of freedom omitted, node #'s
Constraint, Retained DOF	RDOF	Degree of freedom retained, node #'s
Transformation Matrix	TRAN	Coordinate system type, matrix values

1. Neutral file is human-readable.
2. Changes may be made by hand using a standard text editor.
3. Information is not column dependent.
4. Each record may occupy as many lines in the file as needed.

While node and element records have specific data assigned to each field, the property cards contain an arbitrary set of values. A general material property record is difficult to define because of the wide disparity between material models in different FEA codes. Much data could be required to handle the range of FEA problems from elastic-plastic models to thermal stress calculations. The neutral format stores a set of real numbers in an arbitrary order. The values are associated with specific material property values (poisson's ratio, etc.) when an FEA input file is created.

The element and nodal load records consist solely of a group of elements or nodes, as shown in the Content section of Table 3.1. Additional information is usually needed to complete the load information, depending on the type of analysis (linear or nonlinear, static or dynamic, constant or variable load history, etc.). Like the property records, additional information is entered by the user to complete the load information when the neutral data is reformatted to prepare an input set. As more experience is gained with the neutral format, more information may be added to the load cards.

```

NODE, 127,0,-.18370E2,0.,.10500E1,0.;
NODE, 273,0,-.16214E2,-,77640E0,.97686E0,0.;
ELEM, 83,QUAD,0,0,260,259,112,113;
ELEM, 1,QUAD,0,0,148,273,126,127;
ELEM, 2,QUAD,0,0,169,189,273,148;
NODE, 275,0,.39994E1,-.37848E0,.29113E0,0.;
NODE, 108,0,.49734E1,0.,.25809E0,0.;
MATL, 3,STEEL,ISOTROPIC,TEMP,100.,.37058E7,.14028E7,
      0.,.37058E7,0.,.14028E7,.00772;

```

Figure 3.2 Sample Portion of Neutral File from CV

Producing the Neutral File

Writers of computer-aided design and mesh generator software are often reluctant to release information on their database structure (considered proprietary). However, they generally offer a way to extract data indirectly from the database. One method is to output a general-purpose FEA data file (used by Patran [35]). This method has the limitation that some FEA data may not be present in the general-purpose file. The second method is to provide on-line access to the database through vendor-supplied subroutines (used by CV). However, the second method requires more software development to gain the flexibility of accessing the database.

The Navy software uses CV subroutines to cycle through the database to output all FEA data from the CADDS 4 database. The resulting neutral file is an unordered mixture of entities, as shown in Figure 3.2. Neutral file records are not order-dependent, since each record contains a keyword which identifies its type.

Reformatter Development

Once the neutral file is extracted from the mesh generator, the next step is to reformat the data into an input file for a target FEA program. A reformatter code, Rosetta.BYU, was developed at Brigham Young University with an emphasis on portability and flexibility (documentation is found in Appendix A). Portability results from writing structured code using ANSI Fortran-77 [36]. Flexibility is reflected by the support of three data standards: IGES Version 2.0 [5], Movie.BYU [8], and the neutral file (See Figure 3.3). The main options of Rosetta are summarized in Table 3.2.

The IGES Version 2.0 format includes node and element entities, but not loads, constraints, or properties. The IGES FEM Subcommittee has suggested formats for the missing entities through "Request For Change" documents [36]. Rosetta uses the suggested IGES format for nodes and elements.

Movie.BYU uses either a polygonal or solid representation of objects. Both formats may be translated to an equivalent finite element mesh. Support of Movie.BYU data allows mesh data in neutral or IGES format to be converted to Movie format for display and verification. Movie utilities may also be used to create mesh geometry.

A versatile data structure is the most important part of Rosetta. Various data anomalies were considered as data structure concepts were developed. One problem is that the records may not be input in order (as shown in Figure 3.2), but FEA programs often require ordered data in their input files. Another problem is that gaps may occur in entity numbering as the result of mesh editing. FEA programs may not accept gaps in the entity numbers.

Table 3.2 Main Options of Rosetta

Select process option (c,e,i,l,s,w,q):

- c - Check for model completeness
 - b - optimize Bandwidth through node renumbering
 - m - check for Missing items
 - e - list Elements without material property
 - n - list Nodes not used
 - q - quit model check option
- e - Edit the current model data
 - a - Add/delete items to node/element lists
 - h - Header information
 - n - Node data
 - e - Element data
 - c - Constraint data (p,m,o,r)
 - p - Property data (m,g)
 - l - Load data (n,e)
 - t - Transformation matrix data
 - q - quit list/edit option
- i - read in a different Input file
 - i - IGES version 2.0
 - n - Navy neutral file, version 1.0
 - m - Movie.byu geometry file
 - q - Quit execution of Rosetta
- l - List the current model data
 - a - list All model data
 - n - Nodes
 - e - Elements
 - m - Material properties
 - g - Geometric properties
 - l - Loads
 - c - Constraints
 - q - Quit list option
- s - give Status of data storage
- w - Write out the model to a file
 - a - Abaqus input data file
 - i - Iges version 2.0 format
 - m - Movie.byu polygonal format
 - n - Navy neutral file version 1.0
 - s - Sap iv input data file
 - q - Quit output option
- q - Quit execution of Rosetta

Third, the relationship between quantities of one entity vs. another varies widely, depending on the application. For example, a complex, machined part might contain thousands of nodes and elements and a single material property, while a cast part might have a different material property for each element (properties dependent on cooling curve). The data structure should adapt to either extreme. A custom data structure has been developed for Rosetta to handle these problems and is described in Appendix A.

Rosetta will support input and output of an arbitrary number of data formats through subroutines that load or unload records from data files. Output subroutines not only reformat existing data, but also allow the user to add specific information for a FEA program. A general purpose code like Abaqus may have a dozen different material property cards that may be used, depending on the type of problem. The Abaqus output subroutine allows the user to link values carried in the material property record of the neutral file to the appropriate material property card. In this way the neutral file stores only values needed for the particular problem, and the user has the flexibility to use the values as needed.

CHAPTER 4

AN ENGINEERING WORKSTATION ENVIRONMENT

An engineering workstation is a stand-alone computer system that gives the engineer personal control of the data processing power of a dedicated minicomputer. Computer software aids have been developed for engineering workstations in design and analysis, as well as general office automation [37,38,39,40,41]. With the right workstation management tools, the engineer's computing environment can be optimized to best use available hardware and software. The engineer should be able to use software for data and file management, communication, and analysis without being a computer expert. The ideal environment uses software that was purchased or developed in-house. Davinci.BYU software (see Appendix B) has features of the ideal engineering environment (Figure 4.1).

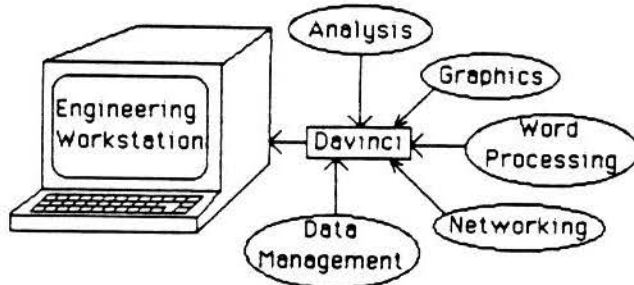


Figure 4.1 Davinci Software for Workstations

In the past, an engineer spent a great deal of time learning how to use each program. Improved user interfaces will result in less emphasis on computer literacy and more emphasis on technical proficiency. In the future, the average engineer will use a wider variety of software and become more of a generalist. The contrast between the general and specialized approach is illustrated by two famous men, Leonardo Da Vinci and Thomas Edison. Da Vinci was a "renaissance man" who worked independently and found success in integrating his profound knowledge of diverse fields (painting, sculpture, anatomy, hydraulics, aeronautics). Edison, on the other hand, obtained success by training an army of specialists to perform specific subsets of a project. Workstations allow individual control of resources for more convenient data processing and make possible increased integration of applications software. The benefits of both the Da Vinci and Edison approach may be obtained by the proper use of a well-managed workstation.

WORKSTATION ENVIRONMENT

The author assumes a workstation with the data processing capabilities of a VAX-11/750, 50+ megabytes of mass storage, fully supported ANSI Fortran-77, and multi-tasking. Multi-tasking is the ability to call or initiate one process from another. When the second process is terminated, control returns to the initiating process. Multi-tasking is not part of ANSI Fortran-77, but is a library function callable from Fortran under the VMS and Unix operating systems [42,43].

Davinci.BYU lets the engineer organize software by setting up menus which classify programs by their function or application. When a

category, such as "Finite Element Analysis," is selected from the menu, Davinci displays the corresponding sub-menu. The sub-menu may contain a list of more detailed sub-categories, such as "Mesh Generators", "Post Processers", etc., or it may contain a list of the available computer programs. If the user wishes to run a program, Davinci will execute it as a secondary process, using the execution instruction previously stored in Davinci's database. A set of user commands permits the engineer to move quickly through the menus and execute programs or procedures. Using the Davinci edit commands, the engineer can set up a software library by creating new menus, adding new programs, and changing or deleting any menu item. A summary of the Davinci commands is listed in Figure 4.2.

Select DAVINCI option: (m,i,x,e,#,q)

- | | | | |
|--------|--------------------------------|----|-------------------------------|
| m- | move through menus | e- | Edit menus, file data |
| #- | move to menu number (#) | m- | edit Menu |
| u- | move Up one menu level | f- | edit a single data file |
| t- | move to Top of menu structure | c- | edit data file Categories |
| f- | find menu containing (name) | l- | edit the Library title |
| q- | Quit, return to main options | r- | Restore a program on-line |
| | | o- | put a program Off-line |
| i- | Information on menu items | q- | Quit edit option |
| s- | give Synopsis of menu item (#) | #- | select item number |
| r- | show References for item (#) | | execute program or moves to # |
| m- | display Menu structure | v- | Verify data file info |
| f- | list data File information | q- | Quit DAVINCI program |
| x- | eXecute a program | | |
| ITEM # | - number of program to execute | | |
| (name) | - program code name | | |
| s- | enter your own System command | | |
| q- | Quit execute option | | |

Figure 4.2 Davinci Commands

In addition to the execution instruction (up to 75 characters), other information may be stored for each program. A brief description of up to ten lines (75 characters per line) may be used to describe the purpose and capabilities of the program. Text containing references to program documentation and persons to contact for assistance may also be stored in the same manner. A keyword of up to nine characters is associated with each program. The engineer may use a keyword to execute a program directly, without moving through the menus, or to quickly move to the menu containing the corresponding program. The keyword is also used to detect the use of the same program in different menus, in which case both menus use the same program information.

The execution command is not limited to executing software, but may also execute any system-level command. The user may choose to edit or list a data file. On-line documentation may be included in Davinci by setting up a menu of programs, each listing a file that contains a portion of a user's manual. Figure 4.3 illustrates the use of Davinci to retrieve on-line documentation for the Movie.BYU graphics package [8]. Automatic demonstrations can be added by running a program with input redirected from a file. Redirection is a standard feature of Unix [44], and is also available in the Squire.BYU library input routines (see Chapter 5). Figure 4.4 shows an automatic demonstration of analysis and graphics output.

Data File Management

A large number of data files will result from the many programs used on a workstation. Davinci contains a data file management feature that organizes the user's files. The user may define up to six categories

Description of DISPLAY commands

1. Basic commands: DRAW, EXIT, FAST, HELP, SCOPE, VIEW
2. Move model: CENTER, DISTANCE, FIELD, RESET, RESTORE, ROTATE, SAVE, SHIFT, TRANSLATE
3. Change model: EXPLODE, IMMUNE, PARTS, PIVOT, SCALE, WARP
4. Line drawing: CONTOUR, DASH, DOTTED, FEATURE, NODE, POLYGON, RECORD, SHRINK
5. Basic color: COLOR, FLAT, FRINGE, LIGHT, SMOOTH, UNIFORM
6. Advanced color: ALIAS, DIFFUSE, GLASS, HAZE, MULTIPLE, SHADOW

Select DAVINCI option: (m,i,x,#,q)

6

Advanced color: ALIAS, DIFFUSE, GLASS, HAZE, MULTIPLE, SHADOW

1. info for ALIAS command (DALIA)
2. info for DIFFUSE command (DDIFF)
3. info for GLASS command (DGLAS)
4. info for HAZE command (DHAZE)
5. info for MULTIPLE command (DMULT)
6. info for SHADOW command (DSHAD)

Select DAVINCI option: (m,i,x,#,q)

1

The following command is used to run program: DALIA
more /users/terrill/Devinci/Dmovman/alias.mov

ALIAS

The ALIAS command allows the user to toggle the anti-aliasing option off and on. This command is a switch. When given the first time, the anti-aliasing option is enabled (<ANTI-ALIASING ENABLED>). When given

...

Hit <RETURN> to continue

<return>

Select DAVINCI option: (m,i,x,#,q)

Figure 4.3 On-Line Documentation

HP 9000 Brigham Young University Software Library

1. Movie.BYU Graphics Package
2. OPTDES.BYU Design Optimization Package
3. Finite Element Modeling
4. Lumped-Mass Vibration/Modal Analysis
5. State-Space Control Systems Tools

Select DAVINCI option: (m,i,x,#,q)

1

Movie.BYU Graphics Package

1. Movie.BYU Software
2. Documentation
3. Demos

Select DAVINCI option: (m,i,x,#,q)

3

Demos

1. Clear the screen of text and graphics (CLEAR)
2. Multi-view (COMPOSE) output with UTILITY primitives (EXP9)
3. Line drawing of the robot work cell (ROBOTL)
4. Continuous color picture of robot work cell, DISPLAY (ROBOT)
5. Multi-view (COMPOSE) output, clipping by SECTION (SECOUT)
6. Color Temperature fringe output of Coyote.BYU (COYOUT)
7. Mode shape of triangular wing from SAP IV analysis (WING)
8. Line Drawing of Mode Shape of Wing Using SAP IV (WING2)

Select DAVINCI option: (m,i,x,#,q)

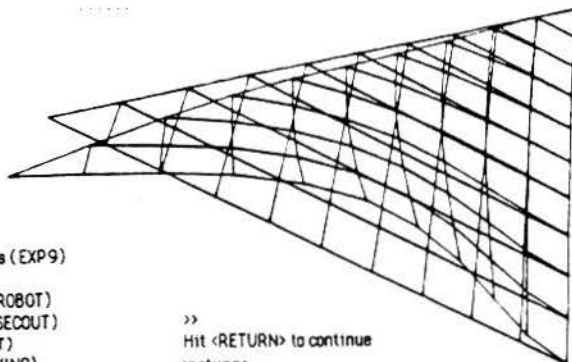
8

The following command is used to run program: WING2

/users/terrill/Dmovie/display </users/terrill/Dmovie/Drobot/answing2

<MOVIE SYSTEM DISPLAY>
<READ GEOM FILE>
<READ: 1 PARTS; 66 COORDINATES; 55 ELEMENTS.>

.....



>>

Hit <RETURN> to continue

<return>

Select DAVINCI option: (m,i,x,#,q)

Figure 4.4 Software Demonstration

to classify data files. For example, if an engineer used the sample categories "Analysis Program," "Project," and "Part" given above, then Davinci would prompt for the corresponding analysis program, project number and part number for each data file that is created. This information is organized in a simple relational data base, along with three other categories that are automatically defined: 1) file name, 2) file size, and 3) time of creation. The user may obtain selective lists of data files that fall within a specified range of one or several categories. For example, the user could request a list of data files used with analysis program Nastran; dealing with part XYZ, which were created during October. As files are backed up and removed from the system, the user can retain the data file information and store the location of each backed up file.

Two subsets of the Davinci.BYU program, Sentinel.BYU and Gateway.BYU, were created to provide advantages of the workstation environment to multi-user computing environment. Classification of data files is not included in these two programs. The library manager uses Sentinel to set up the menu information for the software in the library. Each user executes Gateway (without edit capabilities) to efficiently use and share the software. The function of the three programs is illustrated in Figure 4.5.

INDUSTRY APPLICATION

The Davinci software was used to manage BYU software used on an HP9000 engineering workstation. This application was a cooperative effort between the author and a Hewlett-Packard employee at BYU, who

was given the charter to develop ways for HP to use BYU-developed software. This software is available to HP as a member of BYU's Alliance With Industry program [45].

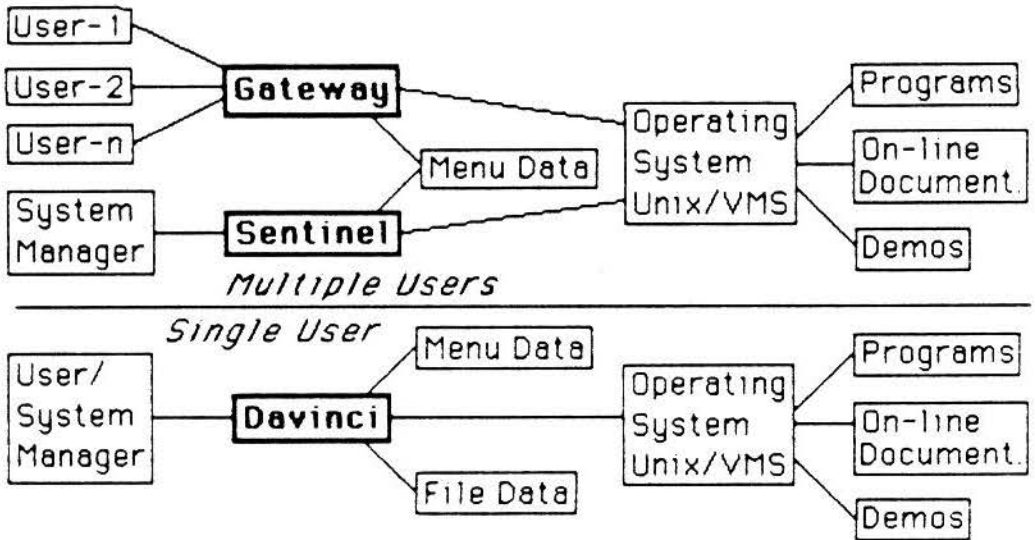


Figure 4.5 Functional Diagram of Davinci Software

Davinci.BYU was recognized as an ideal tool for organizing software on HP workstations. As the Davinci program was used for this "real world" task, it became more polished than is typical for university software. Through testing Davinci with several new users, the clarity of prompts and program information was improved.

At a recent meeting of HP's corporate Mechanical Engineering Design Council, Davinci was presented to several managers from locations throughout HP. The managers were enthusiastic about its capabilities to: 1) help engineers who are unfamiliar with Unix use applications software, and 2) allow software upgrades and maintenance from a remote location.

HP CAE managers may obtain software from their BYU liaison by requesting a tape, which is then loaded into a dedicated directory on their

9000 system. Once loaded, the BYU programs, manuals, demos, bulletin boards, etc. are immediately available within the Davinci menu structure (see Figures 4.3, 4.4). As BYU software is developed and refined, periodic updates to the Davinci database will be sent to the HP managers. Included in the Davinci information are the latest tips on using BYU software, with HP-specific instructions. By standardizing the access to programs and support (via the Davinci programs), companies such as HP can begin using BYU software quickly and with much less training than would otherwise be required.

CHAPTER 5

DEVELOPING ENGINEERING SOFTWARE

The success of engineering software, like any other product, depends on ease of use, reliability, ease of upkeep, and market demand. These factors apply to software developed for in-house use, as well as commercial codes. Good software is the result of a well-thought development procedure, style and structure rules, complete documentation guidelines, and quality software development tools. A friendly user interface is also essential for the software to be fully accepted. This chapter describes techniques that help the programmer create good software the first time. These techniques were used to develop the Rosetta and Davinci software.

A software development procedure that can help the programmer make best use of resources is given in Table 5.1. Planning, specification, and designs steps should be performed before any program coding is done. Once the program is written, many evaluation and implementation steps are also needed. Unfortunately, many programmers concentrate too much on the coding step, which can result in an unsupportable program that solves a problem that nobody cares about. The planning phase can eliminate unjustified software projects and prioritize worthy projects.

TABLE 5.1 - Software Development Process

PLANNING PHASE

1. Find out what the user wants as output.
2. Find out what the user has available for input.
3. Bracket a range of computer literacy for the expected users.
4. Estimate the expected benefits of the software, in terms of dollars and/or man-hours saved.
5. Estimate the software development time and cost.
6. Confer with management, decide if benefits justify the cost.

SPECIFICATION PHASE

7. Firm the exact content and layout of the input and output.
8. Confer with the user, make sure he agrees with the layout.
9. Establish what interaction is needed with other software or engineering databases.
10. Write a description of the calculations, theories, limitations and assumptions used in the software.
11. Establish the overall flow of the program.

DESIGN PHASE

12. Organize software into subroutines that perform a distinct function.
13. Detail the data structure for internal arrays and external files.
14. Diagram the flow of data among all the subroutines.
15. Check if existing subroutines may be used.
16. Write comment headers for the main module and each routine.

CODING PHASE

17. Write the actual FORTRAN statements. At the same time include numerous in-line comments to explain the logic.
18. Let another programmer review the code for logic errors and evaluate understandability.

EVALUATION PHASE

19. Compile the program and correct compiler errors.
20. Test the program with a complete set of sample problems.
21. Check that the software meets the requesting user's expectation.
22. Ask a casual (occasional) computer user to run the program. Fix any feature that may be confusing.

IMPLEMENTATION PHASE

23. Completely document the software. Include a user's guide that summarizes input and output, a sample execution of the program, and a program listing.
24. Move the executable file(s) to a user library area on disk.
25. Backup the source code and test data to a protected media.

The basic parts of software documentation are given in Table 5.2. Good documentation is needed to help users and programmers work with software. Since users and programmers need different information, it is best to divide documentation into two sections: user's and programmer's guides. This outline is illustrated in the Davinci and Rosetta documents, which are contained in Appendices A and B.

The quality of the user interface determines whether or not computer software is used by practicing engineers. However, the importance of the user interface is often overlooked in engineering software, because the engineer/programmer prefers to work with technical aspects of software. User interface development would be less painful if the engineer/programmer had interface utility routines available, rather than writing each interface from scratch. Squire.BYU, a general library of routines, meets this need (see documentation in Appendix C). It can be used to create a good user interface with relatively little effort from the programmer. A good interface possesses the following qualities:

TABLE 5.2 - Sections of Software Documentation

USER'S GUIDE INFORMATION	PROGRAMMER INFORMATION
1. Brief program description	1. Program description
2. Sample output, input	a) Theory and equations
3. Menu flow (optional)	b) Program flow
4. Operating instructions	c) Library, external routines
5. Input sheet (optional)	2. Program listings
	3. Test and verification
	4. Record log of changes

1. Accommodates both new and experienced users. The challenge is to permit the new user to get additional information when necessary, without boring the experienced user with long prompts. The use of three levels of prompts is adequate for most situations. The first level is terse for the experienced user. The second level is a fairly complete description and the third level suggests a response. Figure 5.1 shows a sample set of prompts for a program input. The user can move down a level by hitting the return key. A good user interface also lets the user buffer his input. Multiple responses may be given at one time, by placing them on a single input line, separated by spaces or commas. This allows the expert user to answer ahead in the software and bypass unneeded prompts.
2. Provides consistent input procedures. Today's engineer may use many software packages. A good way to achieve consistency in the user interface is to use the same input/editing utility library in each program. A big problem with utility libraries is getting programmers to use them. Programmers often resist libraries, because they were "not invented here." A good user interface should appeal to both the user and programmer. Most Squire.BYU routines are used by adding a call statement to the program and supplying an auxiliary subroutine containing input prompts. The routines handle crash-proof error recovery and input buffering.
3. Provides consistent editing procedures. In Squire.BYU, a selective data editing technique is used, where each data item is associated with an item number. Various data items are displayed on the screen simultaneously. The user selects an edit operation (such as

change, delete or insert), then identifies the item to be edited by number and enters the new value or verification for a delete. This method emulates full-screen editing, but is much less machine-dependent. A machine-dependent method for clearing the terminal screen is needed for repainting the new values after editing, but no cursor control is needed. Command buffering makes this editing procedure fast and easy to use. Each edit operation requires a call to one of four general Squire editing subroutines, and an auxiliary subroutine to arrange the data. The programmer has the flexibility to arrange the data according to the application, and the Squire routines handle the mechanics of editing.

4. Runs consistently on different computer hardware. The engineer may have to use a mainframe to solve difficult analysis problems. A portable user interface could be used with mainframe computers or workstations, providing a consistent environment for the user. As computer hardware advances are made, portability is crucial as software is moved to smaller machines [9].
5. Encourages a flexible program structure. Software that forces the user to follow a set path of operation may stifle user creativity and cause resentment [46]. The user should be able to select options as desired to operate on his data. Squire.BYU routines simplify the use of menus or command words for option selection and permit moving up and down in the program's menu structure.

The Squire library can contribute greatly to the efficiency of the user and programmer. The engineer can better use software that prompts

according to need, and flexibility encourages creativity. The programmer can develop software more efficiently because more time can be spent on developing good prompts, instead of on input techniques. Most engineering programmers prefer to work with the technical aspects of a problem rather than user interfaces. With Squire, the software naturally develops with a good user interface.

The sample executions contained in the Davinci and Rosetta user guides (Appendices A,B) are good examples of Squire functions. The author's experience with Squire suggests that users can best control programs that are a collection of menus, where options are selected by letter. The use of a single letter reduces the number of keystrokes, and association between the letter and a keyword in the command helps the user associate letters and commands. Each menu should have from three to ten options. Where there are two options, a yes/no question should be used with function QYESNO.

Select DAVINCI option: (m,i,x,e,#,q)

- m - Move through menus
- i - Information on menu items
- x - eXecute a program by menu item
number or name
- e - Edit menus, or library title
- # - select item # (execute program,
OR moves to category)
- q - Quit Davinci program

Why not enter: m to Move through
menus. NO further help!

Figure 5.1 Sample Multi-Level Prompts

In a good input procedure, each piece of data is requested individually. The use of multi-level prompts can explain what data is needed and any rules that may apply (data must be greater than zero, whole number, etc). Error traps can detect invalid data and ask the user to reenter the data.

