NASA CR-159042

(NASA-CR-159042) ATLAS, AN INTEGRATED STRUCTURAL ANALYSIS AND DESIGN SYSTEM. VOLUME 2: SYSTEM DESIGN DOCUMENT (Boeing Commercial Airplane Co., Seattle) 464 p HC A20/MF A01

N79-31621

CSCL 20K G3/39

Unclas 35732

ATLAS – An Integrated Structural Analysis and Design System

System Design Document

W. J. Erickson

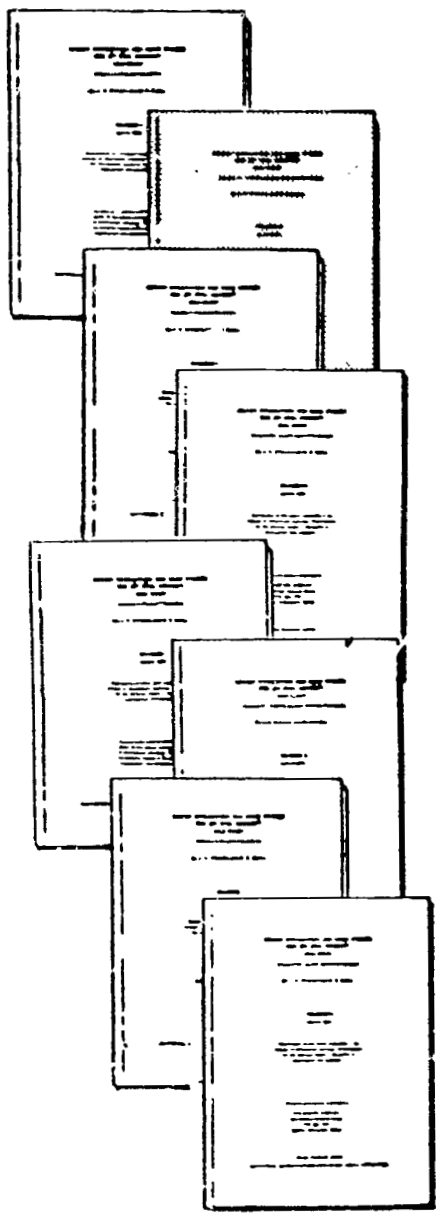
Boeing Commercial Airplane Company
Seattle, Washington 98124

Prepared for
Langley Research Center
under contract NAS1-12911

National Aeronautics and Space Administration

1979

ATLAS SYSTEM DOCUMENTATION



VOLUME I
ATLAS User's Guide
NASA CR-159041

VOLUME II
System Design Document
NASA CR-159042

VOLUME III
User's Manual-Input and Execution Data
NASA CR-159043

VOLUME IV
Random Access File Catalog
NASA CR-159044

VOLUME V
System Demonstration Problems
NASA CR-159045

VOLUME VI
DESIGN Module Theory
NASA CR-159046

VOLUME VII
LOADS Module Theory
Boeing Commercial Airplane Company
D6-25400-0101

VOLUME VIII
SNARF User's Manual
Boeing Computer Services
BCS-G0686

FOREWORD

Development of the ATLAS integrated structural analysis and design system was initiated by the Boeing Commercial Airplane Company in 1969. Continued development efforts have resulted in the release and application of several extended versions of the system to aerospace and civilian structures. Those capabilities of the current ATLAS version developed under the NASA Langley Contract No. NAS1-12911 include the following: geometry control, thermal stress, fuel generation/management, payload management, loadability curve generation, flutter solution, residual flexibility, strength design of composites, thermal fully stressed design, and interactive graphics. The monitor of this contract was G. L. Giles. The inertia loading capability was developed under Army Contract No. DAAG46-75-C-00[illegible].

This document is one volume of a series of documents describing the ATLAS System. The remaining documents present details regarding the input data and program execution, data management, the engineering method used by the computational modul and system-demonstration problems.