After a reasonable number of values has been input (5-20 items), an editing routine should be used to allow the user to review his input before continuing. The same edit procedure should be used to make subsequent changes to the data. The enterprising programmer may wonder why input isn't handled directly with the editing routines, thereby saving the work of writing an input procedure. The edit routines do a poor job of input because they cannot make individualized suggestions for each piece of data. Explanatory text is limited to a few sentences for the entire set of data. Editing is more effective after the user has gone through a previous input procedure which answered any questions about the data. An editing routine may be used as a starting point with a group of default values.

CHAPTER 6

A SAMPLE PROBLEM

A sample problem was needed that could illustrate the use of Rosetta and Davinci software (using the Squire user interface library) for integrated finite element modeling. Geometry data would be generated on a Computervision CADD54 system, and subsequent data preparation and FE analysis would be done on an HP9000 workstation. This process is described in detail in this chapter, including unforeseen problems and their solutions. Such details should be of use to future workers in this area.

A free-vibration analysis of a disc drive recording head was selected to test the data structure concepts of Rosetta. This sample problem has sufficient complexity to evaluate the performance of Rosetta under realistic conditions. Figure 6.1 presents the head as displayed by Movie.BYU. This head is used on Winchester disc drives, and consists of three major parts: a stainless steel sheet metal "suspension" and "flexure" (spot welded together), and a ferrite "slider". The flexure produces a ten gram preload on the slider so it "flies" at the proper height (10 μ in.) over the disc. The actual recording device is located on the slider. Positioning this head represents a problem in servo control of a flexible structure.

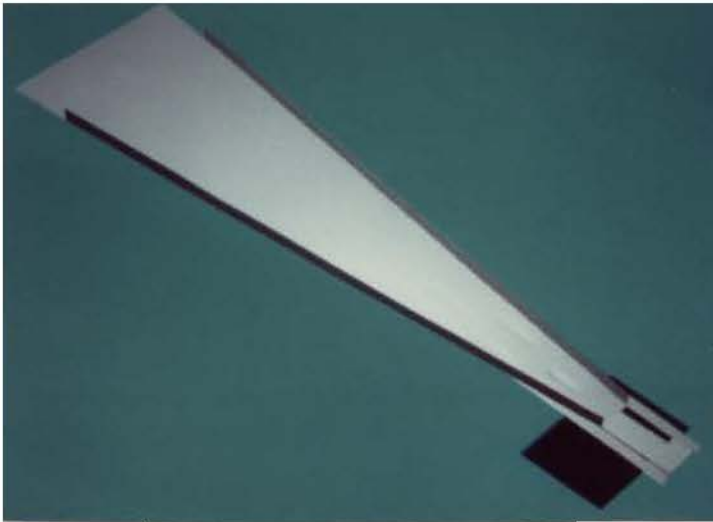


Figure 6.1 – Disc Drive Head

The disk drive head analysis was chosen as a sample problem because:

1. Modeling the assembly is a significant problem of geometric construction and mesh generation. Three separate parts must be modeled: suspension, that necks down by factor of four from one end to the other; flexure, where two narrow strips require smaller elements than needed elsewhere in the model; and slider.
2. It is an opportunity to use Movie.BYU's graphics capability to augment engineering analysis. In a high-performance servo system an understanding of system dynamics is crucial. Graphics can help the engineer understand vibration analysis output.
3. Computer graphics and Rosetta's data integration concepts were of interest to Hewlett-Packard, a participant in BYU's Alliance With Industry Program [10]. They supplied part dimensions, material characteristics, and modeling tips. The data transfer from CV to SAP IV to Movie was of interest to China Lake.

Model Generation

A wire frame model was generated using the CV CADD5 4 software. Those who participated in the CV work on the sample problem had less than 40 hours of experience with CV. As a result, the model generation that was supposed to be trivial was not. (Somehow it is always an intimidating experience for novices to confront software with a user's manual of over 1000 pages). Appendix D contains a three page reference of the CV commands needed to generate a basic model.

The first step of the modeling process was to generate a wire frame model, and then subdivide the model into flat regions. The generation of plate elements for each region was a more time consuming process. Variable node spacing was needed on the main part to preserve a good aspect ratio in the elements as the part narrowed. The CV software did not support automatic meshing with variably-spaced nodes. Consequently the nodes had to be created line by line and the elements generated by hand.

The first and second parts are connected with four spot welds, which appear as irregularities on the surface of the suspension shown in Figure 6.1. The spot welds were modeled in the mesh by sharing nodes between the two parts. It was necessary to move nodes on the parts in order to align them at the weld points. Uneven spacing in the mesh reflect adjustment in node locations. Careful use of construction planes was needed to prevent misplacement of nodes. The final mesh consisted of 555 nodes and 456 elements (see Figure 6.2). Bends in the outline of the flexure where straight lines are expected are the result of the 512 resolution used in calculating node locations in Movie.

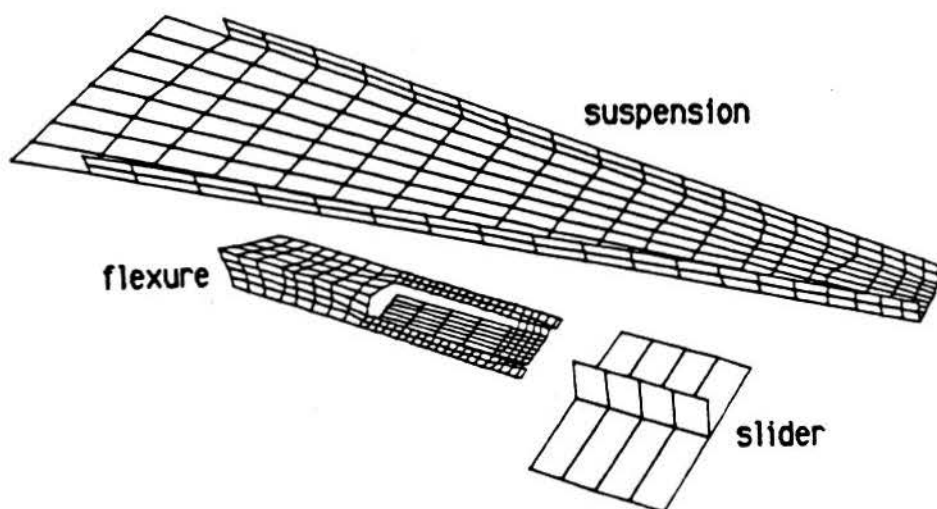


Figure 6.2 - Disc Drive Head Finite Element Mesh (Exploded View)

Neutral File Generation and Transfer

A neutral file was created for the model and written to a tape. The file was transferred to a VAX-11/750 and downloaded electronically through a data switch to an HP9000 Series 500 workstation. The neutral file was read into the Rosetta data structure.

One result from the "iterative" method of mesh generation was many gaps in the numbering of the mesh. The 555 nodes were numbered from 107 to 1079. Rosetta was used to compress the node numbering such that the nodes were numbered from 1 to 555. The compression of nodes executed so quickly that there was concern that nothing happened. A Movie.BYU data file was output, and the mesh was displayed and checked for errors. An inspection of the data revealed that all of the entities had been renumbered correctly. The program management of Davinci made it easy to switch between software during the modeling process.

It was necessary to return to the CV to rearrange the data, since the original data organization had prevented the separation of the model into parts for selective viewing. Movie.BYU requires consecutive numbering of the elements (or polygons) in a part. This was not the case in the symmetric model of the head, where one-half of each part had been constructed, then mirror-copied to obtain the final model. Back at the CV, a separate file was created for part 1, and another for parts 2 and 3. Each file was transferred to the HP workstation, and then compressed and rewritten in the neutral format using Rosetta. The node and element numbers in the second file were incremented by 400, and then the compressed files were merged. The new combined file was recompressed. The slider was separated from the flexure by hand, since it consisted of only a dozen elements.

SAP IV was selected to perform the analysis because it was the only FEA program available on BYU's HP9000. At this point a major obstacle was noted. The bandwidth [47] of the model was too large for SAP IV due to the sharing of nodes at the weld points, and the mirror-copy mesh generation technique. A low bandwidth is crucial in executing SAP IV. The CADD5 4 software does not contain a bandwidth optimizer. Some bandwidth renumbering software was available on BYU's VAX computers, but would not handle the transition between parts through the weld points.

The quickest solution was to renumber the nodes by hand. This tedious experience was simplified somewhat by using Rosetta's compression feature. An option was added to Rosetta that would input a list of the old and new node numbers. The node directory array was updated, and the node numbers were modified in the element records.

The SAP IV Analysis

Once the mesh geometry was finalized, Rosetta was used to add constraints, geometric properties, and material properties. A subroutine was developed for Rosetta to produce a SAP IV input data set. One unusual aspect of SAP IV input data is that the coordinates and constraints for each node are combined on the same line. It was a simple matter to work with the data structure to output coordinate and constraint information together. Bookkeeping concerns for the input data set were eliminated because Rosetta output the data in the correct format. Placing the data in the correct columns, etc. may be difficult when a SAP IV input data set is created by hand.

The output subroutine may also query the engineer for additional information to be added to the input file of the FEA program (such as solution specifications, time limits, error tolerances, etc.) For example, additional card images were added to the SAP IV input file to select the free vibration analysis option, and request the first six mode shapes.

The version of SAP IV on BYU's HP9000 had been modified to create a Movie.BYU geometry file for the model and a displacement file for each mode shape. The SAP IV results agreed well with the analysis done previously at Hewlett-Packard. The small differences that were noted were attributed to differences in the elements available in the FEA programs used in the two locations.

Figure 6.3 shows the extreme and equilibrium positions of a torsional mode of vibration at 2100 Hz for the disc drive head assembly. The slider has no torsional motion due to the rigid air bearing that develops between the head and the spinning disc (3000 rpm). It is

important to remember that a free vibration analysis outputs *relative* displacements, and absolute deflections are not known. The displacements are magnified in Movie.BYU to make each mode shape more understandable.

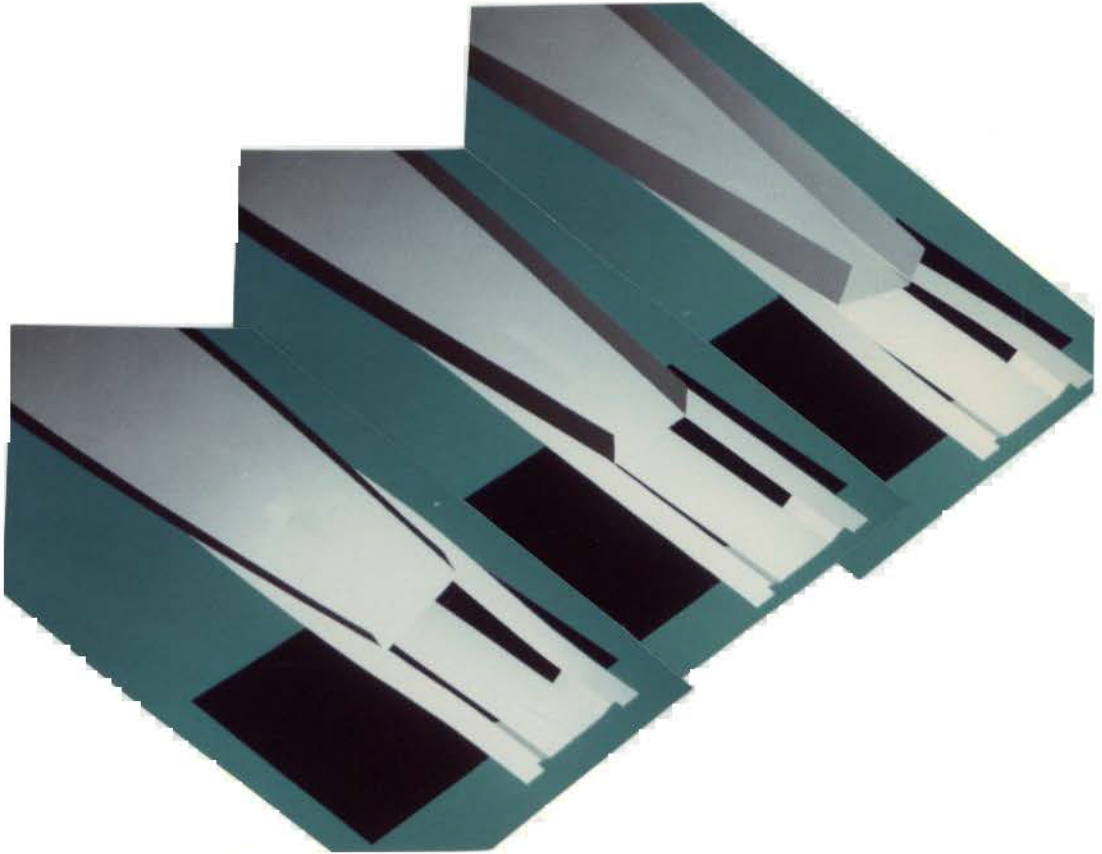


Figure 6.3 Oblique View of Torsional Mode

A front view of the torsional mode is shown in Figure 6.4. An arrow has been added to emphasize the side-to-side error in location due to vibration. The solid geometry of the slider was modeled with plate elements. The nodes in the slider are constrained in Z-translation (Z-axis is perpendicular to the disc surface), and rotation about the X- and Y-axes to model the air bearing. This allows the sparse use of plate elements to model this part.

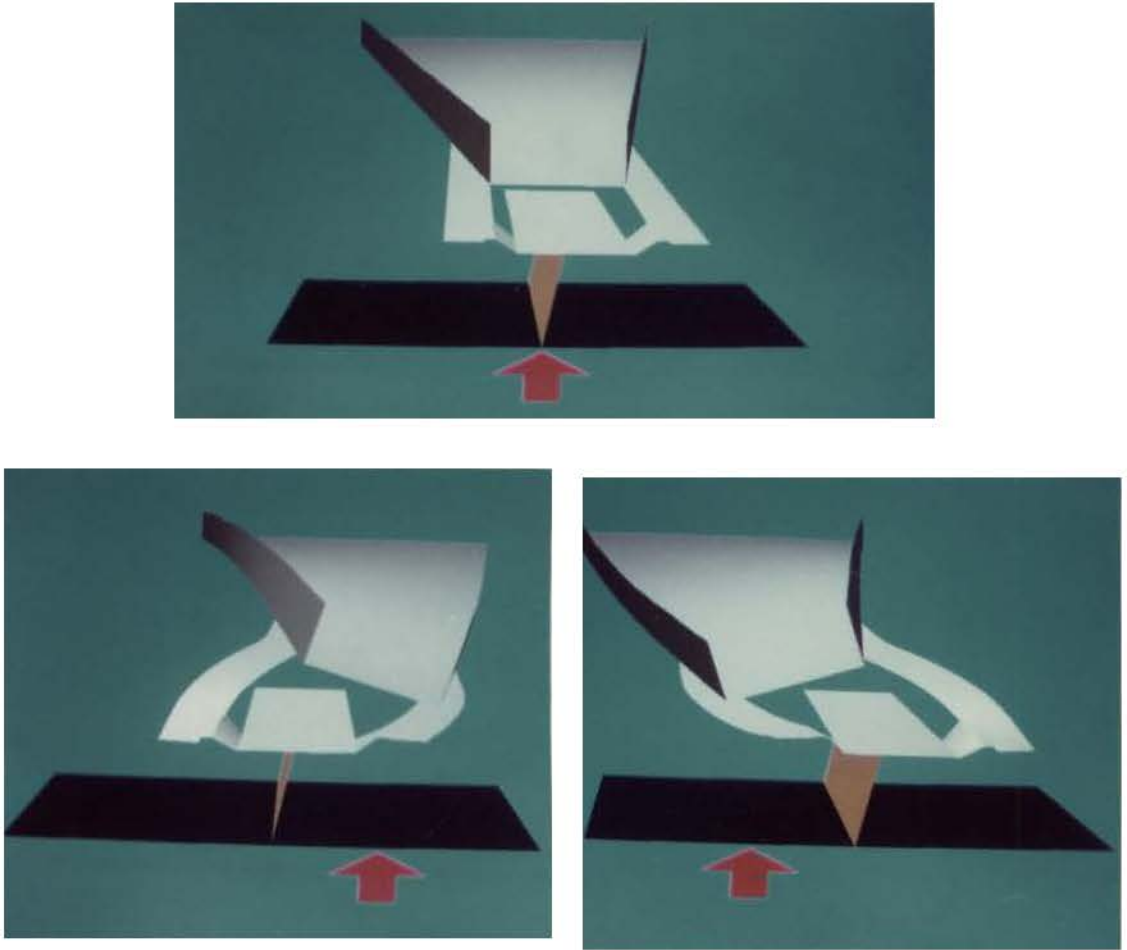


Figure 6.4 - Front View of Torsional Mode

A movie was made that animates four vibration sequences selected from the mode shapes. A computer-controlled movie camera was used continuously for 24 hours to film some 1800 frames, one at a time.

This sample problem has successfully demonstrated the Rosetta and Davinci software. These programs did not eliminate all of the problems that may occur in finite element modeling, but they did allow more attention to be focused on technical problems, rather than bookkeeping or data transfer problems.

CHAPTER 7

DISCUSSION AND CONCLUSIONS

The objective of this research was to solve specific problems in finite element data transfer and software integration, and develop an engineering workstation environment. Both specific contributions in finite element modeling and general methods of developing engineering software have resulted from these activities.

FEA Data Transfer

The Rosetta software provides a unique solution to the problem of data transfer between mesh generators and FEA programs. The finite element modeling process was simplified by linking the CV modeling and mesh generation capabilities with the analysis capability of the HP9000 and the graphics aids of Movie.BYU. As FEA tools becomes easier to use, they will become more widely applied in the engineering community. However, as experience has shown, efforts to simplify data transfer and "bookkeeping" tasks do not remove the need for the engineer's technical competence and judgement, but allow him to concentrate more time on modeling skills and design.

The development of a neutral file format in conjunction with the Navy provided a solid base for software development. The support of the

IGES standard and Movie.BYU added additional flexibility, allowing existing software to be used interchangeably. The use of Movie.BYU software to view the model as changes were made was invaluable. The use of Davinci to manage FEA tools improved the efficiency of the modeling process.

The Rosetta data structure worked well. A variety of problems were encountered along the path from CV model generation to SAP IV output. The flexibility of Rosetta's data structure helped solve these problems. Data structure operations were performed with acceptable response time for interactive use.

Engineering Workstation Environment

Although various futurists have predicted the time when the computer would greatly improve engineering productivity, advances to date have been more an increase in technical quality than productivity. Great strides have been taken to develop good engineering design and analysis software, but little work had been done to link these tools together. Much time is lost as a result.

Davinci.BYU creates an engineering workstation environment that the engineer may tailor to meet his individual needs. In addition to organizing engineering software, Davinci organizes data files in a simple relational data base.

Davinci provides a layer between the engineer and the computer operating system. The engineer's interaction with the computer may be simplified by storing frequently used system commands and procedures in the menu, which can be later invoked with a single character. Multi-tasking allows the user to execute other programs selectively from Davinci. These capabilities can improve engineer productivity.

The author found that the Davinci is not only a good tool for running application software, but also for developing new software. Software development tools can be organized and related applications can be easily accessed, without losing access to system functions such as editing.

Engineering Software Development

The Rosetta and Davinci software demonstrate a new style of engineering software. This software can be used without frequent reference to a manual, and gives the user the flexibility to be creative. More user and programmer time can be spent on technical issues, because utility routines help the programmer provide meaningful assistance to the user through multiple-level prompts for each piece of input. Edit routines help the user conveniently alter data. These features are provided by the Squire.BYU library, a general purpose set of routines written in Fortran-77. The Squire library could be used for the user interface of any Fortran program. It can help programmers be more efficient and spare much user frustration.

Future Work

Several participants in BYU's Alliance With Industry program are interested in the neutral format. Their input will help refine the format and identify additional options to be added to Rosetta. The use of this software with a wider variety of sample problems can better gage the effectiveness of the integration process.

Links should be established to other CAD systems and mesh generators by adding the capability to output a neutral or iGES file. The

ability to move finite element data to CAD systems would give the engineer more flexibility.

For example, each data field has a certain purpose and content within the node and element records. This is not the case with the constraint, load, and property records. Currently, the output subroutine for each FEA code must recognize the meaning of the various values as they were entered in Rosetta or the mesh generator. Some standardization is needed in the values stored in the constraint, load, and property records.

A new neutral file entity to define parts or regions in the neutral file would be helpful. Complex models often need to be subdivided into separate parts. The part card could have the format:

PART, ID #, Element #1, Element #2, Element #3, ... ;

Future FEA integration research at BYU will examine a combination of Movie.BYU, Rosetta.BYU, and mesh generator software to form a general package for pre- and postprocessing finite element data.

Terril Hurst of Hewlett-Packard is using the Davinci program to support computer tools in mechanical engineering within the corporation. HP feedback will be used to further enhance the software.

As the IGES standard becomes more complete for finite element data, it can more fully emulate the neutral file functions. At present, the neutral file is a more compact and concise method of storing FE data. The IGES standard can relate FE data with wire frame and surface geometry. The neutral file may change in this direction. Interaction between these two data standards would be beneficial.

REFERENCES

1. Mufti, A. A., and Jaegger, L. G., "Computer-Aided Engineering Presents Challenge," *Engineering Journal*, September 1982, pp. 8-11.
2. Acker, D. D., "Productivity: A Continuing Management Challenge," American Society of Mechanical Engineers, Paper No. 83-WA/Mgt-1.
3. Fielding, T. M., and DeJong, H. J., "Friendly FEA Sends Ripples Through a Corporation," *Computer-Aided Engineering*, July/August 1983, pp. 62-69.
4. Burchard, R., "CAEDOS Neutral File meeting minutes," Department of the Navy, Naval Weapons Center, China Lake, California, April 1984.
5. Smith, B. M., et al., *Initial Graphics Exchange Specification (IGES) - Version 2.0*, National Bureau of Standards, Report No. NBSIR 82-2631 (AF), February 1983.
6. "ABAQUS User's Manual, Version 4," Copyright 1982, Hibbitt, Karlsson, and Sorensen, July 1982.
7. Bathe, K., Wilson, E. L., Peterson, F. E., *SAP IV - A Structural Analysis Program for Static and Dynamic Response of Linear Systems*, University of California, Report No. EERC 73-11, June 1973, Revised April 1974.

8. Christiansen, H. N., and Stephenson, M. B., "Movie.BYU-A Computer Graphics Software System," *Journal of the Technical Councils of ASCE*, Vol.5, No. TCI, April 1979, pp. 3-12.
9. Hurst, T. N., and Ross, B. A., "Fortran That Travels: Programming for Portability," *Computers in Mechanical Engineering*, November 1984, pp. 25-27.
10. Dargie, P. P., Parmeshwar, K., and Wilson, W. R. D., "MAPS-1: Computer-Aided Design System for Preliminary Material and Manufacturing Process Selection," *Journal of Mechanical Design*, January 1982, pp. 126-136.
11. Toffler, A., *The Third Wave*, New York: William Morrow and Company, Inc., 1980, p. 66.
12. Chase, M. A., and Dawson, G. A., "An End to the Guesstimate in Mechanical Design," *Computer-Aided Engineering*, July/August 1983, pp. 22-30.
13. Werner, F. D., "Equation Solver for Engineers," *Computers in Mechanical Engineering*, November 1983, pp. 53-59.
14. Seireg, A. A., "The Growth of Automation Demands a Grasp of the Whole," *Computers in Mechanical Engineering*, Sept. 1983, p. 2.
15. Ross, B. A., and Chase, K. W., "Computer-Aided Analysis of Stiffness Sensitive Linkages in Multiple Positions," Society of Automotive Engineers Paper No. 821078, 1982.
16. Fong, H. H., "Interactive Graphics and Commercial Finite Element Codes," *Mechanical Engineering*, June 1984, pp. 18-25.
17. Krose, J. K., "Selecting a Graphic Input Device for CAD/CAM," *Machine Design*, October 6, 1983, pp. 75-80.

18. Ross, B. A., and Ulrich, R. D., "Applying 2-D Finite Element Analysis to Heat Transfer Problems," *Computers in Mechanical Engineering*, March 1984, pp. 16-19.
19. Panasuk, C., "Software Standards Will Usher in the Age of Graphics," *Electronic Design*, July 12, 1984, pp. 94-106.
20. Borrell, J., "Engineering Workstations Meet Demands for Individual Design Needs," *Digital Design*, December 1983, pp. 87-92.
21. Zecher, J. E., et al., "Engineering Workstations Meet Demands for Individual Design Needs," *Mechanical Engineering*, November 1983, pp. 50-61.
22. Schaeffer, H., "Putting the Pieces Together for CAE," *Computer-Aided Engineering*, March/April 1984, p. 112.
23. Potter, R. J., "Data Processing in Blue Jeans," *Computer*, March 1983, pp. 73-77.
24. Peters, T. J., and Waterman, R. W. Jr., *In Search of Excellence*, New York: Warner Books, 1982, p. 221.
25. ME Staff, "Pat Hanratty Speaks Out on Interactive Graphics," *Mechanical Engineering*, November 1983, pp. 50-61.
26. Dube, R. P., and Johnson, H. R., "Computer-Assisted Engineering Data Base," American Society of Mechanical Engineers, Paper No. 83-WA/Aero-11.
27. Finkel, J. I., "Building a Wall Around an Engineering Data Base," *Computer-Aided Engineering*, March/April 1984, pp. 76-82.
28. Felippa, Carlos A., "Database Management in Scientific Computing-I. General Description," *Computers and Structures*, V.10, pp. 53-61.

29. Navathe, S. B., "Data Base Management of Computer-Aided-Design Data," Data Base Implementation and Applications, *AICHE Symposium Series*, No. 231, Vol 79.
30. Dube, R. P., and Smith, M. R., "Managing Geometric Information with a Database Management System," *IEEE Computer Graphics and Applications*, October 1983, pp. 57-62.
31. Christman, A. M., "The Coupling of CAD and CAM," *Computers in Mechanical Engineering*, September 1983, pp. 26-29.
32. Affuso, T. and Sevak, N., "Integration Through Translation: The CAD/CAM Pilot Probe at Xerox," *Computers in Mechanical Engineering*, September 1983, pp. 14-24.
33. Interview with Philip Kennicot, General Electric Corporate Research and Development Center, Schenectady, New York, June 12, 1984.
34. *Second Annual CAEP Users Meeting*, Naval Weapons Center, China Lake California, January 1984.
35. *PDA/Patran-G User's Guide*, PDA Engineering, Santa Ana, California, 1980, pp. 18.1-18.28.
36. IGES FEM Subcommittee, *IGES Request for Change #135, 164, 247*, drafts obtained from Bob Ivey, subcommittee chairman.
37. Radisich, M., "Taking the Drudgery Out of Engineering," *Computer-Aided Engineering*, March/April 1984, pp. 42-47.
38. Hirsch, A., "Toolkit Extends the Benefits of Lisp-Based Computer to Fortran Programming," *Electronic Design*, May 31, 1984, pp. 193-202.
39. Borrel, J., "Engineering Workstations Meet Demands for Individual Design Needs," *Digital Design*, December 1983, pp. 87-92.

40. Villapiano, G., "Slow in Coming, A Full Complement of Software for Software Design on Workstations is Not Yet Here, But Many of the Pieces Now Exist and Are Up and Running," *Electronic Design*, May 31, 1984, pp. 110-126.
41. Collett, R., "Trends and Developments in Engineering Workstations," *Digital Design*, October 1984, pp. 54-58.
42. Digital Equipment Corporation, *VAX-11 Run-Time Library User's Guide*, Maynard, Massachusetts, April 1982, p. 5-10.
43. Kernighan, B. W., and Ritchie, D. M., *The C Programming Language*, Bell Telephone Laboratories, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978, p. 157.
44. Waite, M., et al., *UNIX Primer Plus*, Howard W. Sams & Co., Inc., Indianapolis, Indiana, 1983, pp. 250-254.
45. Benzley, S., and Chase, K., "Computer-Aided Engineering, Design, and Manufacturing at Brigham Young University," *Proceedings*, NCGA Annual Meeting, Anaheim, California, May 1984.
46. Smith, C. C., et al., "Mechanical Systems Applications of Real-time Computer Graphics," *Proceedings*, NSF Study of Supercomputers in Mechanical Systems Research, Lawrence Livermore Laboratory, Livermore, California, September 1984.
47. Reiger, N. F., and Steele, J. M., "The Basics of Finite Element Modeling: A Plain Language Guide for Designers," *Machine Design*, April 9, 1984, pp. 165-170.

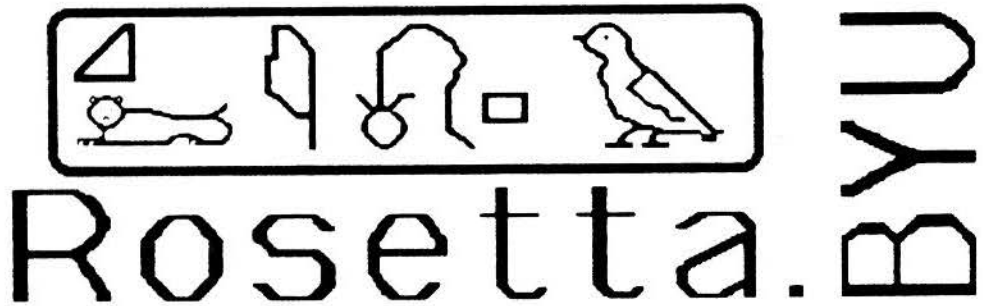
APPENDIX A

ROSETTA DOCUMENTATION

Rosetta.BYU

User and Programmer Guide

Brant A. Ross
College of Engineering
Brigham Young University
Provo, Utah 84602



Neither Brigham Young University nor its employees makes any warranty expressed or implied, or assumes any legal responsibility for the accuracy, completeness or usefulness of this computer program or its associated documentation. Readers are reminded that the information and ideas contained in this document are the property of Brigham Young University and may not be used without permission.

TABLE OF CONTENTS

USER'S GUIDE INFORMATION

Program Description	68
Sample Output	70
Rosetta.BYU Sample Execution	74

PROGRAMMER INFORMATION

Data Standards	81
Rosetta Data Structure	84
Adding Input/Output Subroutines. ...	90
Element Type Data Base.	91

User's Guide Information

PROGRAM DESCRIPTION .

Rosetta.BYU is a computer program written in Fortran-77 that accepts finite element modeling (FEM) information in a neutral format, allows editing, and produces input data sets for various Finite Element Analysis (FEA) programs. It provides a link between mesh generators (stand-alone or CAD system based) and analysis programs, through data standards (see Figure A.1).

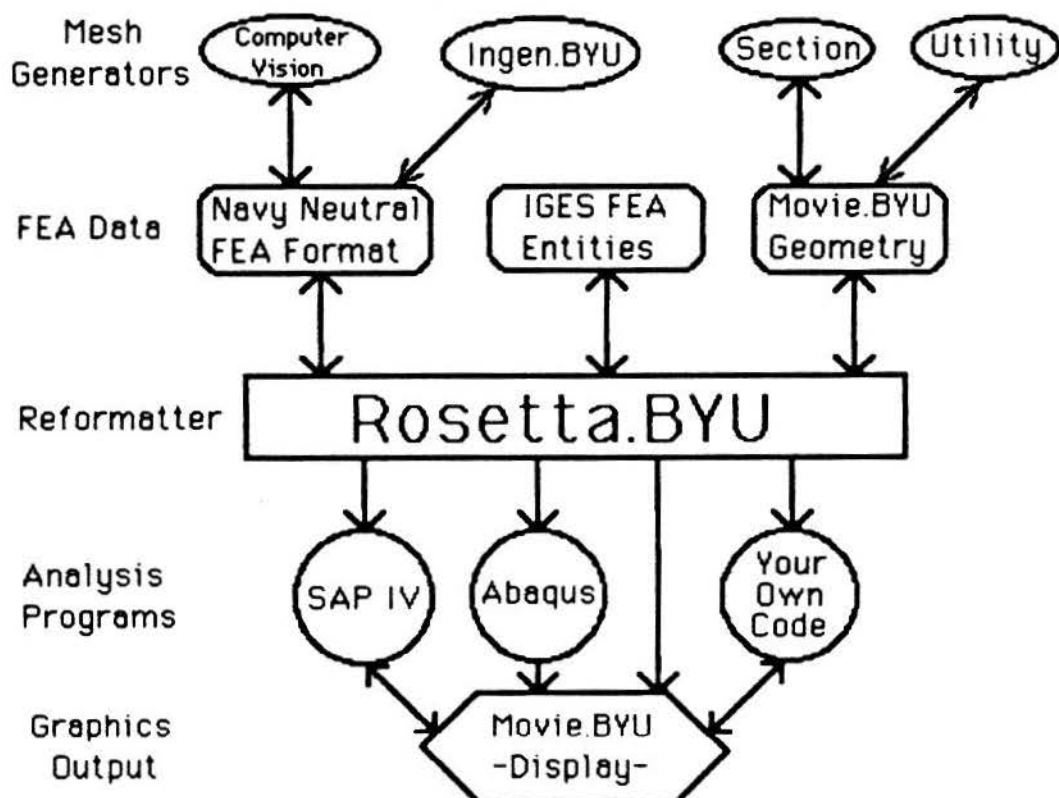


Figure A.1 Finite Element Modeling Data Paths

Most mesh generators have the capability to create input data sets for popular FEA codes such as Nastran, Ansys, SAP, etc. However, the user may also need to transfer data to specialized FEA codes that have no direct data path. Rosetta opens that data path via three standard formats: 1) the Navy FEM neutral file developed jointly by China Lake NWC and BYU, 2) the IGES Version 2.0 FEM entities (nodes and elements only), and 3) the Movie.BYU polygon data file. This version of Rosetta.BYU allows FEM output to be reformatted into an input data set for either SAP IV or ABAQUS. Support of additional programs requires the addition of a subroutine to unload data in the format of that program.

Rosetta uses a custom data structure to maintain relationships between finite element entities. The data structure provides the following capabilities:

1. Model checking to detect missing nodes and elements, and elements without material properties.
2. Complete editing for each of the 14 FEA data entries. Geometry may be input from a file, and the remaining portion of the model entered interactively.
3. Compression of entities to eliminate gaps in ID numbers, which are unacceptable to some FEA programs. Gaps are sometimes left in the model by the mesh generator.
4. Reordering of the data set to make it easier to understand.

This program uses the Squire.BYU library for terminal input and editing. It does not require a graphics terminal.

SAMPLE OUTPUT

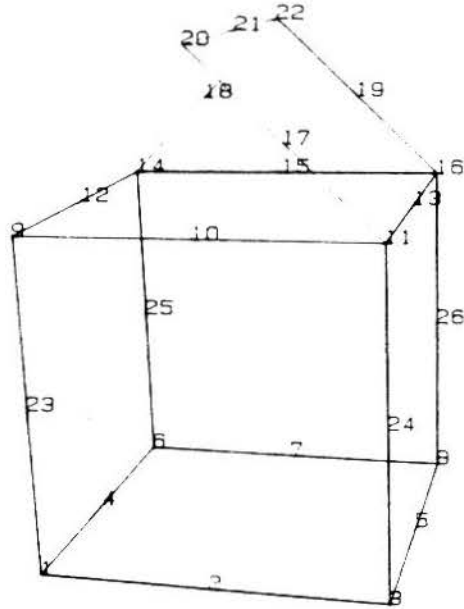
The data for the small model shown is listed below in the Navy neutral, IGES, and Movie.BYU formats. Note the difference in file size between the Navy and IGES files.

Navy Neutral Format

```

HEAD, This model tests the Rosetta program;
NODE, 1,, 0.0,0.0,2.0;
NODE, 2,, 1.0,0.0,2.0;
NODE, 3,, 2.0,0.0,2.0;
NODE, 4,, 0.0,0.0,1.0;
NODE, 5,, 2.0,0.0,1.0;
NODE, 6,, 0.0,0.0,0.0;
NODE, 7,, 1.0,0.0,0.0;
NODE, 8,, 2.0,0.0,0.0;
NODE, 9,, 0.0,2.0,2.0;
NODE, 10,, 1.0,2.0,2.0;
NODE, 11,, 2.0,2.0,2.0;
NODE, 12,, 0.0,2.0,1.0;
NODE, 13,, 2.0,2.0,1.0;
NODE, 14,, 0.0,2.0,0.0;
NODE, 15,, 1.0,2.0,0.0;
NODE, 16,, 2.0,2.0,0.0;
NODE, 17,, 1.5,2.5,2.0;
NODE, 18,, 0.5,2.5,0.0;
NODE, 19,, 1.5,2.5,0.0;
NODE, 20,, 1.0,3.0,2.0;
NODE, 21,, 1.0,3.0,1.0;
NODE, 22,, 1.0,3.0,0.0;
NODE, 23,, 0.0,1.0,2.0;
NODE, 24,, 2.0,1.0,2.0;
NODE, 25,, 0.0,1.0,0.0;
NODE, 26,, 2.0,1.0,0.0;
ELEM, 1,C3D20 ,0.0, 1,3,8,6,9,11,16,14,2,5,7,4,10,13,15,12,23,24,26,25;
ELEM, 2,CPS8 ,0.0, 11,16,22,20,13,19,21,17;
ELEM, 3,DINTER3,0.0, 16,15,14,19,22,18;

```

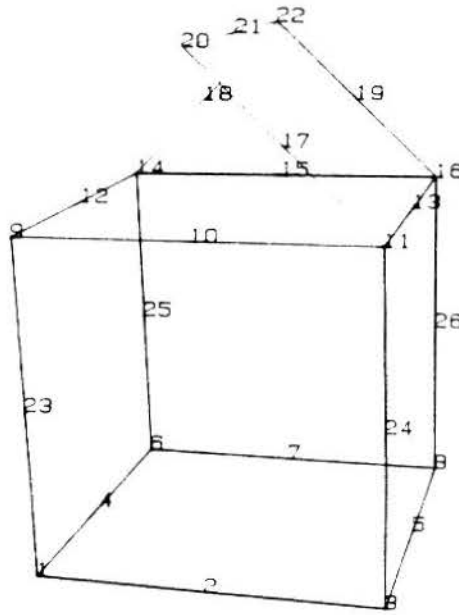


IGES Version 2.0 Format

This model tests the Rosetta program

.....,11HRosetta.BYU;

134 1
 134
 134 2
 134
 134 3
 134
 134 4
 134
 134 5
 134
 134 6
 134
 134 7
 134
 134 8
 134
 134 9
 134
 134 10
 134
 134 11
 134
 134 12
 134
 134 13
 134
 134 14
 134
 134 15
 134
 134 16
 134
 134 17
 134
 134 18
 134
 134 19
 134
 134 20
 134
 134 21
 134
 134 22
 134

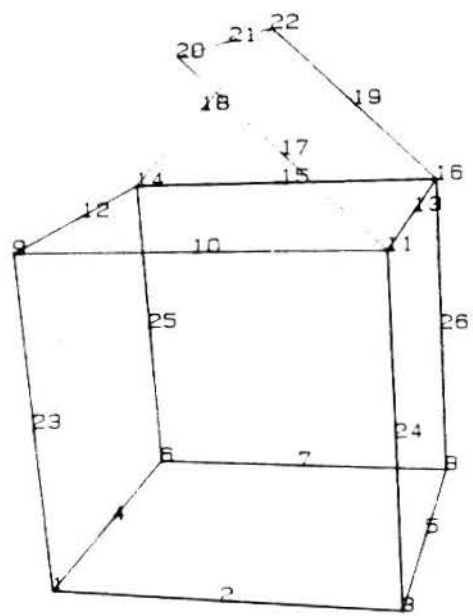


S 1
 G0000001
 D 1
 1D 2
 D 3
 2D 4
 D 5
 3D 6
 D 7
 4D 8
 D 9
 5D 10
 D 11
 6D 12
 D 13
 7D 14
 D 15
 8D 16
 D 17
 9D 18
 D 19
 10D 20
 D 21
 11D 22
 D 23
 12D 24
 D 25
 13D 26
 D 27
 14D 28
 D 29
 15D 30
 D 31
 16D 32
 D 33
 17D 34
 D 35
 18D 36
 D 37
 19D 38
 D 39
 20D 40
 D 41
 21D 42
 D 43
 22D 44

134 23
 134
 134 24
 134
 134 25
 134
 134 26
 134
 136 27
 136
 136 29
 136
 136 30
 136

134, 0.0, 0.0, 2.0, 0.0;
 134, 1.0, 0.0, 2.0, 0.0;
 134, 2.0, 0.0, 2.0, 0.0;
 134, 0.0, 0.0, 1.0, 0.0;
 134, 2.0, 0.0, 1.0, 0.0;
 134, 0.0, 0.0, 0.0, 0.0;
 134, 1.0, 0.0, 0.0, 0.0;
 134, 2.0, 0.0, 0.0, 0.0;
 134, 0.0, 2.0, 2.0, 0.0;
 134, 1.0, 2.0, 2.0, 0.0;
 134, 2.0, 2.0, 2.0, 0.0;
 134, 0.0, 2.0, 1.0, 0.0;
 134, 2.0, 2.0, 1.0, 0.0;
 134, 0.0, 2.0, 0.0, 0.0;
 134, 1.0, 2.0, 0.0, 0.0;
 134, 2.0, 2.0, 0.0, 0.0;
 134, 1.5, 2.5, 2.0, 0.0;
 134, 0.5, 2.5, 0.0, 0.0;
 134, 1.5, 2.5, 0.0, 0.0;
 134, 1.0, 3.0, 2.0, 0.0;
 134, 1.0, 3.0, 1.0, 0.0;
 134, 1.0, 3.0, 0.0, 0.0;
 134, 0.0, 1.0, 2.0, 0.0;
 134, 2.0, 1.0, 2.0, 0.0;
 134, 0.0, 1.0, 0.0, 0.0;
 134, 2.0, 1.0, 0.0, 0.0;

136, 18, 20, 1, 3, 5, 47, 21, 19, 17, 45, 7, 9, 25, 23, 11, 13, 15, 51, 31, 29, 27,
 49, 5HC3D20;
 136, 6, 8, 21, 25, 31, 37, 43, 41, 39, 33, 4HCPS8;
 136, 3, 6, 31, 29, 27, 35, 43, 37, 7HDINTER3;
 S 16 1D 58P 30



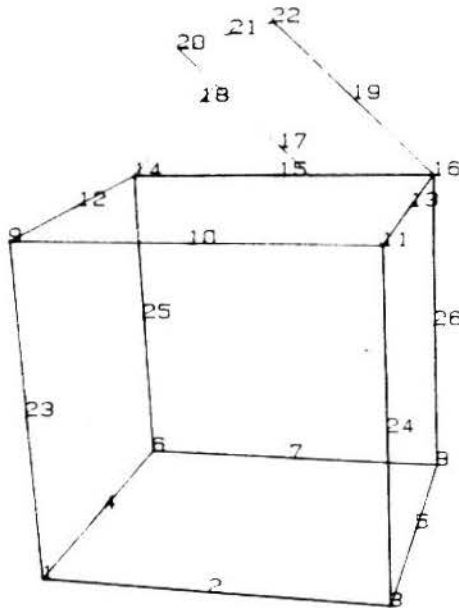
D 45
 23D 46
 D 47
 24D 48
 D 49
 25D 50
 D 51
 26D 52
 D 53
 1D 54
 D 55
 2D 56
 D 57
 3D 58
 1P 1
 3P 2
 5P 3
 7P 4
 9P 5
 11P 6
 13P 7
 15P 8
 17P 9
 19P 10
 21P 11
 23P 12
 25P 13
 27P 14
 29P 15
 31P 16
 33P 17
 35P 18
 37P 19
 39P 20
 41P 21
 43P 22
 45P 23
 47P 24
 49P 25
 51P 26
 53P 27
 53P 28
 55P 29
 57P 30
 T 1

Movie.BYU Polygon Format

```

1 26 8 62
1 8
0.00000E+00 0.00000E+00 2.00000E+00 1.00000E+00 0.00000E+00 2.00000E+00
2.00000E+00 0.00000E+00 2.00000E+00 0.00000E+00 0.00000E+00 1.00000E+00
2.00000E+00 0.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
1.00000E+00 0.00000E+00 0.00000E+00 2.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 2.00000E+00 2.00000E+00 1.00000E+00 2.00000E+00 2.00000E+00
2.00000E+00 2.00000E+00 2.00000E+00 0.00000E+00 2.00000E+00 1.00000E+00
2.00000E+00 2.00000E+00 1.00000E+00 0.00000E+00 2.00000E+00 0.00000E+00
1.00000E+00 2.00000E+00 0.00000E+00 2.00000E+00 2.00000E+00 0.00000E+00
1.50000E+00 2.50000E+00 2.00000E+00 5.00000E-01 2.50000E+00 0.00000E+00
1.50000E+00 2.50000E+00 0.00000E+00 1.00000E+00 3.00000E+00 2.00000E+00
1.00000E+00 3.00000E+00 1.00000E+00 1.00000E+00 3.00000E+00 0.00000E+00
0.00000E+00 1.00000E+00 2.00000E+00 2.00000E+00 1.00000E+00 2.00000E+00
0.00000E+00 1.00000E+00 0.00000E+00 2.00000E+00 1.00000E+00 0.00000E+00
  1  2  3 24 11 10  9 -23  6 25 14 15 16 26  8 -7
  3  5  8 26 16 13 11 -24 11 13 16 15 14 12  9 -10
  9 12 14 25  6  4  1 -23  1  4  6  7  8  5  3 -2
11 13 16 19 22 21 20 -17 16 15 14 18 22 -19

```



Rosetta.BYU Sample Execution

This section contains a step-by-step sample execution of Rosetta. Major options and input/editing procedures of the Squire.BYU library are shown. The Squire.BYU library implements multi-level prompts and buffering of commands to accommodate varied user capabilities. The first level prompt is terse for the expert user, the second level is a fairly complete description, and the third level suggests a response. The second and third level prompts are shown when the user hits the return key. The expert user may also answer questions ahead by placing the appropriate responses on a single line, separated by spaces.

Comments shown in *italics* do not appear on the terminal screen but have been added here for description purposes. User input is shown in **bold** to make the session easier to understand.

login: ross *(login procedure)*
 Password:

Brigham Young University - Mechanical Engineering Department
 Welcome to Hewlett-Packard System 9000 HP-UX

Welcome sir: Your terminal is set to be a vt100

[1] cd format

[2] ls *(check for input data file)*

Dobject/	convert*	hole.igs	spline.out	test.neu
Dsource/	convert.f	outrose*	test.geo	
EDBASE	flexure.n7	rosetta*	test.igs	
EDBASE.SEQ	hole.dat	rosetta.for	test.igs2	
EDBASE.seq	hole.del	spline.dat	test.mov	

[3] rosetta *(execute the program)*

Enter terminal type: (1,2,3,4)

<return> *(hitting return for 2nd level prompt)*

- 1 - Hewlett-Packard Terminals
- 2 - Digital VT100
- 3 - Visual 200/500
- 4 - Dumb Terminal (no cursor control)

2 *(pick terminal type so the Squire utilities know how to clear screen)*

```
*****
* Rosetta.BYU - Finite Element Data Translator & Reformatter *
*
* Note: at any time you can...
* 1) hit the return key to get more information.
* 2) Answer question ahead (if desired) by entering the
* corresponding answers, separated by spaces.
*****
```

Select input file type (i,n,m,q):

<return>

Select the type of input file to read in:

- i - IGES version 2.0
- n - Navy neutral file, version 1.0
- m - Movie.byu geometry file
- q - Quit execution of Rosetta

n

Initializing FEM data structure....

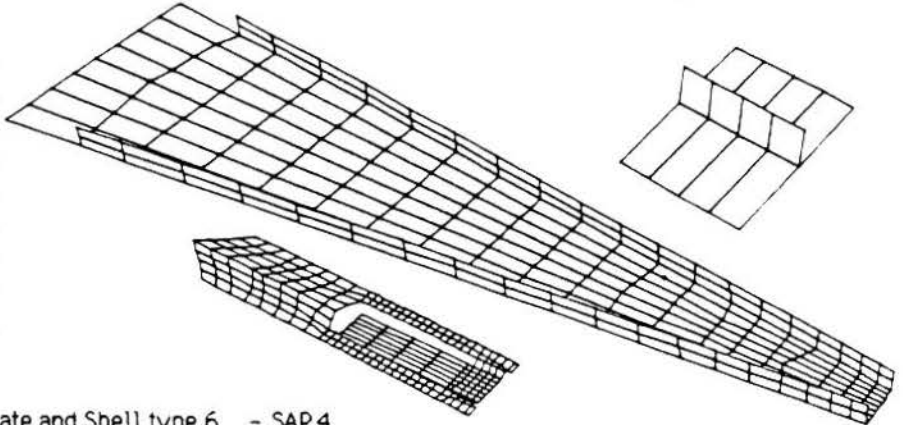
Navy Neutral File Input.

Enter data set name (15 CHARS MAX).

flexure.n7

*(this file was generated by a CV CAD system,
with 456 elements, 555 nodes, as shown below:)*

50 Lines Read...
100 Lines Read...
150 Lines Read...
200 Lines Read...
250 Lines Read...
300 Lines Read...
350 Lines Read...
400 Lines Read...
450 Lines Read...
500 Lines Read...
550 Lines Read...



SHELL Plate and Shell type 6 - SAP4

Connectivity: 1 2 3 4

(as new element types are read,

IGES Topology # 5:LQUAD - Linear Quadrilateral

related information is shown)

600 Lines Read...
650 Lines Read...
700 Lines Read...
750 Lines Read...
800 Lines Read...
850 Lines Read...
900 Lines Read...
950 Lines Read...
1000 Lines Read...

Hit <RETURN> to continue

Summary of Maximum ID numbers:

Elements = 456	Constraints:	Loads:
Element Types = 1	Permanent = 0	Element = 0
Nodes = 555	Multiple = 0	Nodal = 0
	Omitted DOF = 0	Harmonic = 0
Material Props = 0	Retained DOF = 0	
Geometric Props = 0		Trans Matrices = 0

Storage: Integer=30.8% Real= .0% Text= .0%

Select process option (c,e,i,l,s,w,q):

<return>

Select a processing option: *(main options of Rosetta)*

- c - Check for model completeness
- e - Edit the current model data
- i - read in a different Input file
- l - List the current model data
- s - give Status of data storage
- w - Write out the model to a file
- q - Quit execution of Rosetta

e

Select edit option (a,h,n,e,c,p,l,t,q):

<return>

Select an edit option: *(edit options of Rosetta)*

- a - Add/delete items to node/element lists
- h - Header information
- n - Node data
- e - Element data
- c - Constraint data (p,m,o,r)
- p - Property data (m,g)
- l - Load data (n,e)
- t - Transformation matrix data
- q - quit list/edit option

c

Select constraint type to list/edit (p,m,o,r,q):

<return>

- p - Permanent constraint set
- m - Multiple constraint data
- o - Omitted degree-of-freedom constraint set
- r - Retained degree-of-freedom constraint set
- q - Quit load list/edit.

d

Enter EDIT command: (c,d,i,l,q)

<return>

c - change
 d - delete
 i - insert
 l - list data
 q - quit edit

i

Enter entity ID number to edit:

1

Please enter the DOF(s) constrained:

123456

Enter value for displacement of permanent constraint

0

Create List of Nodes for this Constraint

Enter beginning, ending ID number for range:

1 9

Enter beginning, ending ID number for range:

<return>

You need to generate an element or node list. To simplify the input, a range of numbers are entered each time by inputting low and high ID numbers. For example, by entering: 1 5 the number 1,2,3,4, and 5 will be added to the list. Single numbers are entered by making the ending number in the range the same or less as the beginning number. To make a list with 1 3 5 6 7 8 9 10, you would enter 1 1 3 3 5 10 . Enter q to quit entering ranges of numbers for the list.

q

1) 1 2) 2 3) 3 4) 4 5) 5
 6) 6 7) 7 8) 8 9) 9

This data OK? (y,n)

y

EDIT OF PERMANENT CONSTRAINT # 1

Degree of freedom constrained: 1) 123456

Magnitude of permanent constraint: 1) .00000000E+00

Constrained Nodes:

2) 1 3) 2 4) 3 5) 4 6) 5 7) 6
 8) 7 9) 8 10) 9

This data OK? (y,n)

y

Select edit option (a,h,n,e,c,p,l,t,q):

c p i 2

Please enter the DOF(s) constrained:

345

Enter value for displacement of permanent constraint

0

Create List of Nodes for this Constraint

Enter beginning, ending ID number for range:

491 494 505 512 518 520 q *(buffering responses)*

1) 491 2) 492 3) 493 4) 494 5) 505

6) 506 7) 507 8) 508 9) 509 10) 510

11) 511 12) 512 13) 518 14) 519 15) 520

This data OK? (y,n)

<return>

Enter: y -to accept data as shown.

n -to make changes.

n

Enter EDIT command: (c,d,i,l,q)

<return>

c - change

d - delete

i - insert

l - list data

q - quit edit

C

Change WHICH item:

q

Enter EDIT command: (c,d,i,l,q)

d

Delete WHICH item?

14

Item # 14 = 519

Delete this item? (y/n)

y

1) 491 2) 492 3) 493 4) 494 5) 505

6) 506 7) 507 8) 508 9) 509 10) 510

11) 511 12) 512 13) 518 14) 520

Enter EDIT command: (c,d,i,l,q)

q

EDIT OF PERMANENT CONSTRAINT # 2

Degree of freedom constrained: 1) 345

Magnitude of permanent constraint: 16) .00000000E+00

Constrained Nodes:

2) 491 3) 492 4) 493 5) 494 6) 505 7) 506

8) 507 9) 508 10) 509 11) 510 12) 511 13) 512

14) 518 15) 520

This data OK? (y,n)

y

Select edit option (a,h,n,e,c,p,l,t,q):

q

Select process option (c,e,i,l,s,w,q):

l

Select category to list (a,n,e,m,g,l,c,t,q):

<return>

- a - list All model data
- n - Nodes
- e - Elements
- m - Material properties
- g - Geometric properties
- l - Loads
- c - Constraints
- q - Quit list option

C

Save copy to disk file? (y/n)

n

CONSTRAINT INFORMATION

Permanent Constraints:

ID

NUM	DOF	VALUE	Nodes:										
1	123456	.0000E+00	1	2	3	4	5	6	7	8	9		
2	345	.0000E+00	491	492	493	494	505	506	507	508	509	510	511
			512	518	520								

Select category to list (a,n,e,m,g,l,c,t,q):

q

Select process option (c,e,i,l,s,w,q):

w

Select output file type (c,i,m,n,s,q): *(output options of Rosetta)*

<return>

Select the type of output file to write out:

- c - Coyote.BYU heat transfer program
- i - Iges version 2.0 format
- m - Movie.byu polygonal format
- n - Navy neutral file version 1.0
- s - Sap iv input data file
- q - Quit program execution

S

Output of SAP IV Input Data File...

_____ Enter data set name (15 CHARS MAX).

flexsap.out

Select type of analysis to perform (s,e,f,r,d,q): *(FEA Code-specific questions)*

<return>

Select the type of analysis to perform with SAP IV:

- s - Static analysis
- e - Eigenvalue/eigenvector solution
- f - Forced dynamic response by mode superposition
- r - Response spectrum analysis
- d - Direct step-by-step integration
- q - Quit attempt to create a SAP IV input data set

e

Enter number of natural frequencies to be found:

6

Reviewing constraints....

Writing Node Cards...

Writing Element Cards...

Select process option (c,e,i,l,s,w,q):

q

[4] exit

(log off computer)

Programmer Information

DATA STANDARDS

NAVY NEUTRAL STANDARD

The Navy Neutral file was developed jointly between the Naval Weapons Center at China Lake, California and Brigham Young University. It stores finite element information as one of thirteen entities within a free-format text file. Each entity (node, element, etc.) occupies a separate record and is identified by a four-character keyword. An identification (ID) number follows the keyword. The entities, their keywords, and content are shown in Table A.1. Data fields are separated by commas, and each record is terminated with a semicolon (See Figure A.2). This format has the following advantages:

1. Neutral file is human-readable.
2. Changes may be made by hand using a standard text editor.
3. Information is not column dependent.
4. Each record may occupy as many lines in the file as needed.

The element and nodal load records consist solely of a group of elements or nodes, as shown in the Content section of Table 1. Additional information is usually needed to complete the load information, depending on the type of analysis (linear or nonlinear, static or dynamic, constant or variable load history, etc.). The additional information is entered by the

user when the neutral data is reformatted to prepare an input set for a particular FEA program. As more experience is gained with the neutral format, more information may be added to the load cards.

Table A.1 FEA Neutral File Entities

<u>Type</u>	<u>Key</u>	<u>Content</u>
Header	HEAD	Comment or title card
Node	NODE	Transf. matrix #, coordinates, scalar
Element	ELEM	Name, mat'l #, geom prop #, node #'s
Material Property	MATL	Name, isotropic condition, reference type, reference value, property values
Geometric Property	PROP	Name, property values
Load, Element	ELOD	Element #'s
Load, Nodal	NLOD	Node #'s
Load, Harmonic	HARM	DOF, displacement value, node #'s
Constraint, Perm	PCON	DOF(s), displacement value, node #'s
Constraint, Multiple.	MCON	Dependent, independent node #'s, ratios
Constraint, Omitted DOF	ODOF	Degree of freedom omitted, node #'s
Constraint, Retained DOF	RDOF	Degree of freedom retained, node #'s
Transformation Matrix	TRAN	Coordinate system type, matrix values

```

NODE,127,0,-.18370E2,0,.,10500E1,0.;
NODE,273,0,-.16214E2,-.77640E0,.97686E0,0.;
ELEM,83,QUAD,0,0,260,259,112,113;
ELEM,1,QUAD,0,0,148,273,126,127;
ELEM,2,QUAD,0,0,169,189,273,148;
NODE,275,0,.39994E1,-.37848E0,.29113E0,0.;
NODE,108,0,.49734E1,0,.,25809E0,0.;
MATL,3,STEEL,ISOTROPIC,TEMP,100,.,37058E7,.14028E7,
0,.,37058E7,0,.,14028E7,.00772;

```

Figure A.2 Sample Portion of Neutral File from CV

IGES VERSION 2.0 STANDARD

The IGES Version 2.0 format includes node and element entities, but not loads, constraints, or properties. The IGES FEM Subcommittee has suggested formats for the missing entities through "Request For Change" documents [8]. Rosetta uses the suggested IGES format for nodes and elements. As other entities are officially added to the standard, they will be added to Rosetta.

Each IGES entity is stored as a combination of two directory records and a parameter record. Each directory record contains 10 fields. The first field identifies the entity type (134=Node, 136=Element). The second field of the first card contains the number of the associated parameter card and the eighth and ninth field of the second record contains the entity label and entity ID number. Other directory fields are not used by Rosetta.

Parameter records store data in a free format. The node parameter record contains the entity type and the X, Y, and Z coordinates. The element parameter record contains the entity type, IGES topology type, number of node in element, pointers to directory records of associated nodes, and element type name.

MOVIE.BYU POLYGONAL FORMAT

Movie.BYU uses either a polygonal or solid representation of objects. Rosetta can translate a polygon file to an equivalent finite element mesh, allowing Movie utilities to create mesh geometry. Both polygon and solid Movie data files may be created from Navy neutral or IGES format data files, allowing display and verification of meshes.

Rosetta Data Structure

A versatile data structure is the most important part of Rosetta. Various data anomalies were considered as data structure concepts were developed. One problem is that the records may not be input in order (as shown in Figure A.2), but FEA programs often require ordered data in their input files. Another problem is gaps in entity numbering that may result from mesh editing. FEA programs may not accept gaps in entity numbers.

Also, the relationship between quantities of one entity vs. another varies widely, depending on the application. For example, a complex, machined part might contain thousands of nodes and elements and a single material property, while a cast part might have a different material property for each element (properties dependent on cooling curve). The data structure should adapt to either extreme.

A complete data structure will support input and output of an arbitrary number of data formats through subroutines that load or unload records from data files. The purpose of the reformatter software is to simplify the support of any new FEA input file format.

Rosetta uses three arrays, IVAL, RVAL and NAME, to store integer, real, and text data, respectively. Data from each entity type is stored in blocks within these arrays. A directory array (named after the entity keyword, i.e. NODE, ELEM, MATL) points to an associated block in the IVAL array. The IVAL block may contain integer data for the entity and pointers to the RVAL and NAME array (see Figure A.3). Each value in the directory arrays is initialized to -1. Thus, a directory array location with a value of -1 indicates that the associated entity has not been defined.

Node Information:**NODE** (node number) = transformation matrix numberCoordinates, scalar: **X** - Array (4,N)

Node #	X	Y	Z	Scalar
1	x_1	y_1	z_1	s_1
2	x_2	y_2	z_2	s_2
⋮	⋮	⋮	⋮	⋮
N	x_n	y_n	z_n	s_n

Element Information:**ELEM** (element number) = pointer to IVAL block

IVAL element block

+0	Pointer- Elem Name →
+1	IGES element type (+) or Number of Nodes (-NN)
+2	Material Prop ID
+3	Geometric Prop ID
+4	Node Numbers
⋮	⋮
+3-NN	Last Node Number

ENAM Array

1	Elem Nam 1
2	Elem Nam 2
⋮	⋮
	Elem Nam

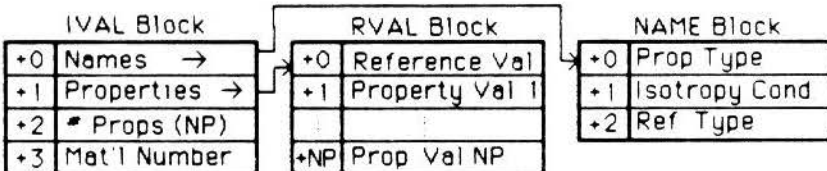
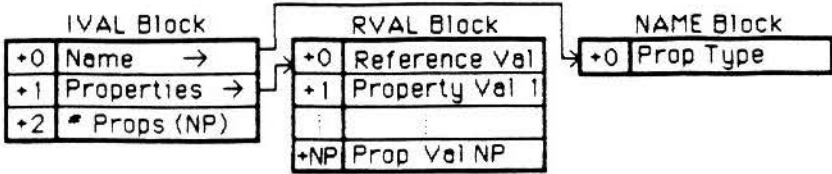
Material Property Information:**MATL** (Mat'l Property Number) = pointer to IVAL block

Figure A.3 Entity Data Blocks in Rosetta Data Structure
(sheet 1 of 3)

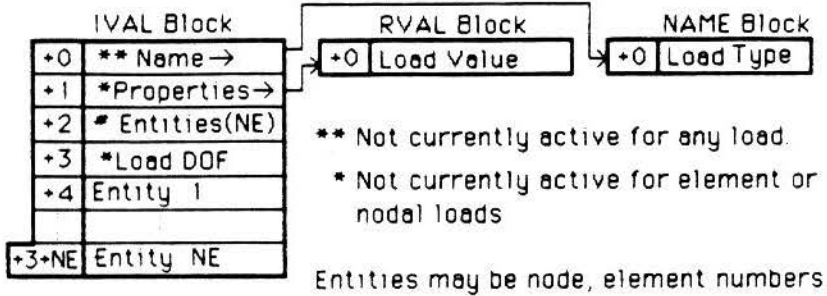
Geometric Property Information

GEOM (Geom. Property Number) = pointer to IVAL block



Element^①, Nodal^②, and Harmonic^③ Load Information:

LOAD (1,2 or 3 , Load Number) = pointer to IVAL block



Transformation Matrix Information:

TRAN (Tran. Matrix Number) = pointer to IVAL block

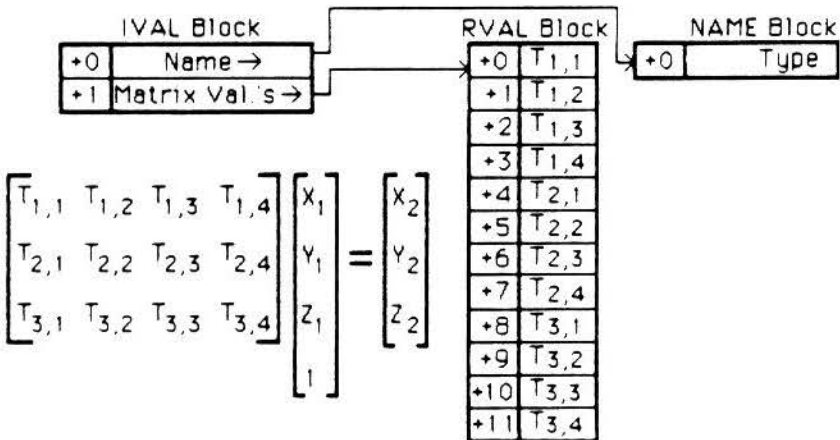
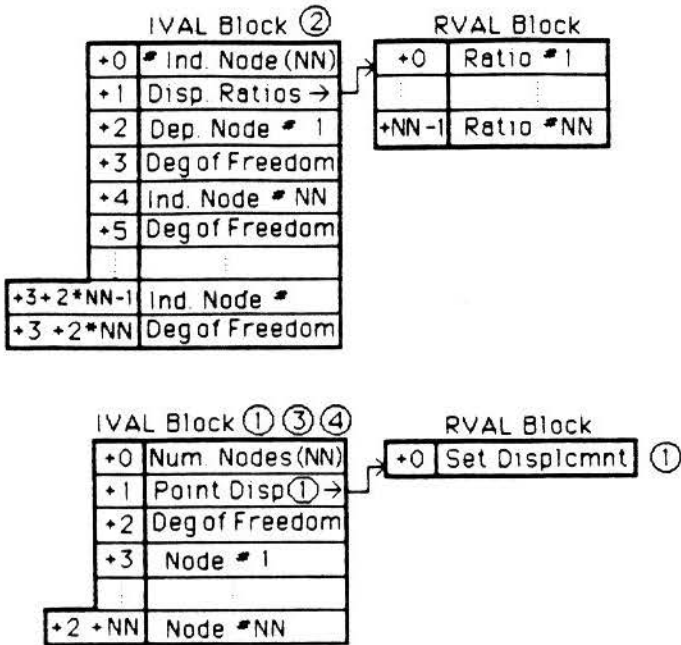


Figure A.3 Entity Data Blocks in Rosetta Data Structure (sheet 2 of 3)

Constraint Information:

Permanent ①, **Multiple** ②, **Omitted DOF** ③, **Retained DOF** ④

CONS (1,2,3,4 , Constraint Number) = pointer to IVAL block

**Element Types Information:**

LELM (Element Type) = Pointer to IVAL Block

ENAM (Element Type) = Name of Element Type

IVAL Block

+0	IGES Topology #
+1	Num Nodes (NN)
+2	Map to Node # 1
⋮	⋮
+1+NN	Map to Node # NN

Figure A.3 Entity Data Blocks in Rosetta Data Structure
(sheet 3 of 3)

The first 42 locations in the IVAL array contain the dimensioned size and the current number of values stored in each array in the data structure (see Figure A.4). This information is used to check for data overflow and to monitor array use.

Entity	Array Name	Size	Current Value	Initial Value
Node Point	NODE	1	21	0
Finite Element	ELEM	2	22	0
Material Property	MATL	3	23	0
Geometric Property	PROP	4	24	0
Element Load	LOAD	5	25	0
Nodal Load			26	0
Harmonic Load			27	0
Permanent Constraint	CONS	6	30	0
Multiple Constraint			31	0
Omitted Degree of Freedom			32	0
Retained Degree of Freedom			33	0
Transformation Matrix	TRAN	7	36	0
Distinct Element Names	ENAM	8	37	0
Element Types (Topology)	LELM			
Integer Value Storage	IVAL	9	38	42
Real Value Storage	RVAL	10	39	0
Text String Storage	NAME	11	40	0
Header Information	HEAD	12	41	0

Figure A.4 IVAL Header Block Contents

Figure A.5 demonstrates a sample application of the data structure, given the neutral data records for geometric property #4 and element #2. When a record is input, the keyword is checked to determine the entity type. The entity number is then read. Since the record is for geometric property #4, the 4th slot in the geometric property directory array, PROP, is set to point to the beginning location of a block in the IVAL array. Since the first 42 values compose the header, the first available location is slot 43. The first value in the IVAL geometric property block points to the slot in array NAME containing the name of the property. The next slot (44) points to the beginning location of a block in the RVAL array that contains the geometric properties. Slot 45 in the IVAL block contains the number of values in the RVAL block.

Neutral File Input

```
PROP,4,BEAMSECT,7.8E7,1.3E6,5.6;
ELEM,2,TRIA,1,4,13,8,7;
```

Resulting Data Structure:

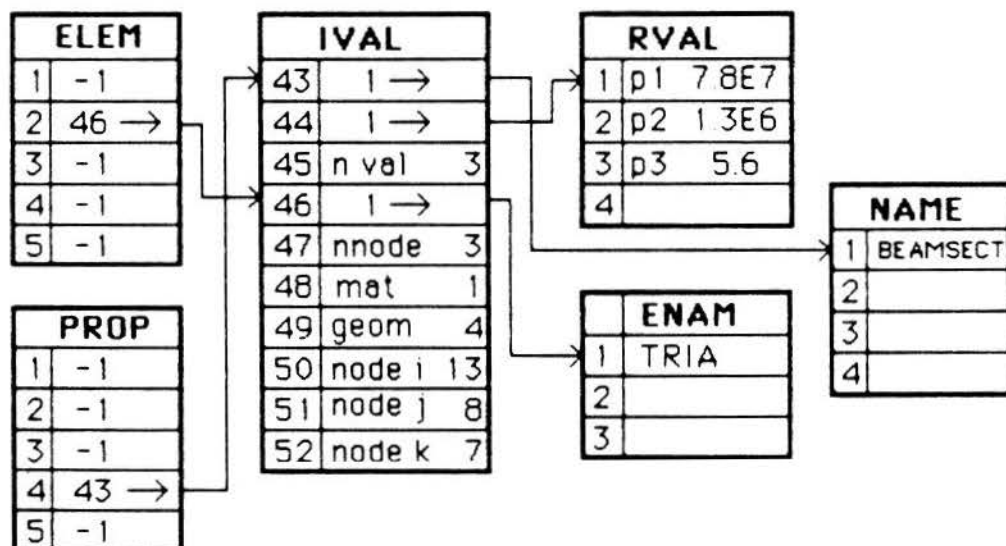


Figure A.5 Sample Application of Rosetta Data Structure

A similar procedure is followed with the element record. The record is interpreted, and found to describe an element with an ID number of 2. A pointer for element #2 is stored in directory array, ELEM, and an element block is added to array IVAL, starting at the first available location, #46. The first slot in the block points to a location in array ENAM, which stores all distinct element types by name. (The element names could have been stored in array NAME, but there were several advantages to localizing the information in ENAM. Each time an element record is read, array ENAM is searched to see if the element name has been previously stored. If the name is not present, it is added to the list. It is easier to keep track of distinct element names, than to store repetitively element names of the same element type.) The second slot of the IVAL block (47) stores the number of nodes in the element. The third and fourth slots store the ID numbers of the material and geometric properties for the element. The remaining slots store the node numbers.

Adding Input/Output Subroutines

Subroutines are easily added to Rosetta to support additional FEA codes and new FEM data standards. Input subroutine names begin with "RED" and output subroutine names begin "WRT" (short for "read" and "write"). The best way to learn how to write an input or output subroutine is to examine existing subroutines in the program. Various utility routines are available to help perform operations with the data structure. A map of Rosetta subroutines is given in Figure A.6, and Table A.2 contains a brief summary of the purpose of each routine. Numerous comments in the program listing explain the algorithms used in Rosetta.

Table A.2 Rosetta Subroutine Description

BANDWD	- Inputs a node cross-reference file to improve bandwidth
BUFOUT	- Outputs a line of element connectivity for Movie files
CHECK	- Calls subroutines for model checking options
CMPDIR	- Compresses a directory array (ELEM, MATL, etc.)
CMPELM	- Compresses element numbering
CMPMAT	- Compresses material property numbering
CMPNOD	- Compresses node numbering
EDCHNG	- Editing: change an existing entity
EDELET	- Editing: delete an entity
EDINSR	- Editing: insert an entity
EDIT	- Call subroutines for editing options
EENLIS	- Editing: Nodes or elements in list (constraints, loads)
ENTNUM	- Request entity number for edit, checks if it exists
FNDELM	- Search element type database file for each new element type.
GETDIR	- Get directory record data from an IGES input file
GETGLB	- Get global record data from an IGES input file
GETPAR	- Get parameter record data from and IGES input file
IDLIST	- Allow user to quickly create list of node, element #'s
LIST	- List entities in the data structure, save list on disk
MISSNG	- List missing entities for a given entity
NEXT	- Input and convert the next field in free format input.
NOMORE	- Checks an entity's ID number to see if it is within range
ORDERE	- Reorders 2-D elements to be CCW as viewed from above.
PACOTI	- Outputs char buffer as IGES parameter rec, blanks removed
PACOUT	- Outputs a character buffer to disk, after removing blanks
POLYOT	- Calculate polygon output to Movie.BYU for 2- and 3-D elements
REDIGS	- Call subroutines to read in a IGES input file
REDMOV	- Input a Movie.BYU geometry file (DISPLAY)
REDNEU	- Read in a Navy Neutral input file
TSPACE	- Check for temporary scratch space in the IVAL array.
WRTIGS	- Write out and IGES data file
WRTMOV	- Write out a Movie.BYU file (2-D:Display and 3-D:Section)
WRTNEU	- Write out a Navy Neutral file
WRTSAP	- Write out a SAP IV input data set
Z*****	- Multi-level prompt routines or edit format routines.

Element Type Data Base

The element names used in the Navy data format match names of elements in the target FEA code. For example, CPS8 is the name of an 8-node quad used for plane stress problems with Abaqus. Rosetta inputs the element name and the number of nodes. Without additional information,

there would be no way to distinguish between an 8-node quad element and an 8-node brick element. Even if the general form of the element could be determined, differences in local element node numbering between FEA programs could be confusing. Some codes order nodes in a counterclockwise order, while others number corner nodes, then mid-side nodes. Node number order is crucial in creating correct Movie.BYU data files.

These problems are solved with an element type data base that stores element names, descriptions, and node numbering information. This database is stored in a direct-access file that is created and maintained using EDBASE.BYU. Rosetta used the direct-access file to recognize element names when data files are input. The format used in the direct access file is shown in Table A.3. The dual set of pointers allow the data base to be searched according to number of nodes in the element (needed for Navy neutral input) or according to the IGES topology type (needed for IGES input).

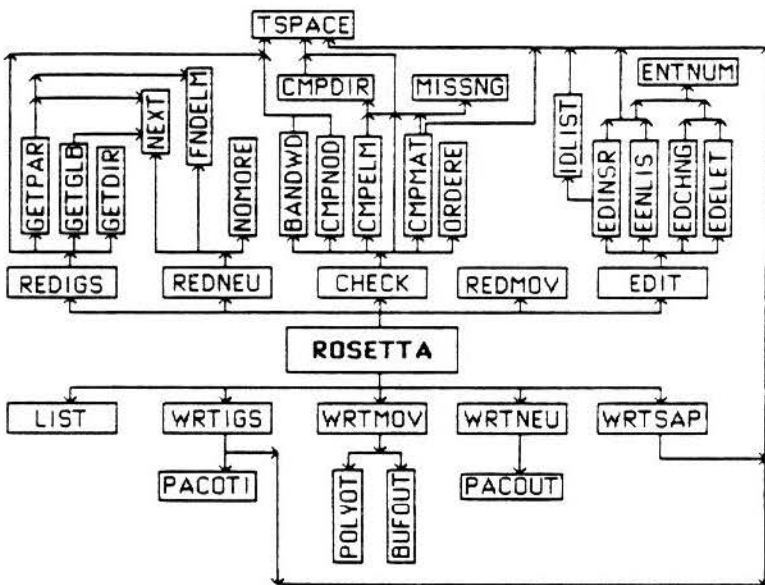


Figure A.6 Rosetta Subroutine Map

Table A.3 - Direct Access Record Formats for Element Type Data

A. Header Records

Record Number 1 - Format(20/4)

Columns	Variable	Description
1- 4	IPOINT(1)	Pointer to element type with 2 nodes
5- 8	IPOINT(2)	Pointer to element type with 3 nodes
9-12	IPOINT(3)	Pointer to element type with 4 nodes
13-16	IPOINT(4)	Pointer to element type with 6 nodes
16-20	IPOINT(5)	Pointer to element type with 8 nodes
21-24	IPOINT(6)	Pointer to element type with 9 nodes
25-28	IPOINT(7)	Pointer to element type with 10 nodes
29-32	IPOINT(8)	Pointer to element type with 12 nodes
33-36	IPOINT(9)	Pointer to element type with 15 nodes
37-40	IPOINT(10)	Pointer to element type with 16 nodes
41-44	IPOINT(11)	Pointer to element type with 18 nodes
45-48	IPOINT(12)	Pointer to element type with 20 nodes
49-52	IPOINT(13)	Pointer to element type with 24 nodes
53-56	IPOINT(14)	Pointer to element type with 32 nodes
57-60	AVAIL	Location of next empty record

Record Number 2,3 - Format(20/4)

Columns	Variable	Description
1-80	JPOINT(1-20)	Pointer to element with IGES topology 1-20
1-24	JPOINT(21-26)	Pointer to element type IGES topology 21-26

B. Element Type Data Records

Record Number 1 - Format(2/4,A15,A57)

Columns	Variable	Description
1- 4	LOCNOD	Pointer to next element with N nodes.
5- 8	LOCIGS	Pointer to next element IGES topology K.
9-23	TEXT(1:15)	Element Name
24-80	TEXT(16:72)	Element Description

Record Number 2 - Format(33/2)

Columns	Variable	Description
1-64	IGSMAP(1-32)	Local element numbers corresponding to IGES topology numbers
65-66	IGEST	IGES topology type

APPENDIX B

DAVINCI DOCUMENTATION

Davinci.BYU

An Engineering Workstation Manager
(User & Programmer Guide)

Evan S. Christensen
Brant A. Ross
Kenneth W. Chase
Terril N. Hurst
Steven E. Benzley

College of Engineering
Brigham Young University
Provo, Utah 84602



Neither Brigham Young University nor its employees makes any warranty expressed or implied, or assumes any legal responsibility for the accuracy, completeness or usefulness of this computer program or its associated documentation.

Table of Contents

	Page
I. USER'S GUIDE INFORMATION	
Program Description	97
Davinci.BYU Options Summary	98
Davinci.BYU Sample Execution	100
Move Commands	102
Information Commands	105
Execution Commands	111
Edit Commands	113
Davinci Option - #	124
Verify Command	124
Quit	124
Initialization of Data Files	125
II. PROGRAMMER INFORMATION	
Program Organization	129
Data File Structure	131
FILMEN Data File	131
ZFILEZ Data File	135

I. USER'S GUIDE INFORMATION

This section of the documentation contains a brief description of the Davinci, Sentinel, and Gateway programs, a summary of the available Davinci options, and a sample execution of the Davinci program. Information regarding the initialization of the FILMEN and ZFILEZ data files is also included in this section. Table 1, which is a listing of the Davinci options, may be used as an index to the sample execution.

Program Description

Davinci.BYU is a new approach to establishing an engineering workstation environment. It is a computer program written in Fortran that helps engineers use computer software more efficiently through classification of programs and data files. The user can easily organize a library of programs into menus interactively. For each program, Davinci will store:

1. The program execution instruction.
2. A brief description of the program.
3. Location of manuals and support people.
4. Location of the source file for the program.

Data files may also be classified using up to six user-defined categories, as well as by name, size, time of creation, and status. Davinci automatically prompts for information on new data files as they are created. It can also store information on backed-up files. A few suggestions for categories that might be used to organize data files follow:

1. By program name - linking each file to the program it is used with: Nastran, Visicalc, etc.
2. By project - linking each file to the appropriate project number, name, or charge code.
3. By product - using a product number or name.
4. By customer - for users provider analysis services.

Two subsets of the Davinci program, Sentinel and Gateway, were created for use in multi-user environments (mainframes, shared minicomputers, etc.). Software shared by a group of computer users would be managed by Sentinel and accessed by Gateway. Data file management features are removed from both of these versions, and edit features appear only in the systems manager's version, Sentinel. Sentinel and Gateway help users share software and aid communication of software status.

Davinci, Sentinel, and Gateway run on VAX/VMS, VAX/Unix, and HP9000/HPUX systems. They require multi-tasking capabilities to be fully operational.

Davinci.BYU Options Summary

Davinci.BYU offers a flexible program structure by allowing the user to select desired options. The various paths the user can take during program execution are shown in Table 1. Although this table does not illustrate all of the requests for user input, it does provide the user with a general idea of the sequence of prompts encountered. As the available options differ between the Davinci, Sentinel, and Gateway programs, options that apply to Davinci only are marked with * and those that apply to Davinci and Sentinel only are marked with **. Table 1 also serves as an index to the sample run which follows. Although three levels of prompt are available with each request for user input, only the first two levels are shown.

Table 1 - DAVINCI OPTIONS SUMMARY AND SAMPLE RUN INDEX

Page	
101	Select DAVINCI option: (m,i,x,e,#,v,q)
102	m - Move through menus - Select MOVE option: (#,u,t,f,q)
102	# - paths to menu number (#)
103	u - paths Up one level
103	t - paths to Top of menu structure
103	f - Find a menu containing a specific program
103	q - Quit, return to DAVINCI options
105	i - menu item Info - Select INFORMATION option: (q,s,r,m,f)
105	q - Quit information, return to DAVINCI options
105	s - types Synopsis of menu item
105	r - types external Reference summary for item
106	m - lists Menu structure with indentations
107	f - lists data File Information - Select LIST option: (q,a,c)*
107	q - Quit file list option
107	a - ALL files to the screen
108	c - by Categories - Select SORT option: (n,t,b,s,#,q)
108	n - Name of data file
109	t - Time of creation
110	b - Block size of data file
110	s - Status (Backed-up, Online, and Temporary)
111	# - User Category number 1-6: (user-defined)
111	q - Quit, to abort or complete SORT options
111	x - eXecute a program - Enter program NUMBER or NAME
111	# - menu number of program to be executed
112	name - program's code name
112	s - enter your own System command
112	q - Quit execute request
113	e - Edit - Select EDIT TYPE option: (q,f,c,l,r,o,m)**
113	q - Quit edit option
120	f - edit a single data File specified by name*
121	c - edit data file Categories*
123	l - edit the Library title
123	r - Restore a program in the menu on-line
123	o - save a program in the menu Off-line
113	m - edit Menus - Select EDIT option: (q,l,c,d,i)
113	q - Quit edit of a menu
119	l - List information on menu item
118	c - Change information on a menu item
119	d - Delete a menu item
113	i - Insert a menu item - Select MENU TYPE option: (q,c,p)
124	# - Move to category item or execute a program item
124	v - Verify data file information, categorize new files*
124	q - Quit DAVINCI program
125	Initialization of Data Files*

Davinci.BYU Sample Execution

This section contains a step by step execution of the Davinci, Sentinel, and Gateway programs. Davinci is used to illustrate those characteristics common to all three programs. The capabilities of the Squire.BYU library are demonstrated, which include multi-level prompts and buffering of commands. The first level prompt is terse for the expert user, the second level is a fairly complete description, and the third level suggests a response. The second and third levels are summoned by simply hitting the RETURN key as stated. In addition to the three levels of information on each prompt, prompts can also be answered ahead by buffering up the appropriate commands. The more familiar the user becomes with the sequence of prompts, the better able he will be to anticipate the next request for input and answer ahead. When commands are buffered, the corresponding prompts are suppressed. This ability to buffer up commands by placing the appropriate responses on a single line separated by spaces allows the experienced user to move quickly through the program.

Comments shown in *italics* do not appear on the terminal screen but have been added here for description purposes. User input is shown in **bold** to help the reader understand the sample execution. The menu data file, FILMEN, was taken from an industry application of Davinci. The data file ZFILEZ was created using a sample directory.

\$ davinci

```

*****
* DAVINCI.BYU, the engineering workstation manager
* - Brigham Young University, College of Engineering -
*
* Note A: At any time you can...
* 1) Hit the RETURN key to get more information.
* 2) Answer questions ahead by entering the corresponding
*    answers, separated by spaces.
*
* Note B: DAVINCI is organized into menus containing num-
* bered items, each of which is either a CATEGORY or PROGRAM.
* 1) CATEGORIES may contain a list of new menu items which
*    are accessed using the m (Move) command.
* 2) PROGRAMS are executed using the x (eXecute) command.
* Each CATEGORY or PROGRAM can also be accessed by simply
* typing its item #, followed by a carriage return.
*****

```

Hit <RETURN> to continue

When the RETURN key is hit to continue, the screen is cleared and the top menu of the program library will appear with a prompt to select a Davinci, Sentinel, or Gateway option. When Davinci and Sentinel are executed for the first time, the user will be prompted for information used to initialize the FILMEN and ZFILEZ data files. (See the section titled "Initialization of Data Files".)

HP 9000 Brigham Young University Software Library

1. Movie.BYU Graphics Package
2. OPTDES.BYU Design Optimization Package
3. Finite Element Modeling
4. Lumped-Mass Vibration/Modal Analysis
5. State-Space Control Systems Tools

Select DAVINCI option: (m,i,x,e,#,v,q)

<Return>

m - Move through menus

i - Information on menu items

x - eXecute a program by menu item number or by name

e - Edit menus, or library title

- select item # (executes program, OR moves to category)

v - Verify data file info, categorize new files

q - Quit DAVINCI program

<Return>

Why not enter: m to Move through menus, NO further help!

If the RETURN key is hit following a third level prompt, the user is told who to see or where to go for additional help and the program is terminated (not shown here). Let's first investigate the move options.

MOVE COMMANDS

The Davinci library is set up such that each menu item is either a program or a category heading to a submenu. The MOVE options are used to move through the menus and apply only to category menu items.

HP 9000 Brigham Young University Software Library

1. Movie.BYU Graphics Package
2. OPTDES.BYU Design Optimization Package
3. Finite Element Modeling
4. Lumped-Mass Vibration/Modal Analysis
5. State-Space Control Systems Tools

Select DAVINCI option: (m,i,x,e,#,v,q)

m

Select MOVE option: (#,u,t,f,q)

<Return>

- move to menu number (#)

u - move Up one menu level

t - move to Top of menu structure

f - Find menu containing (name)

q - Quit, return to main options

MOVE TO ITEM NUMBER

1 *Let's move to item #1, Movie.BYU Graphics Package.*

Movie.BYU Graphics Package

1. Movie.BYU Software
2. Documentation
3. Demos

Select DAVINCI option: (m,i,x,e,#,v,q)

The Davinci option # can also be used to move to a category menu item by entering the menu item number without the MOVE command m. Let's move to item 1 with this option:

1

Movie.BYU Software

1. DISPLAY - line drawings & continuous color output (DISPLAY)
2. UTILITY - create or edit Movie data files (UTILITY)
3. SECTION - clips and caps 3-D finite element models (SECTION)
4. COMPOSE - produce multiple image line drawings (COMPOSE)
5. TITLE - generate polygon-based characters for use in DISPLAY (TITLE)
6. MOSAIC - converts contour lines into polygon mosaics (MOSAIC)

Select DAVINCI option: (m,i,x,e,#,v,q)

MOVE UP

Now, let's try the MOVE option u and move back to the previous menu, "Movie.BYU Graphics Package." We will make this move by buffering the commands as described earlier.

m u

Movie.BYU Graphics Package

1. Movie.BYU Software
2. Documentation
3. Demos

Select DAVINCI option: (m,i,x,e,#,v,q)

As you can see, when commands are buffered, the prompts to the buffered commands are suppressed. This allows the experienced user to move quickly through the menus.

MOVE TO TOP

To move to the top menu, HP 9000 Brigham Young University Software Library, from any location in the library, use MOVE option t.

m t

Movie.BYU Software

1. DISPLAY - line drawings & continuous color output (DISPLAY)
2. UTILITY - create or edit Movie data files (UTILITY)
3. SECTION - clips and caps 3-D finite element models (SECTION)
4. COMPOSE - produce multiple image line drawings (COMPOSE)
5. TITLE - generate polygon-based characters for use in DISPLAY (TITLE)
6. MOSAIC - converts contour lines into polygon mosaics (MOSAIC)

Select DAVINCI option: (m,i,x,e,#,v,q)

FIND A PROGRAM

The find command searches for and jumps to the menu containing a particular program (specified its keyword). For example, let's try to find a menu containing program DISPLAY.

m f display

Movie.BYU Software

1. DISPLAY - line drawings & continuous color output (DISPLAY)
2. UTILITY - create or edit Movie data files (UTILITY)
3. SECTION - clips and caps 3-D finite element models (SECTION)
4. COMPOSE - produce multiple image line drawings (COMPOSE)
5. TITLE - generate polygon-based characters for use in DISPLAY (TITLE)
6. MOSAIC - converts contour lines into polygon mosaics (MOSAIC)

Select MOVE option: (#,u,t,f,q)

To search for another menu containing this program, the user can enter the find command f and hit return. Otherwise, he may search for a different program by entering its name or choose any of the other MOVE options.

f

Finite Element Modeling

1. SAP IV finite element analysis program (SAPIV)
2. Coyote.BYU conduction heat transfer FEM program (COYOTE)
3. Chiles.BYU 2-D fracture mechanics FEM analysis program (CHILES)
4. Rosetta.BYU FEM data reformatter (ROSETTA)
5. DISPLAY - line drawings & continuous color output (DISPLAY)

Select MOVE option: (#,u,t,f,q)

f

Program: DISPLAY was not found

Hit <RETURN> to continue

<Return>

Finite Element Modeling

1. SAP IV finite element analysis program (SAPIV)
2. Coyote.BYU conduction heat transfer FEM program (COYOTE)
3. Chiles.BYU 2-D fracture mechanics FEM analysis program (CHILES)
4. Rosetta.BYU FEM data reformatter (ROSETTA)
5. DISPLAY - line drawings & continuous color output (DISPLAY)

Select DAVINCI option: (m,i,x,e,#,v,q)

An attempt to move to a program menu item produces:

m 2

Error, Not a Category...

Select DAVINCI option: (m,i,x,e,#,v,q)

Similar error messages appear at other places in the program when the user gives an inappropriate response.

INFORMATION COMMANDS

HP 9000 Brigham Young University Software Library

1. Movie.BYU Graphics Package
2. OPTDES.BYU Design Optimization Package
3. Finite Element Modeling
4. Lumped-Mass Vibration/Modal Analysis
5. State-Space Control Systems Tools

Select DAVINCI option: (m,i,x,e,#,v,q)

i

Select INFORMATION option: (s,r,m,f,q)

<Return>

- s - give Synopsis of menu item (#)
- r - list References for additional help for item (#)
- m - display Menu structure with indentations
- f - lists data file information
- q - Quit information, return to DAVINCI options

SYNOPSIS OF MENU ITEM

All menu items, whether categories or programs, have a brief description. It can be seen by entering s, synopsis of a menu item, and the item number. Let's look at the synopsis of item 3, Finite Element Modeling.

s 3

Finite Element Modeling

This category contains software used in the finite element modeling process: preprocessing, analysis, and post processing

Hit <RETURN> to continue

REFERENCE ON PROGRAM ITEM

The reference option r (program items only) contains the location of manuals, documents, or personnel the user can see for help. For example, the reference for item 2 is:

Movie.BYU Software

1. DISPLAY - line drawings & continuous color output (DISPLAY)
2. UTILITY - create or edit Movie data files (UTILITY)
3. SECTION - clips and caps 3-D finite element models (SECTION)
4. COMPOSE - produce multiple image line drawings (COMPOSE)
5. TITLE - generate polygon-based characters for use in DISPLAY (TITLE)
6. MOSAIC - converts contour lines into polygon mosaics (MOSAIC)

Select DAVINCI option: (m,i,x,e,#,v,q)

i r 2

UTILITY - create or edit Movie data files

Refer to the manual in room 318 CB or see Richard Street in room 289 CB.

Hit <RETURN> to continue

LISTING OF MENU STRUCTURE

INFORMATION option m produces an indented listing of the entire library menu structure, which may be saved to a file. Since the menu listing is long, only part will be shown.

Select DAVINCI option: (m,i,x,e,#,v,q)

i m

Write a copy of this list to a file? (y/n)

n

HP 9000 Brigham Young University Software Library

1. Movie.BYU Graphics Package

The MOVIE system of general purpose computer graphics programs facilitate the display of three-dimensional, topological, and architectural models as line drawings or as continuous tone shaded images. This software also provides the capability to clip and cap three dimensional systems; modify geometry, displacement, and/or scalar function files; generate new models or title representations; and convert contour line definitions into polygonal element mosaics.

1. Movie.BYU Software

This category contain the various program that form Movie.BYU: Display, Utility, Section, Compose, Title, and Mosaic

1. DISPLAY - line drawings & continuous color output (DISPLAY)

DISPLAY is an interactive program for the display and animation of any model composed of polygons. The program allows the user to manipulate the model (rotate, translate, etc.), specify colors for the background and the different element parts, and select the display device.

Execution Command

/users/terril/Dmovie/display

External References

See the documentation section and demo section of the MOVIE menu.

2. UTILITY - create or edit Movie data files (UTILITY)

UTILITY creates or edits Fortran data files in a format compatible with the other programs in the MOVIE system. The program allows the user to specify commands to make data files using the model generation

and transform capabilities, to read, write, or change data files, to perform symmetry operations, to order polygon data consistently, to gather ordered panel data into parts for smoothing, to merge or re-organize data files, to facilitate the display of functions of two variables and surfaces of functions of three variables, or exit UTILITY.

Execution Command

/users/terril/Dmovie/utility

External References

Refer to the documentation section and demos section of the MOVIE menu.

.....

Hit <RETURN> to continue

LISTING OF DATA FILES

The user can list ALL data files or data files falling within certain category limits. Data file listings may be obtained at any level in the library and may be saved to a file.

Select DAYINCI option: (m,i,x,e,#,v,q)

i f

Select LIST option: (a,c,q)

<Return>

- a - list out ALL files to the screen
- c - list files within limits of certain Categories
- q - Quit information request and return to main options

a

Write a copy of this list to a file? (y/n)

n

File: Dfortran/ Created: 9/17/ 12:33
 Size: 2736 Status: On-line
 User Cat #1: Related Analysis Program = SAP IV
 User Cat #2: Project Number, Name, or Charge Code = Wing 3.7
 User Cat #3: Customer Name = Internal

File: Disp11 Created: 9/18/ 17:24
 Size: 2409 Status: On-line
 User Cat #1: Related Analysis Program = SAP IV
 User Cat #2: Project Number, Name, or Charge Code = Wing 3.7
 User Cat #3: Customer Name = Boeing

File: Disp12 Created: 9/18/ 17:24
 Size: 2409 Status: On-line
 User Cat #1: Related Analysis Program = SAP IV
 User Cat #2: Project Number, Name, or Charge Code = Wing 3.7
 User Cat #3: Customer Name = Boeing

Hit <RETURN> to continue

<Return>

Select LIST option: (a,c,q)

C

Select SORT option: (n,t,b,s,#,q)

<Return>

- n - Name of data file
- t - Time of creation
- b - Block size of data file
- s - Status (Backed-up, Online, and Temporary)
- 1 - User Category: Related Analysis Program
- 2 - User Category: Project Number, Name, or Charge Code
- 3 - User Category: Customer Name
- q - Quit (if q is entered first time around, the SORT option is aborted. Otherwise q completes the SORT option selections)

Selection of the LIST option c allows the user to sort of data files to be listed, by setting the upper and lower limits (numerically or alphabetically) of the category(s) used in sorting. When more than one category is used to sort, only data files satisfying all of the category limits will be listed. Although Davinci automatically classifies every data file by the first four categories, data file name, time of creation, size, and status, others are user-defined. For more information on how these user-defined categories are set up and edited, see sections: "Edit Commands" and "Initialization of Data Files."

To create a listing of data files that start with the letter s, set the minimum value to sa and the maximum value to sz for the file name category.

n

Enter the minimum value for this category:

<Return>

- Enter: (value) - the minimum value (i.e. name, time of creation, file block size, etc.) sets the lower limit numerically or alphabetically for the data file search.
 - q - Quit, leaves this value blank
-

sa

Enter the maximum value for this category:

SZ

Select SORT option: (n,t,b,s,#,q)

The user is returned to the SORT options in order to select another category to sort on if he desires. Since we are only searching on the file name, we will enter q to quit.

q

Write a copy of this list to a file? (y/n)

n

```
File: sapin.dat           Created: 8/27/ 16:15
Size: 611                Status: On-line
User Cat #1: Related Analysis Program = SAP IV
User Cat #2: Project Number , Name, or Charge Code = Therm Stress
User Cat #3: Customer Name = Thiokol
```

```
File: sapiv*            Created: 9/18/ 17:21
Size: 446228            Status: On-line
User Cat #1: Related Analysis Program = SAP IV
User Cat #2: Project Number , Name, or Charge Code = Internal
User Cat #3: Customer Name = Internal
```

```
File: sapiv.f           Created: 8/28/ 13:26
Size: 401612            Status: On-line
User Cat #1: Related Analysis Program = SAP IV
User Cat #2: Project Number , Name, or Charge Code = Internal
User Cat #3: Customer Name = Internal
```

Hit <RETURN> to continue

Next, let's list files created during September, are between 1 and 20000 bytes long, and relate to program Movie.BYU.

Select LIST option: (a,c,q)

c t

_____ Enter the MINIMUM time: (MM/DD/YY HH:MM)

<Return>

Enter: (MM/DD/YY HH:MM) - use 14 characters to specify the earliest time, by month, day, year, hour and minute.
 enter a space instead of 0 where values are less than 10, include the / and : marks.

q - Quit, sets date to Jan 1, 1900

9/ 1/ 1:01

_____ Enter the MAXIMUM time: (MM/DD/YY HH:MM)

9/30/ 23:59

Select SORT option: (n,t,b,s,#,q)

b

Enter the minimum value for this category:

1

Enter the maximum value for this category:

20000

Select SORT option: (n,t,b,s,#,q)

1

Enter the minimum value for this category:

Movie.BYU

Enter the maximum value for this category:

Movie.BYU

Write a copy of this list to a file? (y/n)

n

File: GEOM.DAT

Created: 9/18/ 17:23

Size: 3839

Status: On-line

User Cat #1: Related Analysis Program = Movie.BYU

User Cat #2: Project Number, Name, or Charge Code = Wing 3.7

User Cat #3: Customer Name = Boeing

Hit <RETURN> to continue

EXECUTION COMMANDS

Programs can be executed: 1) by item number, 2) by program name, or 3) by a system command.

HP 9000 Brigham Young University Software Library

1. Movie.BYU Graphics Package
2. OPTDES.BYU Design Optimization Package
3. Finite Element Modeling
4. Lumped-Mass Vibration/Modal Analysis
5. State-Space Control Systems Tools

Select DAVINCI option: (m,i,x,e,#,v,q)

x

_____ Enter the program NUMBER or NAME to execute:

<Return>

Enter: ITEM # - menu number of program to be executed

(name) - program's description code name

S - enter your own System command

q - Quit execute request

EXECUTE BY ITEM NUMBER

First move to a menu containing the program. Next, enter the DAVINCI option, x, and the menu item number, or simply enter the item number. For example, to run DISPLAY:

Select DAVINCI option: (m,i,x,e,#,v,q)

m f display

Movie.BYU Software

1. DISPLAY - line drawings & continuous color output (DISPLAY)
2. UTILITY - create or edit Movie data files (UTILITY)
3. SECTION - clips and caps 3-D finite element models (SECTION)
4. COMPOSE - produce multiple image line drawings (COMPOSE)
5. TITLE - generate polygon-based characters for use in DISPLAY (TITLE)
6. MOSAIC - converts contour lines into polygon mosaics (MOSAIC)

Select MOVE option: (#,u,t,f,q)

q x 1

The following command is used to run program: DISPLAY

/users/terrill/Dmovie/display

<MOVIE SYSTEM DISPLAY>

<READ GEOM FILE>

Hit <RETURN> to continue

EXECUTE BY PROGRAM NAME

A program can be executed by name from any menu in the library. The user enters x and the program name. Let's move back to the top menu and execute DISPLAY.

HP 9000 Brigham Young University Software Library

1. Movie.BYU Graphics Package
2. OPTDES.BYU Design Optimization Package
3. Finite Element Modeling
4. Lumped-Mass Vibration/Modal Analysis
5. State-Space Control Systems Tools

Select DAVINCI option: (m,i,x,e,#,v,q)

x display

The following command is used to run program: DISPLAY
/users/terril/Dmovie/display

<MOVIE SYSTEM DISPLAY>

<READ GEOM FILE>

.....

Hit <RETURN> to continue

EXECUTE BY SYSTEM COMMAND

Any system command may be run from Davinci: running programs, editing files, or communicating with other users.

HP 9000 Brigham Young University Software Library

1. Movie.BYU Graphics Package
2. OPTDES.BYU Design Optimization Package
3. Finite Element Modeling
4. Lumped-Mass Vibration/Modal Analysis
5. State-Space Control Systems Tools

Select DAVINCI option: (m,i,x,e,#,v,q)

x s /users/terril/Dmovie/display

The following command is used to run program: DISPLAY
/users/terril/Dmovie/display

<MOVIE SYSTEM DISPLAY>

<READ GEOM FILE>

.....

Hit <RETURN> to continue

EDIT COMMANDS (Davinci and Sentinel)

Select DAVINCI option: (m,i,x,e,#,v,q)

e

Select EDIT TYPE option: (m,f,c,l,r,o,q)

<Return>

m - edit Menus

f - edit a single data File*

c - edit data file Categories*

l - edit the Library title

r - Restore a program on-line

o - set program status: Off-line

q - Quit edit and return to DAVINCI options

**Used by the Davinci program only.*

EDIT A MENU ITEM

Only items in the current menu may be edited. Let's insert a new item (category) into the current menu.

m

Enter EDIT command: (c,d,i,l,q)

<Return>

c - change

d - delete

i - insert

l - list data

q - quit edit

i

Insert after which item:

<Return>

ENTER: Value- Insert after this item

0 - Insert before first item

q - Quit insertion

A new item may be inserted before or after any existing item. Let's insert a category, Dynamic Analysis Package, after item 5.

5

Select MENU TYPE option: (c,p,q)

<Return>

c - Category, a heading to a menu below this menu

p - Program, a program or procedure you can run

q - Quit edit, return to DAVINCI options

c

Enter the TITLE of this item for the menu.

Dashes show the maximum number of characters for text.

<Return>

Enter: (Text) - enter up to 72 characters per line,
then hit return. Enter up to 10 lines.

q - Quit, information is left blank if "q" is
entered on first line.

Dynamic Analysis Pckage

q

Edit TITLE of Item

1) Dynamic Analysis Pckage
This data OK? (y,n)

*Davinci echoes recently input data to the screen for user review
and necessary revision. Input errors may be corrected at once.
Since Pckage is misspelled, we will answer n to correct it.*

n

Enter EDIT command: (c,d,i,l,q)

c 1

1) Dynamic Analysis Pckage
Enter new TEXT:

Dynamic Analysis Package

Edit TITLE of Item

1) Dynamic Analysis Package
Enter EDIT command: (c,d,i,l,q)

q

Enter a BRIEF DESCRIPTION of this item.

This package of dynamic analysis programs handles harmonic
response, the duhamel integral, fourier series, fast fourier

transform, elastoplastic behavior, frequencies and mode
 shapes, and response of elastic shear buildings.

q

Edit BRIEF DESCRIPTION of Item

- 1) This package of dynamic analysis programs handles harmonic
- 2) response, the duhamel integral, fourier series, fast fourier
- 3) transform, elastoplastic behavior, frequencies and mode
- 4) shapes, and response of elastic shear buildings.

This data OK? (y,n)'

y

HP 9000 Brigham Young University Software Library

1. Movie.BYU Graphics Package
2. OPTDES.BYU Design Optimization Package
3. Finite Element Modeling
4. Lumped-Mass Vibration/Modal Analysis
5. State-Space Control Systems Tools
6. Dynamic Analysis Package

Select DAVINCI option: (m,i,x,e,#,v,q)

The new category appears as item 6 in the menu. Now, let's move to the new menu titled, Dynamic Analysis Package.

m 6

Dynamic Analysis Package

Select DAVINCI option: (m,i,x,e,#,v,q)

The new menu is empty. Let's insert program: MODS

e m i 0 p

_____ Enter the Program Code Name:

<Return>

Enter: (code) - use up to 9-characters for a code name of this menu item.

Menus sharing a common program should use the same code.

q - Quit, description code left blank

The program code name (usually its disk file name) is used to distinguish between programs, to search for a menu containing a particular program (See "Find a Program") and to check for other occurrences of the program. If another program uses this name, the user must select a different code name. A program that occurs several times in the menu should always use the same code name, for consistency. Let's enter code name MODS.

mods

EDIT OF PROGRAM DESCRIPTION CODE

1) MODS

This data OK? (y,n)

y

Searching for other occurrences of program: MODS

Adding program (MODS) to the menu...

Apparently, no other menu contains a program named MODS.

Enter the TITLE of this item for the menu.

MODS - A Dynamic Analysis Program

q

Edit TITLE of Item

1) MODS - A Dynamic Analysis Program

This data OK? (y,n)

y

Enter a BRIEF DESCRIPTION of this item.

MODS is a dynamic analysis program for problems in:

(a) HARMONIC RESPONSE (SDOF)

(b) DUHAMEL INTEGRAL - DAMPED (SDOF)

(c) FOURIER SERIES - DAMPED (SDOF)

(d) FAST FOURIER TRANSFORM - DAMPED (SDOF)

(e) STEPS - ELASTOPLASTIC BEHAVIOR (SDOF)

(f) JACOBI - FREQUENCIES AND MODE SHAPES

(g) SRESB - RESPONSE OF ELASTIC SHEAR BLDG.

q

Edit BRIEF DESCRIPTION of Item

- 1) MODS is a dynamic analysis program for problems in:
- 2) (a) HARMONIC RESPONSE (SDOF)
- 3) (b) DUHAMEL INTEGRAL - DAMPED (SDOF)
- 4) (c) FOURIER SERIES - DAMPED (SDOF)
- 5) (d) FAST FOURIER TRANSFORM - DAMPED (SDOF)
- 6) (e) STEPS - ELASTOPLASTIC BEHAVIOR (SDOF)
- 7) (f) JACOBI - FREQUENCIES AND MODE SHAPES
- 8) (g) SRESB - RESPONSE OF ELASTIC SHEAR BLDG.

This data OK? (y,n)

y

Enter a summary of EXTERNAL INFORMATION that pertains to this program, including user manuals and reference files.

See Rick Balling in room 370 CB.

q

Edit EXTERNAL INFORMATION of Item

- 1) See Rick Balling in room 370 CB.

This data OK? (y,n)

y

Enter the SOURCE CODE location:

/users/ce/rickb

Enter the EXECUTION INSTRUCTION:

/users/ce/rickb/mods

Edit of SOURCE CODE LOCATION Description
and Program EXECUTION INSTRUCTION

- 1) /users/ce/rickb
- 2) /users/ce/rickb/mods

This data OK? (y,n)

y

Dynamic Analysis Package

1. MODS - A Dynamic Analysis Program (MODS)

Select DAYINCL option: (m,i,x,e,#,v,q)

Now, let's move to the top menu and change an existing item.

m t e m c 6

EDIT OF CATEGORY INFORMATION

1) Item Title (first line only):

Dynamic Analysis Package

2) Brief Description of Item: (first line only)

This package of dynamic analysis programs handles harmonic response

Change WHICH item:

Only the first line of multi-line information items appear in the list, but all lines appear when editing. Let's change the title of this item to Dynamic Analysis Software.

1

Edit TITLE of Item

- 1) Dynamic Analysis Package

This data OK? (y,n)

n

Edit Command: (c,l,q)

c 1

- 1) Dynamic Analysis Package

Enter new TEXT:

Dynamic Analysis Software

If we had entered q, the title would have remained unchanged.

Edit TITLE of Item

1) Dynamic Analysis Software

Edit Command: (c,l,q)

q

EDIT OF CATEGORY INFORMATION

1) Item Title (first line only):

Dynamic Analysis Software

2) Brief Description of Item: (first line only)

This package of dynamic analysis programs handles harmonic response

Change WHICH item:

We could change another item, but instead we will enter q to quit, and use EDIT command d to delete an item in the top menu.

q d

Delete WHICH item?

6

Delete this item? (y/n)

Davinci makes sure that the user wants to delete this item. This requires an additional response, but helps protect against accidental deletions. The user may not delete a category that still contains items. We enter n to avoid deleting this item.

n

Enter EDIT command: (c,d,i,l,q)

The list command give the same listing as the change command.

1 6

LISTING OF CATEGORY INFORMATION

1) Item Title (first line only):

Dynamic Analysis Software

2) Brief Description of Item: (first line only)

This package of dynamic analysis programs handles harmonic response

Hit <RETURN> to continue

EDIT A DATA FILE

This option is available only with the Davinci program. The other programs, Sentinel and Gateway, do not categorize data files. When the user enters the command to edit a data file, he will be prompted for the name of the file as follows:

Select DAVINCI option: (m,i,x,e,#,v,q)

e f

_____ Enter data file name (15 CHARS MAX)

<Return>

- Enter: (name) - Enter the name of the data file to be worked with. The name can be up to 8 characters in length. The first character must be a letter (A-Z). The other characters can be letters (A-Z), numbers (0-9), or symbols \$, #, @.
- l - List out all current data file names on record
 - q - Quit edit and return to main options

l

Dfortran/, Disp11, Disp12, Disp13, Dobject/, GEOM.DAT, filmen, prob3.2.in, prob3.2.out, prob6.4.in, prob6.4.out, prob6.4.out2, sapin.dat, sapiv*, sapiv.f, xxfile, zfilez

_____ Enter data file name (15 CHARS MAX)

Let's edit the information on the data file GEOM.DAT.

GEOM.DAT

Edit of DATA FILE

File Name: GEOM.DAT

Status: 1) 0

Backed-Up Location: 2) (Does NOT Apply!)

User Category #1: 3) Related Analysis Program = Movie.BYU

User Category #2: 4) Project Number, Name, or Charge Code = Wing 3.7

User Category #3: 5) Customer Name = Boeing

This data OK? (y,n)

Data files are classified according to name, time of creation, size, status and the six possible user-defined categories. Because the name, time of creation, and size are set values, they do not appear in the above listing as items to be edited.

The backed-up location information only applies to files whose status is B, backed-up. When the status is O-on-line, or T-temporary, the message "(does not apply)" is written. Let's change the status from on-line to temporary.

n

Edit Command: (c,l,q)

c l

1) 0

Enter new TEXT:

-
t

Edit of DATA FILE

File Name: GEOM.DAT

Status: 1) T

Backed-Up Location: 2) (Does NOT Apply!)

Edit Command: (c,l,q)

Data files whose status is temporary are not classified by the user-defined categories. However, when a previously unclassified data file whose status was temporary is changed to on-line or backed-up, the user-defined category labels will appear and the user can edit this information. If an attempt is made to change the status to something other than O-on-line, T-temporary, or B-backed-up, the status will be set to a default value of T-temporary. Let's change the status back to on-line.

c l o q

EDIT USER-DEFINED CATEGORY LABELS

Davinci allows up to six user categories for classifying data files (See "Initialization of Data Files.") These categories or category labels are set up when Davinci is run for the first time. The user may redefine these category labels using the change, delete, and insert commands. Let's insert a new category label after category 2.

Select DAVINCI option: (m,i,x,e,#,v,q)

e c

Edit CATEGORIES for Data File

- 1) CATEGORY: Related Analysis Program
- 2) CATEGORY: Project Number , Name, or Charge Code
- 3) CATEGORY: Customer Name

This data OK? (y,n)

n

Enter EDIT command: (c,d,i,l,q)

i 2

Insert after item # 2) Project Number , Name, or Charge Code

Enter new TEXT:

Product Number or Name

Edit CATEGORIES for Data File

- 1) CATEGORY: Related Analysis Program
- 2) CATEGORY: Project Number , Name, or Charge Code
- 3) CATEGORY: Product Number or Name
- 4) CATEGORY: Customer Name

Enter EDIT command: (c,d,i,l,q)

q

How many character are needed for category #3

(Product Number or Name)

25 columns are available!

Enter VALUE:

Davinci will prompt the user for the number of columns needed to store data for each new category. Next, Davinci will prompt the user for the information on each category for each data file. Up to this point, 47 of the 72 columns available for the user-defined category information are being used. We will use 15 more columns of the remaining 25 for this category.

Please Enter Info for Category #3
 (Product Number or Name)
 For each of the following data files:

Category Info for Data File: Dfortran/

XP 34.900

Category Info for Data File: Disp11

XP 55.433

EDIT OF LIBRARY TITLE

The library title is the title of the top menu. The procedure for editing this title is the same as changing the information on a menu item or a user-defined category label.

EDIT OF PROGRAM STATUS

A program may have on-line or off-line status. The status is changed by using the EDIT TYPE options r and o. The option r is used to restore a previously backed-up program on-line and the option o is used to change the status to off-line when a program has been backed-up. To edit the program status using the r and o commands, the user must first move to the menu containing that program. Let's move to the menu containing program MODS and try out these commands.

Select DAVINCI option: (m,i,x,e,#,v,q)

m 6 e o

Save WHICH item Off-line:

1

MODS - A dynamic analysis program

See Rick Balling in room 370 CB.

Source Code Location: /users/rickb/Dmods

The tape drive is located in room 317CB. Use the TAPECOPY command to work with foreign tapes and the COPY command to work with VAX ANSI tapes. Hit <RETURN> to continue

Each time the program status is changed, the user is given reference information, the program source code location, and instructions for handling tapes.

DAVINCI OPTION - *

When the user inputs a category item number, Davinci moves to the menu of that category. When a program item number is entered, Davinci executes the program. This option is shown in sections "Move to Item Number" and "Execute By Item Number."

VERIFY COMMAND (Davinci)

When the user first executes Davinci, new data files are classified automatically. This option provides the user the ability to verify data file information and categorized new data files as they are created during the execution of programs contained in the Davinci library. The procedure for classifying new files using this option is the same as that described in the section titled "ZFILEZ Data Files."

QUIT

The Davinci option q is used to quit Davinci execution. At other input levels, the quit option q is used to quit an option or leave information blank. Let's enter q to terminate the Davinci program.

Dynamic Analysis Software

1. MODS - A Dynamic Analysis Program (MODS)

Select DAVINCI option: (m,i,x,e,#,v,q)

q
FORTRAN STOP
\$

INITIALIZATION OF DATA FILES

The first time Davinci and Sentinel are executed, the user will be prompted for information needed to set up data files ZFILEZ and FILMEN. This initialization occurs after the program header is displayed.

ZFILEZ DATA FILE

File ZFILEZ (Davinci only) stores information used to classify data files. A short description of how data files are categorized is given and the user is prompted for the first user-defined category label.

DAVINCI manages both program and data files. Data files are categorized by name, creation time, and size automatically. You may also specify up to six categories of your own to help organize and keep track of your files. Use all 6 categories if you like, but remember that each time a file is created, you will be prompted for information on each category. A total of 72 characters are available for the information in the user-defined categories.

Enter the label for category # 1

<Return>

Enter: (name) - enter up to 40 characters to specify a category label (i.e. SIZE, CLASS, PURPOSE, etc.).
 q - Quit, finished entering category labels

Davinci allows up to six user-defined categories to classify data files. For example, under category "Related Analysis Program", the user might enter "Movie.BYU" for a data file used by Movie. "Related Analysis Program" is our first category.

Related Analysis Program

How many characters are needed for category # 1
 (Related Analysis Program)
 Enter VALUE:

ZFILEZ stores the user-defined category information, "Movie.BYU", etc., in a 72 character string. The user divides the 72 available columns among the user-defined categories. We will allocate 15 columns for category "Related Analysis Program",

15

Enter the label for category # 2

Let's enter two more categories, "Project Number, Name, or Charge Code" and "Customer Name" and allocate 12 and 20 characters respectively.

Project Number, Name, or Charge Code

How many characters are needed for category #2
(Project Number, Name, or Charge Code)

Enter VALUE:

12

Enter the label for category # 3

Customer Name

How many characters are needed for category #3
(Customer Name)

Enter VALUE:

20

Enter the label for category # 4

q

Please be patient and wait while data file information is input...

We entered q when prompted for the fourth category label to quit the input of categories.

After initializing data file ZFILEZ, Davinci uses the multi-tasking capability of the computer to view the current directory and writes a data file XXFILE containing the name, time of creation, and size of these files. Because this operation may take some time to complete the user is asked to wait. This also serves to ease the mind of the user who may become worried that something has gone wrong.

Davinci then compares the list of file names in XXFILE to those in ZFILEZ. Data files appearing in XXFILE and not in ZFILEZ have been created since the last execution of Davinci and

therefore need to be classified. The first time Davinci is executed all data files need to be classified.

Davinci takes each file in the current directory and asks the user if it is a temporary file. If the user intends to classify a particular file using the various user-defined categories, he should enter n. If the file is only temporary and therefore not to be classified, he should enter y. When a data file is to be classified, the user is asked to enter the information regarding each of the user-defined categories. As all of the data files in the current directory will be classified, we will proceed by entering the necessary information on each of the files as prompted by the program. Only the first few will be shown.

A new file: Dfortran/ is found in your directory

Is this a temporary file? (y/n)

<Return>

Enter: y - to avoid entering data file category information

n - this file is permanent and needs to be classified
using the data file categories

n

Please enter info for Category: Related Analysis Program

<Return>

Enter: (info) - enter up to 15 characters to specify the category information

q - Quit, information left blank

SAP IV

Please enter info for Category: Project Number, Name, or Charge Code

Wing 3.7

Please enter info for Category: Customer Name

Internal

A new file: Disp11 is found in your directory

Is this a temporary file? (y/n)

n

Please enter info for Category: Related Analysis Program

SAP IV

Please enter info for Category: Project Number, Name, or Charge Code

Wing 3.7

Please enter info for Category: Customer Name

Boeing

A new file: Disp12 is found in your directory

Is this a temporary file? (y/n)

.....

Note that dashes indicate the number of characters available for each category.

When new data files are found during subsequent runs of the Davinci program, the user will be given prompts identical to those just illustrated. Davinci also detects when files have been deleted, and allows the user to store the location of the backed-up file.

FILMEN DATA FILE

Davinci and Sentinel will ask the user to enter a new library title (for top menu) when it initializes data file FILMEN, which is used to store the library menu structure and program information.

Enter New LIBRARY TITLE:

Sample Run Program Library

EDIT OF LIBRARY TITLE

1) Sample Run Program Library

This data OK?

y

Sample Run Program Library

Select SENTINEL option: (m,i,x,e,#,v,q)

The user can set up a library of programs using the edit commands described previously. For more information on ZFILEZ and FILMEN files, see the "Programmer Information" section.

II. PROGRAMMER INFORMATION

Program Organization

Davinci, Sentinel, and Gateway routines are listed by functional categories in Table B.2. The relationship between the routines is shown in Table B.3. A brief statement of the function of each subroutine is shown in Table B.4.

Table B.2 ROUTINE FUNCTION LIST

Command Routines	Data Set Handlers	Operator Aids	Character Utilities	Data Struct Utilities
ADD**	CHXSTR**	ERROR	BNDSTR	GETAVL**
BACKUP**	DUMP**	SQUIRE	TXTOUR	GETREC
CHANGE**	FILCMP*	ZHELP	UPCASE	PUTAVL**
EXE	FILDIR*			PUTREC**
FILEDT*	FILLIS*			SAVAVL**
FIND	FLINIT**			
INFO	GETDAT**			
KILL**	GODAD			
LIST**	GOSON			
	TRAVRS			

*Routines marked with * are used only by Davinci, routines marked with ** are used by Davinci and Sentinel only.*

Table B.3 - DAVINCI SUBROUTINE MAP

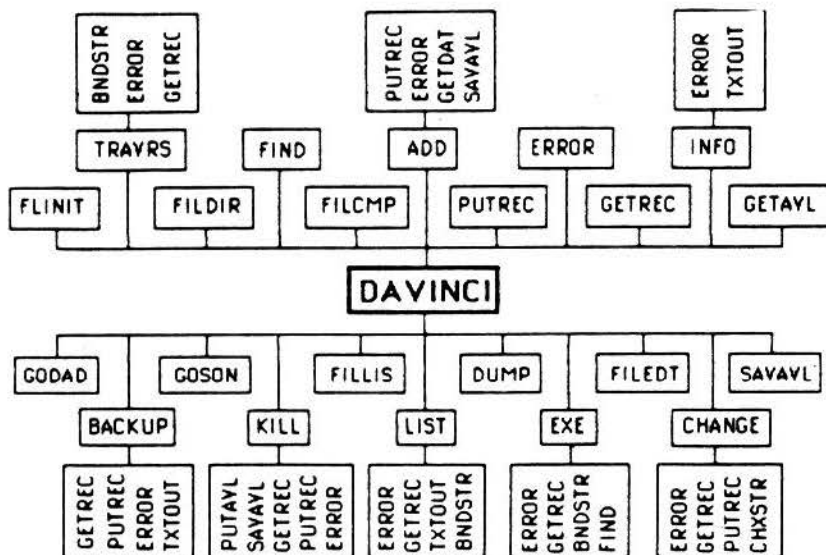


Table B.4 - Subroutine Description

ADD	Creates new program and category menu items
BACKUP	Keeps track as programs are moved on and off-line by setting/clearing the flag that indicates program status
BNDSTR	Bounds a character string and returns the position of the first and last character in the string
CHANGE	Changes an item in the menu by displaying all the information about that item and prompting for instructions
CHXSTR	Changes a multi-line text entry (1 or more records)
DUMP	Outputs the entire FILMEN data file with record numbers appearing in front of each record
ERROR	Displays one line error messages
EXE	Finds, runs the execution instruction for a program.
FILCMP	Compares current data files to files stored in ZFILEZ in search of new files to be classified
FILDIR	Extracts information about the current files in the directory such as name, time of creation, and size
FILEDT	Editing of data file information, user-defined categories
FILLIS	List data file information according to user specifications
FIND	Finds record number of a program and moves to that menu
FLINIT	Initializes direct access file to store data file information
GETAVL	Finds next available line in data file, if value read is zero, GETAVL becomes AVAIL + 1 (GETAVL is an integer function)
GETDAT	Asks for info necessary to add a program, category menu item
GETREC	Retrieves a C or F record from the data file
GODAD	Calls TRAVRS to path up one level
GOSON	Moves to the menu specified by entering the category number
INFO	Lists brief description of a menu item or reference summary
KILL	Deletes a program or category menu listing
LIST	Creates an ordered listing of library menu structure as contained in FILMEN data file, writes it to a disk if desired
PUTAVL	Stores in record the record number of next available record
PUTREC	Adds a C or F record to the data file
SAVAVL	Saves pointer to first empty record, max record pointed to
TRAVRS	Displays menu corresponding to CURDAD
TXTOUT	Writes out strings for title, description, reference summary
UPCASE	Translates a character from lower to upper case, if needed
SQUIRE	Library of input/output routines for user interaction
ZHELP	Input/output help routines called by SQUIRE routines

Data File Structure

Davinci.BYU uses two direct access disk files, FILMEN and ZFILEZ. FILMEN contains menu data and is used to classify programs. ZFILEZ stores data file information and is used to classify data files by name, time of creation, size, status, and six user-defined categories. A brief description of each data file follows.

FILMEN Data Structure

The programmer may produce a sequential file containing FILMEN information using the `d` option (for "dump") in the main level of Davinci or Sentinel. The data file FILMEN used in the sample execution appears in Table B.5.

Read and write statements used with a direct access file can reference any record in the file directly, without having to read all the preceding records sequentially. Records in FILMEN are linked by a system of pointers that maintain relationship between menu items.

Records are linked in a tree structure, similar to a "family tree." A menu category stands as the head of a family of subtopics. The menu heading is the "Dad", the subcategories are the "Sons" or "Brothers." When the user paths down to the next menu, that "Son" becomes the "Dad" over the next group of subcategory "Sons."

There are seven types of data records in the FILMEN. Each type is indicated by the first character field in the record (B, C, D, E, F, I, or N). These record types may be grouped in two classes, corresponding to two different record formats: 1) Master records, and 2) Data records (see Table B.6).

Every menu item has a master record and several associated data records (see Table B.7). For example, lines 2, 12, 21, etc. of the above FILMEN sample listing are Master records. The seven pointers link each item to the library tree structure and to its associated data. If the record type is F, such as line 40, these records also contain the source code location and program code name.

Table B.5 - FILMEN SAMPLE DATA FILE

Select DAVINCI option: (m,i,x,e,#,v,q) *The record numbers at the left help the user follow pointers, and do not appear in data file.*

d *This dump option may help in searching for errors.*

Write a copy of this list to a file? (y/n)

n

1 638 638

2 C 0 0 12 2 0 0 3

3 N 0 1HP 9000 Brigham Young University Software Library

4 D 0 1Movie.BYU Graphics Package

5 I 6 1The MOVIE system of general purpose computer graphics programs

6 I 7 1facilitate the display of three-dimensional, topological, and

7 I 8 1architectural models as line drawings or as continuous tone shaded

8 I 9 1images. This software also provides the capability to clip and ca

9 I 10 1three dimensional systems; modify geometry, displacement, and/or s

10 I 11 1function files; generate new models or title representations; and

11 I 0 1convert contour line definitions into polygonal element mosaics.

12 C 0 21 25 2 5 0 4

13 D 0 1OPTDES.BYU Design Optimization Package

14 I 15 1The OPTDES.BYU package was developed to assist the engineer in obt

15 I 16 1efficient designs. In particular, the package helps the engineer

16 I 17 1perform the following fundamental tasks in the design process:

17 I 18 1 (1) Interfacing with existing analysis software

18 I 19 1 (2) Defining and redefining the design problem

19 I 20 1 (3) Searching for improved designs

20 I 0 1 (4) Interpreting results from the design process

21 C 0 576 151 2 14 0 13

22 D 0 1Movie.BYU Software

23 I 24 1This category contain the various program that form Movie.BYU:

24 I 0 1Display, Utility, Section, Compose, Title, and Mosaic

25 C 0 28 40 12 23 0 22

26 D 0 1Documentation

27 I 0 1This category contains documentation for the Movie.BYU programs.

28 C 0 31 282 12 27 0 26

29 D 0 1Demos

30 I 0 1This category contains demonstrations of the various Movie softwar

31 C 0 0 108 12 30 0 29

32 D 0 3DISPLAY - line drawings & continuous color output

33 I 34 3DISPLAY is an interactive program for the display and animation of

34 I 35 3model composed of polygons. The program allows the user to manipu

35 I 36 3the model (rotate, translate, etc.), specify colors for the backgr

36 I 0 3and the different element parts, and select the display device.

37 I 0 3See the documentation section and demo section of the MOVIE menu.

38 D 0 1Multi view (COMPOSE) output showing primitives from UTILITY

39 E 0 3/users/terril/Dmovie/display

40 F 37 52 0 25 33 39 32/terril/Dmovie/Dfortran/command.f,hidden.fDIS

.....

Hit <RETURN> to continue

Table B.6 FILMEN DIRECT ACCESS RECORD TYPES

Master Records

C (Category)	Heading to a submenu
F (File)	Program menu item

Data Records

B (Backed-up)	Location of tape, etc. of program
D (Description)	Title of menu item
E (Execution)	Program execution statement
I (Information)	Brief Description, Reference Summary of Menu Item
N (Name)	Name of the Library

Table B.7 FILMEN DIRECT ACCESS RECORD FORMAT

A. C and F Record Pointers and Text (Master Records)

FORMAT(A1,7I4,A42,A9)

Column	Variable	Description
1	TYPE	Type of Record in Data File
2-5	POINT(1)	External Reference Summary of Menu Item
6-9	POINT(2)	Brother
10-13	POINT(3)	Son
14-17	POINT(4)	Dad
18-21	POINT(5)	Brief Description of Item
22-25	POINT(6)	Program Execution Statement
26-29	POINT(7)	Title of Menu Item
30-71	TEXT1	Location of Program Source Code
72-80	TEXT2	Program Description Code Name

B. B, D, E, I, and N Record Pointers and Text (Data Records)

FORMAT(A1,I4,I3,A72)

Column	Variable	Description
1	TYPE	Type of Record in Data File
2-5	POINT2(1)	Record number of next line of string
6-8	POINT2(2)	Usage Count, number of items using this string
9-80	TEXT	Information (Title, Brief Description, External Reference, or Execution Statement)

C. Unused Records

FORMAT(2I4)

Column	Variable	Description
1-4	AVAIL	Line number of first available record
5-8	SPAVAL	Highest record number used (record 1 only)

When a menu item is selected, Davinci retrieves the master record of the corresponding son. This becomes the "new Dad". POINT(3) gives the location of the master record of the new dad's eldest son. POINT(2) gives the location of the next son (brother). The chain is followed until a zero is found in POINT(2), indicating no more brothers. As each brother's master record is retrieved, the corresponding menu name is accessed using POINT(7) and the text contained in that data record is displayed.

The MOVE, INFORMATION and EXECUTION options occur similarly. When a program is executed, the command, found by following POINT(6), is also sent to the operating system as a command string.

Data records contain text data such as the title, brief description, reference summary, etc. (see Table 7B). Lines 3, 4, 5, etc. are data records. POINT2(1) is used to point to the next record of multi-line information items (i.e. Titles, Brief Descriptions, and External Reference Summaries). The last line of an information item is identified by POINT2(1) equal to zero. POINT2(2), the usage counter, counts the number of times the program associated with this information appears in the library menu structure. If a program is listed in more than one menu, it will have a separate master record for each menu in which it appears. The pointers, however, will all point to the same data records. POINT2(2), therefore, is used to indicate the number of master records that refer to this data. When a program or category is deleted from a menu, the master record and all associated data records are deleted. However, if POINT2(2) is greater than one, the data record will not be deleted. Instead, the usage counter is reduced by one.

Empty records contain a pointer, AVAIL, which points to the next available record (see Table 7C). Since all the data records are of fixed length, deleted records may be reused. When a data record is deleted, the location of the deleted record is stored in AVAIL to be used by the next record added to the file. When several data records are deleted, a system of pointers is set up forming a chain of unused records. The last record in the chain is the record at the end of the disk file. The pointer SPAVAL points to the highest record number used (record 1 only).

ZFILEZ Data Structure

Data file information is stored in the data file ZFILEZ. The ZFILEZ data file used in the sample execution of the user's guide information appears below.

Table B.8 ZFILEZ SAMPLE DATA FILE

1	53	53	4	17					9			
2	1	15	16	27	28	42	43	62	0	0	0	0
3	Related Analysis Program											
4	Product Number, Name, or Charge Code											
5	Project Number or Name											
6	Customer Name											
7												
8												
9	Dfortran/								84/ 9/17	12:33		110
10	2736SAP IV		Wing	3.7	XP	34.900						Internal
11	Disp11								84/ 9/18	17:24		130
12	2409SAP IV		Wing	3.7	XP	55.433						Boeing
13	Disp12								84/ 9/18	17:24		150
14	2409SAP IV		Wing	3.7	XP	44.672						Boeing
15	Disp13								84/ 9/18	17:24		17BDRAO:[ME.ROSS.FORMAT]
16	2409SAP IV		Wing	3.7	XP	33.981						Boeing
17	Dobject								84/ 7/26	12:49		19T
18	372											
19	GEOM.DAT								84/ 9/18	17:23		210
20	3839Movie.BYU		Wing	3.7	MV	34.9						Boeing
21	filmen								84/ 7/12	13:18		230
22	4894Davinci.BYU		Internal		Internal							Internal

The record numbers in the left column were added to help the user.

The format of this data file consists primarily of two parts: the header and the data file information (see Table B.8). Record 1 stores the location of the next empty record, the bottom most empty record, the number of user-defined category labels, the number of data files in the list, and the record number of the first data file. The second record of the header keeps track of the beginning and ending column numbers of the user-defined category information. The user can define up to six categories for classifying data files and must specify the number of columns (72 available) to contain each category's information. Records 3-8 store the user-defined category labels (up to 40 characters each).

The second part of ZFILEZ contains the data file information (2 records per file). Record a contains the name of the file, its time of creation, the location of the next file in the list, and the file status, including the location of the file, if backed-up (see Table B.9). The file Disp13, record 15 of the above ZFILEZ sample listing, is a backed-up file whose location is [ME.ROSS.FORMAT]. Record b, contains the size of the file, and the category information for each of the user-defined categories. When a file's status is T-temporary, as is the case with the file Dobject at record 17 of the above listing, it is not classified using the user-defined categories. The ZFILEZ data file is only used by Davinci. Sentinel and Gateway do not classify and organize data files.

Table B.9 - ZFILEZ DIRECT ACCESS RECORD FORMAT

A. Header

Record No.1 FORMAT(4I4,38X,I4)

Column	Variable	Description
1- 4	AVAILF	Location of next empty record
5- 8	AVLFMX	Location of bottom most empty record
9-12	NUMCAT	Number of User-defined Category labels
13-16	NUMFIL	Number of data files in the list
55-58	LOCRC1	Location of the first file in the list

Record No. 2 FORMAT(12I3)

1-36	IFLLOC(2,6)	Beginning ending column #'s for user categories
------	-------------	---

Record No. 3-8 FORMAT(A40)

1-40	FLABEL(6)	User-defined Category labels
------	-----------	------------------------------

B. Data File Info

Record a FORMAT(A40,A14,I4,A1,A21)

1-40	FNAME	Up to 40 character filename
41-54	FTIME	Time file was created
55-58	LOCNEW	Location of next data file in the list
59	STAT	File status: on-line(O), temp.(T), backed-up(B)
60-80	LOCBAC	Location (tape ID, etc.) fo backed-up file

Record b FORMAT(I8,A72)

1- 8	LENGTH	File size
9- 80	REC 2	Information for user-defined categories

APPENDIX C

SQUIRE PROGRAMMING GUIDE

Squire.BYU

Programming Guide

Brant A. Ross
College of Engineering
Brigham Young University
Provo, Utah 84602
December 5, 1984

Squire.BYU is a library of Fortran subroutines & functions that:

- creates a consistent interactive environment for users
- provides an arbitrary number of levels of prompts for new users
- allows experienced users to type-ahead answers
- saves programming time on input, editing, & disk access functions.
- allows programmer flexibility.

Neither Brigham Young University nor its employees makes any warranty expressed or implied, or assumes any legal responsibility for the accuracy, completeness or usefulness of this computer program or its associated documentation. Readers are reminded that the information and ideas contained in this document are the property of Brigham Young University and may not be used without permission.

TABLE OF CONTENTS

	Page
Sample Use of the Squire Library	140
Buffering of Commands	146
Multi-Level Prompts	146
Creating Prompt Routines	146
System Dependencies	147
Use of External Statement	147
Redirecting Input and Output	147
Internal Routines	148
Reserved Names	149
ININTQ - Input an integer value with multi-level prompts	150
QINREL - Input a real value using multi-level prompts	151
QRDWRD - Input a single word using multi-level prompts	152
QXTIN - Input a line or phrase using multi-level prompts	153
QEDFIX - Edit fixed combination of real, integer, text items	154
QEDINT - Edit a variable length array of integer numbers	156
QEDREL - Edit a variable length array of real numbers	157
QEDLIS - Edit variable length real, integer, text arrays	158
INWRDQ - Match input to one of various command words	160
MENUSQ - Match input to one of various command letters	161
QYESNO - Ask user a yes/no question	162
QFILRD - Asks for file name and opens file for input	163
QFILWR - Asks for file name and opens file for output	163
QCLEAR - Clears screen and puts cursor at top	164
QINIT - Initializes Squire input buffer and flags	164
QMETRC - Asks user if english or metric units are used	164
QMORE - Used to suppress output when user types ahead	164
QPAUSE - Pauses and waits for return key to continue	165
QSYSTEM - Executes a system (monitor) command	165
QTERM - Asks user to specify terminal attributes	165

Sample Use of Squire Libr

This section contains a sample execution of a subroutine with some of the input/editing features of the Squire Library. Squire uses multi-level prompts and command buffering to accommodate varied user capabilities. The first level prompt is terse for the expert user, the second level is a fairly complete description, and the third level suggests a response. The second and third level prompts are shown when the user hits the return key. The expert user may answer questions ahead by placing the appropriate responses on a single line, separated by spaces.

Comments shown in *italics* do not appear on the terminal screen but have been added here for description purposes. User input is shown in **bold** to make the session easier to understand.

Insertion of a new element type: *(Message given to user when entering subroutine)*
 Enter the IGES topology number:

<return>

The IGES topology number specifies the number of nodes and their arrangement in elements. The topology types (with the number of nodes in parentheses) are: *(2nd level given when user hits return)*

- | | | |
|---------------------|---------------------|---------------------|
| 1. Beam (2) | 2. Triangle (3) | 3. Triangle (6) |
| 4. Triangle(9) | 5. Quad (4) | 6. Quad (8) |
| 7. Quad (12) | 8. Thk Shl Wedg(12) | 9. Thk Shl Wedg(18) |
| 10. Thick Shell(16) | 11. Thick Shell(24) | 12. Solid Tetra(4) |
| 13. Solid Tetra(10) | 14. Solid Wedge(6) | 15. Solid Wedge(15) |
| 16. Solid Wedge(24) | 17. Solid Brick(8) | 18. Solid Brick(20) |
| 19. Solid Brick(32) | 20. Axisym Line(2) | 21. Axisym Line(3) |
| 22. Axisym Line(4) | 23. Axisym Tria(3) | 24. Axisym Tria(6) |
| 25. Axisym Quad(4) | 26. Axisym Quad(8) | |

6

Selected IGES Element Type: PQUAD - Parabolic Quadrilateral
 Element contains 8 nodes.
 This data OK? (y,n)

y

_____ Enter the Name for this element type

CPS8RT2

Enter the Description for this element type

<return>

Enter: text- description for this element type, such as
 "Axisymmetric Fluid Element of Three Nodes."
 q - Quit option

Plane Stress, Quad Disp/Lin. Temp, Red. Int. - Abaqus

Beginning at local node # 1....

Enter the node numbers corresponding to IGES numbers:

1 5 2 6 3 7 8 4

Edit of an element type...

(Edit of a fixed combination of data)

1) Name: CPS8RT2

2) Description: Plane Stress, Quad Disp/Lin. Temp, Red. Int. - Abaqus

Node number map to IGES topology:

3) 1 4) 5 5) 2 6) 6 7) 3 8) 7 9) 8 10) 4

This data OK? (y,n)

n

Edit Command: (c,l,q)

c

Change WHICH item:

9

ITEM # 9 = 8

Enter NEW Value:

4

Edit of an element type...

1) Name: CPS8RT2

2) Description: Plane Stress, Quad Disp/Lin. Temp, Red. Int. - Abaqus

Node number map to IGES topology:

3) 1 4) 5 5) 2 6) 6 7) 3 8) 7 9) 4 10) 4

Edit Command: (c,l,q)

c 10 8

(answer questions ahead the 2nd time around)

Edit of an element type...

1) Name: CPS8RT2

2) Description: Plane Stress, Quad Disp/Lin. Temp, Red. Int. - Abaqus

Node number map to IGES topology:

3) 1 4) 5 5) 2 6) 6 7) 3 8) 7 9) 4 10) 8

Edit Command: (c,l,q)

q

The subroutine used for the sample execution is listed below, with accompanying prompt subroutines.

```

SUBROUTINE INSERT(AVAIL,IPOINT,JPOINT,IGSMAP,IGSNOD,IGSLAB,NODPNT)
C*****
C Insert an element type into the file.
C
C Arguments:
C AVAIL - Next available record in zfilez.dat
C IPOINT- Start pointers to elements with N nodes.
C JPOINT- Start pointers to elements with IGES topology K.
C IGSMAP- Topology map between thsi element and IGES topology.
C TEXT( 1:15) -Name of element.
C TEXT( 16:72) -Element Description.
C IGSLAB- Label for each IGES Element Topology Type.
C IGSNOD- Number of Nodes in Each IGES Element Topology Type.
C Important Internal Variables:
C
C Called by: main
C Calls: QCLEAR
C*****
COMMON /ELEDIT/ NNODEE
COMMON /IODEV/ IDSK
INTEGER AVAIL
CHARACTER TEXT*72,IGSLAB(26)*40
DIMENSION IPOINT(14),JPOINT(26),IGSMAP(32),IGSNOD(26),
1 NODPNT(32),ITCNT(2)
LOGICAL QYESNO,QTXTIN,QMORE
EXTERNAL ZENAME,ZEDESC,ZIGEST,ZTMAP,QED7,ZEDIT
DATA ITCNT/15,57/
C
WRITE(*,*) ' Insertion of a new element type:'
DO 5 I=1,32
IGSMAP(I) = 0
5 CONTINUE
C
C Ask for the IGES topology type of the element to be insert.
10 IGEST = ININTQ(ZIGEST,1)
IF(IGEST.EQ.-99999) GO TO 900
IF(IGEST.LT.1.OR.IGEST.GT.26) THEN
WRITE(*,15) IGEST
15 FORMAT(' Sorry...','I3,' is not a valid topology type')
GO TO 10
END IF
C
C Echo information about this IGES element type, ask if OK.
NNODE = IGSNOD(IGEST)
WRITE(*,20) IGSLAB(IGEST), NNODE

```

```

20 FORMAT(// ' Selected IGES Element Type: ',A40/
1      ' Element contains',I3,' nodes.')
      IF(.NOT.QYESNO(QED7)) GO TO 10
C
C Ask for the element type name
      CALL QRDWRD(TEXT(1:15),15,ZENAME)
      IF(TEXT(1:15).EQ.' ') GO TO 900
C
C Ask for the element description ( up to 57 characters)
      IF(.NOT.QTXTIN(TEXT(16:72),57,ZEDESC)) GO TO 900
C
C Ask for Node number mapping between IGES connectivity and
C element connectivity.
      DO 50 J=1,IGSNOD(IGEST)
          IF(QMORE(IDUMMY)) WRITE(*,25) J
25      FORMAT(/ ' Beginning at local node #',I2,'....')
30      IGSMAP(J) = ININTQ(ZTMAP,-1)
          IF(IGSMAP(J).LT.1.OR.IGSMAP(J).GT.IGSNOD(IGEST)) THEN
              WRITE(*,40) IGSMAP(J)
40          FORMAT(/ ' The node number: ',I2,' is out of bounds')
              GO TO 30
          END IF
50      CONTINUE
C
C Let the user review the data before putting it into the file.
      NNODEEE = NNODE
      CALL QEDFIX(ZEDIT, 1,A, NNODE,IGSMAP, 72,TEXT, 2,ITCNT,
1      0,0, 3,NNODE+2, 1,2)
C
C File (i) needs to be inserted into the data set.
C Find the location of the next empty space after AVAILF.
      .....
C
C Write info for this file at location specified by AVAIL.
      .....
C
900 RETURN
      END

SUBROUTINE QABORT(LEVEL)
WRITE(*,10)
10 FORMAT(' Contact your local CAESO representative.')
STOP
END

```

SUBROUTINE ZEDESC(LEVEL)

```

C Called by: (main)
  GO TO (20,40,50,10), LEVEL
10 CALL QABORT
20 WRITE(*,60)
30 RETURN
40 WRITE(*,70)
  GO TO 30
50 WRITE(*,80)
  GO TO 30
60 FORMAT('/ Enter the Description for this element type'/57('_'))
70 FORMAT(' Enter: text- description for this element type, such as'/
  2      '      "Axisymmetric Fluid Element of Three Nodes."'/
  3      '      q - Quit option'/57('_'))
80 FORMAT(' Why not enter: q NO further help!')
  END

```

SUBROUTINE ZEDIT(A,IGSMAP,TEXT)

```

C Called by: INSERT,CHANGE
  COMMON/ELEDIT/NNODEE
  DIMENSION A(1),IGSMAP(1)
  CHARACTER*72 TEXT
  WRITE(*,10) TEXT(1:15),TEXT(16:72)
10 FORMAT(' Edit of an element type... '//
  1      ' 1)      Name: ',A15/
  2      ' 2) Description: ',A57//)
  WRITE(*,*) ' Node number map to IGES topology:'
  WRITE(*,20) (J+2,IGSMAP(J),J=1,NNODEE)
20 FORMAT(8(3X,12,')',13))
  RETURN
  END

```

SUBROUTINE ZENAME(LEVEL)

```

C Called by: (main), GETDAT
  GO TO (20,40,50,10), LEVEL
10 CALL QABORT
20 WRITE(*,60)
30 RETURN
40 WRITE(*,70)
  GO TO 30
50 WRITE(*,80)
  GO TO 30
60 FORMAT('/ _____ Enter the Name for this element type')
70 FORMAT(' Enter: text- name for this element type, such as QUAD8, '/
  2      '      CAX8R, CELAS2. These names should be recognized' /
  3      '      by the target FEA codes (Abaqus, Nastran, SAP).'/
  4      '      q - Quit option'/' _____')
80 FORMAT(' Why not enter: q NO further help!')
  END

```



```

SUBROUTINE ZIGEST(LEVEL)
C Called by: (main)
GO TO (20,40,50,10), LEVEL
10 CALL QABORT
20 WRITE(*,60)
30 RETURN
40 WRITE(*,70)
WRITE(*,75)
GO TO 30
50 WRITE(*,80)
GO TO 30
60 FORMAT(' Enter the IGES topology number:')
70 FORMAT(
1 ' The IGES topology number specifies the number of nodes and /
2 ' their arrangement in elements. The topology types (with the /
3 ' the number of nodes in parentheses) are: //
4 ' 1. Beam (2)          2. Triangle (3)          3. Triangle (6) /
5 ' 4. Triangle(9)      5. Quad (4)           6. Quad (8) /
6 ' 7. Quad (12)        8. Thk Shl Wedg(12)    9. Thk Shl Wedg(18) /
7 ' 10. Thick Shell(16) 11. Thick Shell(24)   12. Solid Tetra(4) /
8 ' 13. Solid Tetra(10) 14. Solid Wedge(6)     15. Solid Wedge(15) /
9 ' 16. Solid Wedge(24) 17. Solid Brick(8)    18. Solid Brick(20) /
75 FORMAT(
1 ' 19. Solid Brick(32) 20. Axisym Line(2)   21. Axisym Line(3) /
2 ' 22. Axisym Line(4) 23. Axisym Tria(3)   24. Axisym Tria(6) /
3 ' 25. Axisym Quad(4) 26. Axisym Quad(8) /
80 FORMAT(' Why not enter: 5 for a 4-node quad. No further help')
END

C
C
SUBROUTINE ZTMAP (LEVEL)
C Called by: (main)
GO TO (20,40,50,10), LEVEL
10 CALL QABORT
20 WRITE(*,60)
30 RETURN
40 WRITE(*,70)
GO TO 30
50 WRITE(*,80)
GO TO 30
60 FORMAT(' Enter the node numbers corresponding to IGES numbers:')
70 FORMAT(' Enter the node numbers for this element type that correspond /
1 ' to each number in the appropriate IGES topology type.')
80 FORMAT(' Why not enter: q NO further help!')
END

```

Buffering of Commands

Squire.BYU inputs all information into an 80-character buffer. The user may enter as many answers as he would like in each line of input. The answers must be separated by spaces or commas. Buffering is cancelled when a text string is read in, because no basis exists for evaluating the meaning of delimiters.

Multi-Level Prompts

User actions determine the quantity of prompts given for each input. No prompt is given when an answer has been typed ahead. Once the buffer is empty, the first-level prompt is written to the screen. The user may answer the question or hit the Return key without giving any response to get further levels of prompts. It is suggested that three levels of prompts be used. The first level is a terse request appropriate to an expert user. The second level is a complete description of the information needed. The third level suggests a response and tells the user that no further help is available. Although three levels of prompts are recommended, the Squire library will allow any number of prompt levels.

Creating Prompt Routines

A Fortran program, Smithy.BYU, may be used to create multi-level prompt subroutines. Smithy requires a subroutine name and a set of prompts to produce a subroutine for each input.

System Dependencies

System dependent features are localized in Squire to minimize conversion efforts. Such features are found in routines: QCLEAR, QEDO, QFILRD, QFILWR, and QSYSTEM. An explanation of the feature and suggested implementations for UNIX and VAX/VMS systems are given in the routine.

Use of External Statement

External statements are not frequently used in most Fortran software, but are used in almost every routine by SQUIRE. An external statement allows a subroutine name to be passed in an argument list. This feature is needed to allow the programmer to easily use his own subroutines to write prompts and display data for editing. When using Squire subroutines, carefully note how external statements are used in the examples that follow. Errors in the use of external statements usually produce no errors until the program is executed. When a program reaches a call statement containing a routine name that is not listed in an external statement, it will abort.

Redirecting Input and Output

Input and output may be redirected to or from a disk file at any time except at the input of a line of text. This feature can create a record of the use of a program for verification, and allows the saved file to be used to quickly run through the program and bring you to the point where you left off. To save input, type a > (greater than sign) followed by a space and the name of the file where the data is to be saved. All subsequent input will be saved to the file until another > symbol is typed, which causes the file to

be closed. To extract input information from a file, type a < (less than sign), followed by the file name. The program will read all input from the file until it completely reads the file. Currently Fortran Unit Numbers 18 and 19 are used for the redirected input and output files. This can easily be changed by changing the appropriate statements in subroutine QINIT.

Internal Routines

Various subroutines and functions are called by other routines in the Squire library, and not intended for use by the application programmer. They are summarized here:

Input Utilities:

- QBADIN - Call if invalid input, echos bad input, clears input buffer
- QCIINT - Conditional input of an integer number. False if invalid
- QCIREL - Conditional input of a real number. False if invalid
- QINBUF - Reads in line of input when needed from keyboard or file
- QNEXT - Parses next item of input from the input buffer
- QNEXT2 - Same as QNEXT, used to avoid recursion
- QNOBUF - Clears input buffer to initialize at start or after error

Prompt Routines:

- QEDO to - Used in edit routines to prompt for: 0) new text,
- QED9 1) edit command(5), 2) item to delete, 3) verify delete,
- 4) item to insert, 5) value to insert, 6) edit command(3)
- 7) data OK?, 8) item to change, 9) value to change
- QFILNM - Asks for name of input or output disk file
- QFLIND - Asks for option when input file doesn't exist
- QFLSAV - Asks for option when output file already exists
- QRASTR - Asks for resolution of graphics terminals
- QSITXT - Asks user if he is using metric units
- QTBAUD - Asks user for baud rate of terminal
- QTTYPE - Asks user for terminal type

Reserved Names

The following is a list of subroutine, function and labelled common names that are used in Squire. The use of these names by the application program may cause multiple definition errors.

QBADIN	QBUFFR	QCHBUF	QCIINT	QCIREL
QCLEAR	QED0	QED1	QED2	QED3
QED4	QED5	QED6	QED7	QED8
QED9	QEDFIX	QEDINT	QEDLIS	QEDREL
QFILRD	QFILWR	QFILNM	QFLIND	QFLSAV
QINBUF	ININTQ	QINIT	QINREL	INWRDQ
QLINEL	MENUSQ	QMETRC	QMORE	QNEXT
QNEXT2	QNOBUF	QPAUSE	QRASTR	QRDWRD
QREDIR	QSITXT	QSYSTEM	QTBAUD	QTERM
QTERML	QTTYPE	QXTIN	QYESNO	

Note that most of these names begin with a Q, to avoid most conflicts with names used in application programs. Some integer functions have a Q at the end of their name. This allows integer names to be used, and avoids the need of integer declaration statements where these functions are used.

INTEGER FUNCTION ININTQ

Purpose: ININTQ asks user to enter an integer value. If user inputs a string that is an invalid integer (contains a period, letters, etc.), he will be prompted to reenter the number. The user may enter a Q or q to quit input (ININTQ is given a value of -99999), or a D or d to use the default value.

Format:

Use an external statement at the top of module for prompt subroutine:

```
EXTERNAL PROMPT
```

Call the function where needed

```
IVALUE = ININTQ(PROMPT,IDEFLT)
```

where IVALUE is the integer value input, PROMPT the name of prompt routine and IDEFLT the default value.

EXAMPLE

```
SUBROUTINE ITERAT(IVALUE)
```

C Enter the number of iterations, then do them.

```
EXTERNAL ZNITER
```

C

C Enter the number of iterations, default is one.

```
NUM = ININTQ(ZNITER,1)
```

```
IF(NUM.LT.0) RETURN
```

```
:
```

```
RETURN
```

```
END
```

C

C

```
SUBROUTINE ZNITER(LEVEL)
```

```
GO TO (20,40,50,10), LEVEL
```

```
10 CALL QABORT
```

```
20 WRITE(*,60)
```

```
30 RETURN
```

```
40 WRITE(*,70)
```

```
GO TO 30
```

```
50 WRITE(*,80)
```

```
GO TO 30
```

```
60 FORMAT(' Enter number of iterations:')
```

```
70 FORMAT(' Please enter the number of iterations of algorithm Z' /
```

```
1 ' that you would like to do.')
```

```
80 FORMAT(' Why not enter: 2 to try a couple. NO further help.')
```

```
RETURN
```

```
END
```

REAL FUNCTION QINREL

Purpose: QINREL asks the user to enter a real value. If the user inputs an invalid string (contains invalid letters, etc.), he will be prompted to reenter the number. The user may enter a Q or q to quit input (QINREL is given a value of -99999), or a D or d to use the default value.

Format:

Use an external statement at the top of module for prompt subroutine:

```
EXTERNAL PROMPT
```

Call the function where needed

```
VALUE = QINREL(PROMPT,DEFAULT)
```

where VALUE is the integer value input, PROMPT the name of the prompt routine and DEFAULT is the default value.

EXAMPLE

```

SUBROUTINE COORD(X,Y,Z)
C Enter X, Y, and Z coordinates of the pipe.
EXTERNAL ZXCOORD, ZYCOORD, ZZCOORD
C
X= QINREL(ZXCOORD,0.0)
IF(X.EQ.-99999.) RETURN
Y= QINREL(ZYCOORD,0.0)
Z= QINREL(ZZCOORD,0.0)
.
RETURN
END
C
C
SUBROUTINE ZXCOORD(LEVEL)
GO TO (20,40,50,10), LEVEL
10 CALL QABORT
20 WRITE(*,60)
30 RETURN
40 WRITE(*,70)
GO TO 30
50 WRITE(*,80)
GO TO 30
60 FORMAT(' Enter X,Y,Z coordinates of pipe bend:')
70 FORMAT(' Please enter the X, Y, and Z coordinates of the pipe bend or enter just the/'
1 ' X coordinate now and be prompted for the Y and Z coordinates later.')
80 FORMAT(' Why not enter: 0.0 NO further help.')
RETURN
END

```

Note: In this example, ZXCOORD prompts for all three values, ZYCOORD would prompt for the X and Y values, and ZZCOORD would prompt for just the Z value. The result is an adapting prompt.

SUBROUTINE QRDWRD

Purpose: QRDWRD is used to input a word of text that does not contain spaces or commas. This is useful to input file names, program names, etc. If the user enters a Q or q, a blank string will be returned.

Format:

Use an external statement at the top of module for prompt subroutine:

```
EXTERNAL PROMPT
```

Call the subroutine where needed

```
CALL QRDWRD(TEXT,NCHAR,PROMPT)
```

where TEXT is the text string input, NCHAR is the maximum number of characters allowed in the string, and PROMPT the name of the prompt routine.

EXAMPLE

```
SUBROUTINE FOPEN(IDSK)
```

C Open a user named disk file for unit number IDSK for output

```
CHARACTER*15 FILNAM
```

```
EXTERNAL ZFILNM
```

C

C Enter the file name

```
CALL QRDWRD(FILNAM,15,ZFILNM)
```

```
OPEN(UNIT=IDSK,FILE=FILNAM,STATUS='UNKNOWN')
```

```
.
```

```
.
```

```
.
```

```
RETURN
```

```
END
```

C

C

```
SUBROUTINE ZFILNM(LEVEL)
```

```
GO TO (20,40,50,10), LEVEL
```

```
10 CALL QABORT
```

```
20 WRITE(*,60)
```

```
30 RETURN
```

```
40 WRITE(*,70)
```

```
GO TO 30
```

```
50 WRITE(*,80)
```

```
GO TO 30
```

```
60 FORMAT('_____ Enter output file name:')
```

```
70 FORMAT(' Please enter the name of the output file for the XYZ data.')
```

```
1 _____)
```

```
80 FORMAT(' Why not enter: XYZ.DAT NO further help.')
```

```
RETURN
```

```
END
```


LOGICAL FUNCTION QXTIN

Purpose: QXTIN clears the input buffer and asks the user to enter a line of text. Spaces and commas are ignored. The user may enter a Q or q and a blank string will be returned and the function will be false.

Format:

Use an external statement at the top of module for prompt subroutine:

```
EXTERNAL PROMPT
```

Call the function where needed

```
IF(.NOT.QXTIN(STRING,NCHAR,PROMPT)) RETURN
```

where STRING is the character string to be input, NCHAR is the number of characters in the string, and PROMPT the name of the prompt routine.

EXAMPLE:

```

SUBROUTINE START(TITLE)
C Enter a title for this analysis
  CHARACTER*72 TITLE
  LOGICAL QXTIN
  EXTERNAL ZTITIN
C
  IF(QXTIN(TITLE,72,ZTITIN)) RETURN
C User didn't give a title, so set up a default title
  TITLE = ' Analysis of Hoop Stress, XYZ Company'
  .
  .
  .
  RETURN
END
C
C
  SUBROUTINE ZTITIN(LEVEL)
  GO TO (20,40,50,10), LEVEL
10 CALL QABORT
20 WRITE(*,60)
30 RETURN
40 WRITE(*,70)
  GO TO 30
50 WRITE(*,80)
  GO TO 30
60 FORMAT(' Enter title of this hoop stress analysis'/72*('_'))
70 FORMAT(' Please enter a title for this analysis to be used to label it so that someone/'
1      ' besides our competitor may recognize it 6 months from now.'/72*('_'))
80 FORMAT(' Why not enter:   Analysis of part #25398-121-B'/
1      ' NO further help.')
  RETURN
END

```

SUBROUTINE QEDFIX

Purpose: QEDFIX provides a general purpose edit of a fixed combination of real, integer and text items (The user can only **change** items, not insert or delete). To edit values, the programmer needs to write a subroutine to list out the data (arguments are the real, integer, and text arrays), and include a call statement where needed. Real, integer and text items must be numbered consecutively. An arbitrary number of variable-length strings may be edited, but the strings must be concatenated together in a single global string which is passed through the argument list.

Format:

Use an external statement at the top of module for listing subroutine:

```
EXTERNAL LIST
```

Call the subroutine where needed

```
CALL QEDFIX(LIST, KA,A, KM,M, KT,IT,KC,ITCNT, IRS,IRE, IIS,IIE, ITS,ITE)
```

where:

LIST	- Name of subroutine that lists out data (programmer supplied)
KA	- Number of real values to be edited
A	- Array of real numbers to be edited
KM	- Number of integer values to be edited
M	- Array of integer numbers to be edited
KT	- Total number of text characters in the global string
IT	- Global string to be edited
KC	- Number of text strings in the global string
ITCNT	- Array of the number of characters in each text string
IRS,IRE	- Starting, ending item number of real values
IIS,IIE	- Starting, ending item number of integer values
ITS,ITE	- Starting, ending item number of text strings

The programmer-supplied subroutine used to list out the data has the subroutine statement:

```
SUBROUTINE LIST( A, M, IT )
REAL A(KA)
INTEGER M(KM)
CHARACTER*KT IT
```

EXAMPLE:

```

SUBROUTINE ELEMNT(IVAL,ELEM,ENAM)
C Edit element number NUMENT
COMMON/CLSEDT/NUMENT,NUMINT,NUMREL,NUMTXT
INTEGER IVAL(*), ELEM(*), ITCNT(1)
CHARACTER*15 ENAM(50), ENAME
EXTERNAL ZELMHD
DATA ITCNT/15/
C Find location of integer values, number of nodes
LOC = ELEM(NUMENT)
NUMINT = IVAL(LOC+1)
ENAME = ENAM(IVAL(LOC))
C Two other integer values besides nodes are edited
CALL QEDFIX(ZELMHD, 1,A, NUMINT+2,IVAL(LOC+2), 15,ENAME,
1          1,ITCNT, 0,0, 2,NUMINT+3, 1,1)
.
RETURN
END
C
C
SUBROUTINE ZELMHD(A,IVAL,ENAM)
COMMON/CLSEDT/NUMENT,NUMINT,NUMREL,NUMTXT
INTEGER IVAL(1)
CHARACTER*15 ENAM
C
WRITE(*,10) NUMENT,ENAM,IVAL(1),IVAL(2)
10 FORMAT(15X,' Edit of ELEMENT #',I5//
1   ' 1)          Element Name: ',A15/
2   ' 2) Material Property ID:',I6/
3   ' 3) Geometric Property ID:',I6)
WRITE(*,20)
20 FORMAT(//' Nodes in element connectivity:')
WRITE(*,30) (J+1,IVAL(J),J=3,NUMINT+2)
30 FORMAT(1X,5(12,')',I5,4X),I2,')',I5)
RETURN
END

```

SUBROUTINE QEDLIS

Purpose: QEDLIS provides a general purpose edit of a combination of real, integer and text items. The user may change, insert or delete any type of data. To edit a set of values, the programmer needs to write a subroutine to list out the data, and include a call statement where the edit should occur. Real, integer, and text items must be numbered consecutively. For example, if a maximum of 10 real, 15 integer, and 7 text items were edited, the reals would be numbered 1-10, integers 11-25 and text strings 26-32. All strings in the text array have the same length.

Format:

Use an external statement at the top of module for listing subroutine:

```
EXTERNAL LIST
```

Call the subroutine where needed

```
CALL QEDLIS(LIST, R,NR,NRX, M,NM,NMX, T,NT,NTX,NC)
```

where:	LIST	- Name of subroutine that lists out data (programmer supplied)
	R	- Array of real numbers to be edited
	NR	- Number of real values in array. Updated as reals are inserted, deleted
	NRX	- Maximum number of real values array R can hold
	M	- Array of integer numbers to be edited
	NM	- Number of integers in array. Updated as integers are inserted, deleted
	NMX	- Maximum number of integer values array M can hold
	T	- Array of text characters to be edited
	NT	- Number of strings in array. Updated as strings are inserted, deleted
	NTX	- Maximum number of text strings array T can hold
	NC	- Number of characters in each string in the array

The programmer-supplied subroutine used to list out the data has the subroutine statement:

```
SUBROUTINE LIST( NR,R, NM,M, NT,T )
REAL R(NRX)
INTEGER M(NMX)
CHARACTER*NC T(NTX)
```

EXAMPLE:

```

SUBROUTINE MLCONS (RVAL, IVAL, CONS)
C Edit node list for multiple constraint number NUMENT
COMMON/CLSEDT/NUMENT,NUMINT,NUMREL,NUMTXT
DIMENSION RVAL(*), IVAL(*), CONS(*)
CHARACTER*1 T
EXTERNAL ZMCNHD
C Find location of integer values, number of nodes
LOC = CONS(NUMENT)
NUMINT = IVAL(LOC+1)*2 + 2
LOCR = IVAL(LOC+2)
NUMREL = IVAL(LOC+1)
C Two other integer values besides nodes are edited
CALL QEDLIS(ZMCNHD, RVAL(LOCR),NUMREL,20, IVAL(LOC+3),
1          NUMINT,40, T,0,0,0)

RETURN
END

C
C
SUBROUTINE ZELMHD(NR,RVAL,NM,IVAL,NT,T)
COMMON/CLSEDT/NUMENT,NUMINT,NUMREL,NUMTXT
DIMENSION RVAL(20), IVAL(40)
CHARACTER*(*) T

C
WRITE(*,*) ' Edit of Multiple Constraint #', NUMENT

C
WRITE(*,10)
10 FORMAT(// ' Dependent node: 21)',I5/I5X,' DOF: 22)',I5//
1      'Independent Node Degree of Freedom      Ratio')
WRITE(*,20) (19+J*2,IVAL(2*J-1),2*J+20,IVAL(2*J),J,RVAL(J),
1          J=1,NUMREL)
20 FORMAT(1X,2(I3,','),I8,I3,','),0I3.5)
RETURN
END

```

SUBROUTINE QEDINT

Purpose: QEDINT provides a general purpose edit of an array of integer numbers. The user may change, insert, or delete values. To edit a set of values, the programmer needs to write a subroutine to write a header for the data, and include a call statement where the edit should occur.

Format:

Use an external statement at the top of module for header listing subroutine:

```
EXTERNAL HEAD
```

Call the subroutine where needed

```
CALL QEDINT(HEAD, M, N, NMAX, NCOL, NPAGE)
```

where: HEAD - Name of subroutine that writes a header (programmer supplied)
 M - Array of integer numbers to be edited
 N - Number of integer values in array. Updated as items inserted or deleted.
 NMAX - Maximum number of integer values array can hold
 NCOL - Number of columns for listing integer numbers
 (NCOL = 1-5 for that many columns, 6 for 2 double columns)
 NPAGE - Maximum number of integers allowed on the screen at one time

EXAMPLE

```
SUBROUTINE ENCONS(IVAL,CONS)
```

C Edit nodes for permanent constraint NUMENT

```
COMMON/CLSEDT/NUMENT,NUMINT,NUMREL,NUMTXT
```

```
INTEGER IVAL(*),CONS(*)
```

```
EXTERNAL ZCONHD
```

C Find location of integer values, number of nodes

```
LOC = CONS(1,NUMENT)
```

```
NUMINT = IVAL(LOC+1)
```

C Allow up to 100 nodes, but only 75 may appear on the screen at one time.

C The editor will go to 2 full screen edits if over 75.

```
CALL QEDINT(ZCONHD,IVAL(LOC+3),NUMINT,100,5,75)
```

```
RETURN
```

```
END
```

C

C

```
SUBROUTINE ZCONHD
```

```
COMMON/CLSEDT/NUMENT,NUMINT,NUMREL,NUMTXT
```

```
WRITE(*,10) NUMENT
```

```
10 FORMAT(' Edit of Nodes for Permanent Constraint *',I5)
```

```
RETURN
```

```
END
```

SUBROUTINE QEDREL

Purpose: QEDREL provides a general purpose edit of an array of real numbers. The user may change, insert or delete data. To edit a set of values, the programmer needs to write a subroutine to output a header for the data, and include a call statement where the edit should occur.

Format:

Use an external statement at the top of module for header listing subroutine:

```
EXTERNAL HEAD
```

Call the subroutine where needed

```
CALL QEDREL(HEAD, A, N, NMAX, NCOL, NPAGE)
```

where: HEAD - Name of subroutine that writes a header (programmer supplied)
 A - Array of real numbers to be edited
 N - Number of real values in array. Updated as items inserted or deleted.
 NMAX - Maximum number of real values array can hold
 NCOL - Number of columns to list real numbers
 (NCOL = 1-4 for that many columns, 5 for 2 double columns)
 NPAGE - Maximum number of reals allowed on the screen at one time

EXAMPLE:

```

SUBROUTINE EMPROP(IVAL, RVAL, MATL)
C Edit list of material properties for prop # NUMENT
COMMON/CLSEDT/NUMENT,NUMINT,NUMREL,NUMTXT
DIMENSION IVAL(*), RVAL(*), MATL(*)
EXTERNAL ZPRPHD
C Find location of real values for the material property
LOC = MATL(1,NUMENT) + 3
NUMREL = IVAL(LOC+1) + 2
C Edit the material property values
CALL QEDREL(ZPRPHD, RVAL(LOC+4), NUMREL,100, 4, 60)
.
.
RETURN
END

C
C
SUBROUTINE ZPRPHD
COMMON/CLSEDT/NUMENT,NUMINT,NUMREL,NUMTXT
WRITE(*,10) NUMENT
10 FORMAT(' Edit of Properties for Material #',I5)
RETURN
END

```

INTEGER FUNCTION INWRDQ

Purpose: INWRDQ lets the user select an option using key words. INWRDQ returns the number of the selected command. For example, if SAVE is chosen from the commands: READ, WRITE, SAVE, QUIT; INWRDQ returns "3" (3rd word was selected). The user need not enter the entire command, just a unique substring. When a substring matches several commands, the first command is selected. Command words are stored in A1 format, separated by delimiters. The delimiter is the first character in the array.

Format:

Dimension, fill data for command array, use external statement for prompt routine

```
CHARACTER*1 WORDS(NCHAR)
DATA WORDS/'$','C','O','M','I', '$','C','O','M','2','$'/
EXTERNAL PROMPT
```

Call the subroutine where needed

```
IOPT = INWRDQ(WORDS,NCHAR,PROMPT)
```

where WORDS- command words, NCHAR- # of characters in WORDS, PROMPT- prompt routine.

EXAMPLE

```
SUBROUTINE COMMND
```

C Let the user select a command to execute

```
CHARACTER*1 WORD(22)
DATA WORD/'$','R','E','A','D','$','W','R','I','T','E','$','S','A','V','E','$','Q','U','I','T','$'/
EXTERNAL ZCOMMND
```

C Command Selection, branch to the proper option

```
IOPT = INWRDQ(WORD,22,ZCOMMND)
```

C Read,Write, Save, Quit

```
GO TO ( 10, 20, 30, 40), IOPT
```

```
RETURN
```

```
END
```

C

```
SUBROUTINE ZCOMMND(LEVEL)
```

```
GO TO (20,40,50,10), LEVEL
```

```
10 CALL QABORT
```

```
20 WRITE(*,60)
```

```
30 RETURN
```

```
40 WRITE(*,70)
```

```
GO TO 30
```

```
50 WRITE(*,80)
```

```
GO TO 30
```

```
60 FORMAT('Enter Command (READ,WRITE,SAVE,QUIT):')
```

```
70 FORMAT(' Please enter: READ data, WRITE data, SAVE data, or QUIT')
```

```
80 FORMAT(' Why not enter: QUIT to stop program. NO further help.')
```

```
END
```


INTEGER FUNCTION MENUHQ

Purpose: MENUHQ lets the user select an option or command through a key letter. MENUHQ returns the number of the letter that was selected.

Format:

Dimension, fill data for command array, use external statement for prompt routine

```
CHARACTER*1 WORDS(NCHAR)
```

```
DATA WORDS/'A','B','C','D'/
```

```
EXTERNAL PROMPT
```

Call the subroutine where needed

```
IOPT = MENUHQ( WORDS,NCHAR,PROMPT)
```

where WORDS is the command letter array, NCHAR is the number of characters in WORDS, and PROMPT the name of the prompt routine.

EXAMPLE

```
SUBROUTINE COMMND
```

C Let the user select a command to execute

```
CHARACTER*1 WORDS(22)
```

```
DATA WORDS/'r','w','s','q'/
```

```
EXTERNAL ZCOMMND
```

C Command Selection, branch to the proper option

```
IOPT = MENUHQ( WORDS ,4, ZCOMMND)
```

C Read,Write, Save, Quit

```
GO TO ( 10, 20, 30, 40), IOPT
```

```
.
```

```
.
```

```
.
```

```
RETURN
```

```
END
```

C

C

```
SUBROUTINE ZCOMMND(LEVEL)
```

```
GO TO ( 20,40,50,10), LEVEL
```

```
10 CALL QABORT
```

```
20 WRITE(*,60)
```

```
30 RETURN
```

```
40 WRITE(*,70)
```

```
GO TO 30
```

```
50 WRITE(*,80)
```

```
GO TO 30
```

```
60 FORMAT('Enter Command (r,w,s,q):')
```

```
70 FORMAT(' r - Read data from a disk file'/
```

```
1      'w - Write data to the line printer'/
```

```
2      's - Save data to a disk file'/
```

```
3      'q - Quit program execution')
```

```
80 FORMAT(' Why not enter:  q  to stop program. NO further help.')
```

```
END
```

LOGICAL FUNCTION QYESNO

Purpose: INWRDQ lets user select 1 of 2 options with a yes/no question. QYESNO is true when user answers yes, false when user answers no.

Format:

Use external statement at top of module for prompt routine, declare QYESNO logical
 EXTERNAL PROMPT
 LOGICAL QYESNO

Call the subroutine where needed
 IF(QYESNO(PROMPT)) THEN
 do alternative #1
 ELSE
 do the other alternative
 END IF

where PROMPT is the name of the prompt routine.

EXAMPLE

```

SUBROUTINE COMMND
C Let the user select a command to execute
  LOGICAL QYESNO
  EXTERNAL ZCOMND
C See if user wants to continue with this program
  IF(QYESNO(ZCOMND)) THEN
    RETURN
  ELSE
    STOP
  END IF
C
  END
C
C
  SUBROUTINE ZCOMND(LEVEL)
    GO TO (20,40,50,10), LEVEL
10 CALL QABORT
20 WRITE(*,60)
30 RETURN
40 WRITE(*,70)
   GO TO 30
50 WRITE(*,80)
   GO TO 30
60 FORMAT('Continue? (y/n)')
70 FORMAT(' Enter:  y  to continue running this program./
1      '      n  to stop the program.')
80 FORMAT(' Why not enter:  n  to stop program. NO further help.')
  END

```

SUBROUTINES QFILRD & QFILWR

Purpose: QFILRD and QFILWR open disk files for input and output respectively, using file names of the user's choice. If the input file does not exist, the user is warned and asked to: 1) input a new file name, 2) examine current files, or 3) quit execution. If the output file already exists, the user is warned and asked to: 1) input a new file name, 2) overwrite existing file, 3) examine current files, or 4) quit execution.

Format:

Call the subroutine where needed

```
CALL QFILRD(IDSK,ITYP,NBYTE)
```

```
CALL QFILWR(IDSK,ITYP,NBYTE)
```

where

IDSK - Fortran Unit Number for the disk file

ITYP - 1. Sequential, Formatted

2. Sequential, Unformatted

3. Direct Access, Formatted

4. Direct Access, Unformatted

NBYTE - Number of bytes in record (only use for Direct Access Files)

EXAMPLE

```
SUBROUTINE INOUT
```

C Input data from disk, edit it, write it back out to disk

```
WRITE(*,*) 'Selection of bulk data input file:'
```

```
CALL QFILRD(8,1,0)
```

```
READ(8,*) .....
```

```
CLOSE(UNIT=8)
```

C Edit Data

```
.....  
.....  
.....
```

C

```
WRITE(*,*) 'Save corrected bulk data to disk...'
```

```
CALL QFILWR(8,1,0)
```

```
WRITE(8,10) .....
```

```
CLOSE(UNIT=8)
```

```
RETURN
```

```
END
```

SUBROUTINE QCLEAR

Purpose: QCLEAR clears the terminal screen. It may be called by the programmer and is needed by the edit subroutines. QCLEAR is used with QTERM in computers that use different commands to clear different types of terminals. Terminal parameters are passed in COMMON QTERML.

Format: CALL QCLEAR

SUBROUTINE QINIT

Purpose: QINIT is used to initial the input buffer and various Squire flags. It should be called at the beginning of the application program, before any other Squire routine is used.

Format:
 CALL QINIT

SUBROUTINE QMETRC

Purpose: QMETRC asks the user if he is using Metric or English units.

Format:
 CALL QMETRC(METRIC)
where: METRIC is 1 for metric units, 2 for English units. METRIC can then be used as a subscript to identify the correct scale factor ($Y=X*SCALE(METRIC)$)

LOGICAL FUNCTION QMORE

Purpose: QMORE suppresses output when the user has typed ahead. It may be used by the programmer so that local write statements suppress output consistently with the automatic suppression of the prompt routines.

Format:
Declare QMORE to be a logical at the top of the module
 LOGICAL QMORE
Use QMORE where output is directed to the screen between prompts
 IF(QMORE(IDUMMY)) write intermediate output
where: IDUMMY is a dummy argument.

Example:
 IF(QMORE(IDUMMY)) THEN
 WRITE(*,*) ' Summary of Program Status'
 WRITE(*,10) NUMA,NUMB,NUMC
10 FORMAT(/' Widgets A Widgets B Widgets C'/15,112,111)
 END IF

SUBROUTINE QPAUSE

Purpose: QMORE pauses and waits for user to hit return before continuing with program execution. It may be used to pause after output is displayed.

Format:

```
CALL QPAUSE
```

Example:

C List the table to the screen, 20 lines per screen

```

K = 1
DO 10 I=1,N
  WRITE(*,*) 'VALUE(' ,I,') =',VALUE(I)
  IF(K.GE.20) THEN
    CALL QPAUSE
    CALL QCLEAR
    K = 1
  ELSE
    K = K + 1
  END IF
10 CONTINUE

```

SUBROUTINE QSYSTEM

Purpose: QSYSTEM executes a system (monitor) command from Fortran, supported on multi-tasking operating systems such as VMS and UNIX.

Format:

```
CALL QSYSTEM(TEXT)
```

where TEXT is the system command.

Example:

C List the current files on the UNIX operating system

```
CALL SYSTEM('ls -lF')
```

SUBROUTINE QTERM

Purpose: QTERM asks the user to identify the terminal type and other attributes that may be needed to know how to clear the screen and apply graphics commands. On some systems a call to QTERM should be included in QINIT. COMMON block QTERML is used to pass baud rate, terminal type, and terminal size to other routines (such as QCLEAR).

Format:

```
CALL QTERM
```

APPENDIX D

CV FEM COMMAND SUMMARY

QUICK REFERENCE for FEM MESH GENERATION on COMPUTERVISION (CADD54)

Brant A. Ross - October 1984

This reference helps the rusty CADD54 user remember how to generate a finite element mesh. This information is not sufficient to train the novice user, but highlights the FEM commands that should be learned.

Note: Letters essential to CADD54 commands are capitalized. Other letters are given for clarity. The letter "d" indicates that a point on the screen is selected (digitized) using the graphics tablet.

Starting

1. Logging on: hit <ctrl> L, enter user name, password when prompted.
2. To run CADD54:
CADD5
3. To begin to define or edit a part or model:
ACTivate PART Partname -where Partname is the model name
ACTivate DRAWing I -to select name for the "drawing"
4. To define a view: (if you are starting from scratch)
DEFine VIEW TOP: X1 IY8.5
When editing an existing part, a view will be automatically selected.

Defining the Wireframe

5. To insert points using X, Y, and Z absolute coordinates:
INSert POInt: X value Y value Z value
6. To insert a line between two points:
INSert LINE: POI dd

FEM Definition

7. To initialize: (necessary before working with any FEM entities)
INITialize FEM
8. To insert a gridpoint (node) on a point:
INSert GPOInt: POInt d

9. To insert a QUAD element on 4 gridpoints:
INSert ELEMENT TYPE QUAD: <rtn> <rtn> <rtn> dddd
10. To insert gridpoints along a line with variable spacing:
GENerate GPOInt ON N num DISTANCE dis ARITHMETIC delt: d
where num is the number of gridpoints to be inserted, dis the initial distance between points, and delt the distance to be added to each successive interval.
11. To generate a mesh of Quad elements with regular spacing:
GENerate MESH AUTO TYPE QUAD: d;d;d;d <rtn> UNH j UNV k
where d;d;d;d represents digitizing the lines that make up the four sides of the region. The number j is the number of elements to be digitized along the first side digitized, and the number k is the number of elements along the second side.
12. To copy the existing mesh about a "mirror" for symmetric models:
GENerate MESH MIRROR <rtn> GPINCrement n: WINDOW d₁ d₂;
 d₃ d₄ d₅
where d₁ and d₂ define the window encompassing the mesh to be copied, and d₃, d₄ and d₅ define the plane the the existing mesh is "mirrored" around.
13. To merge redundant gridpoints caused by automeshing adjoining regions:
MERge GPOI: dd

Editing

14. To delete items, where name is the entity name (LIN, POI, GPOI, etc.)
DELEte ENTity: name d
15. To move items, where name is the entity name (LIN, POI, GPOI, etc.)
TRANslate ENTity: name d₁; d₂ d₃
where d₁ identifies the item, and d₂ and d₃ specify the local motion.

16. A construction plane may be needed when translating entities to an arbitrary location on a 3-D model. The construction plane sets the third dimension with graphics tablet input. Define a construction plane using three entities in the correct plane in the model:

DEFine CPL name: d d d

then activate that plane using the set command:

SET CPL name

delete old construction planes

DEL CPL name

Miscellaneous

17. To change zoom to get whole part within window:
ZOOM DRAWing ALL
18. To zoom to a specified area, digitizing the corners of the area are: dd
ZOOM DRA WIN: dd
19. To plot graphics on the local thermal printer:
PLOT HARdcopy

Neutral File

20. To create a neutral file with the Navy software (BYU system only)
RUN PROGRAM CADDSAUX.PUTNEUTRAL
The program will prompt you for a filename.
21. To access the tape drive and write the file to a tape:
ATTACH MT,TAPO
CONVERT file.ext :MT //FORMAT=(BLKSIZ=80,RECSIZ=80),CONVERT=C-E

This will write a text file in EBCDIC to the tape, in card image format. The destination computer must be able to read EBCDIC.

FLEXIBLE ENGINEERING SOFTWARE: AN INTEGRATED WORKSTATION
APPROACH TO FINITE ELEMENT ANALYSIS

Brant Arnold Ross

Department of Mechanical Engineering

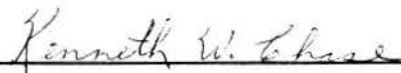
Ph.D. Degree, April 1985

ABSTRACT


One obstacle preventing more engineers from using finite element analysis (FEA) is the difficulty of transferring data between steps in the modeling process. A Fortran computer program, Rosetta.BYU, has been developed to open data paths between finite element preprocessors (mesh generators) and finite element analysis programs, using a custom data structure. It accept a neutral data files, Version 2.0 IGES data files, and Movie.BYU files for input/output. An application of Rosetta is described.

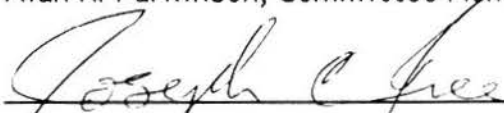
A general workstation manager program, Davinci.BYU, is reviewed that provides a support layer between the engineer and the operating system, organizes software and data files, and facilitates on-line documentation and demonstrations. Requirements of a good user interface are discussed and supporting software, Squire.BYU, is described. An application of this software in an industrial setting is described.

COMMITTEE APPROVAL:


Kenneth W. Chase, Committee Chairman


Steven E. Benzley, Committee Member


Alan R. Parkinson, Committee Member


Joseph C. Free, Department Chairman