The key responsibilities for development of ATLAS have been within the Integrated Analysis/Design Systems Group of the Structures Research Unit of BCAC and the ATLAS System Group of the Boeing Computer Services (BCS) Integrated Systems and Systems Technology Unit. R. E. Miller, Jr. was the Program Manager of ATLAS until 1976 after which K. H. Dickenson assumed this position. The current ATLAS System is the result of the combined efforts of many Boeing engineering and programming personnel. Those who contributed directly to the current version of ATLAS are as follows:

E. F. Backman
G. N. Bates
L. C. Carpenter
R. E. Clemmons
R. L. Dreisbach
W. J. Erickson
S. R. Gadre
F. P. Gray
D. W. Halstead
E. B. Hansteen
B. A. Harrison
J. M. Held
M. Y. Hirayama
J. R. Hogley
H. E. Huffman
D. W. Johnson
A. S. Kawaguchi
C. D. Mounier
F. D. Nelson
M. C. Redman
R. A. Samuel
M. Tamekuni
G. von Limbach
S. O. Wahlstrom
R. A. Woodward
K. K. Yagi

ABSTRACT

ATLAS is a structural analysis and design system, operational on the Control Data Corporation 6600/CYBER computers. This document describes the overall system design, the design of the individual program modules, and the routines in the ATLAS system library. The overall design is discussed in terms of system architecture, executive function, data base structure, user program interfaces and operational procedures. The program module sections include detailed code descriptions, common block usage and random access file usage. The description of the ATLAS program library includes all information needed to use these general purpose routines.

CONTENTS

FIGURES

TABLES

1. INTRODUCTION

ATLAS is an integrated structural analysis and design system operational on the Control Data Corporation (CDC) 6600/CYBER computers. It is a modular system of computer codes integrated within a common executive and data base framework. The system is broad in scope in that its analytical capabilities support many different but related aeroelastic technological disciplines. Execution of selected computational modules is controlled by the user via a concise technically-oriented language. Input data are written in a problem-oriented language which offers versatile automatic data-generation capabilities. Data preprocessors minimize the amount of input data and flowtime required to define the structural problem. Additionally, data postprocessors allow selected data to be extracted, manipulated and displayed so as to enhance human judgment in the interpretation of design results.

This document contains detailed descriptions of the ATLAS system architecture, as well as information required to gain programming insight into the various processors. Documentation of user input data and execution control statements is given in reference 1-1, while the data management system is described in reference 1-2.

This document consists of three major subsections:

a) Sections 10 through 500 cover material relating to the system as a whole, including system architecture, maintenance, modifications, and operation.

b) Sections 600, 700, and 800 discuss in detail the code relating to the preprocessors, processors, and postprocessors, respectively.

c) Section 900 covers Interactive Control.

d) Section 1000 covers the ATLAS Library routines.

10. SYSTEM ARCHITECTURE

The design and development of the ATLAS program system have evolved in keeping with the following basic objectives:

a. Provide a common programming framework for related structural disciplines. This reduces the program development and maintenance costs, and increases the reliability of the aggregate code by coordinating the program management function.

b. Provide an efficient interface between the program and the program user. The usefulness of a computer system depends not on its potential capabilities but rather on how easily those capabilities can be understood and applied. Both concept and design of the program/user interfaces are defined with this in mind.

c. Provide a high degree of computer efficiency in problem solution. Emphasis on this objective will not only minimize computer costs and flowtime, but will also extend the range of feasible solutions.

d. Provide for open-endedness in program design allowing new capabilities to be added to the system.

The architecture of the ATLAS system is illustrated in figure 10-1. In the remainder of this section the basic elements of this design will be discussed.

10.1 PROGRAM MODULES

The code in ATLAS is organized in program modules which have the following characteristics:

a. A particular program module is executed by use of the ATLAS Control Language (sec. 200). There is one ATLAS control statement corresponding to the execution of each module.

b. A module performs a well defined engineering, mathematical, or clerical task.

c. A module consists of a primary and secondary overlay structure.

d. A module communicates with other modules via magnetic disk files, except for some limited executive table information.

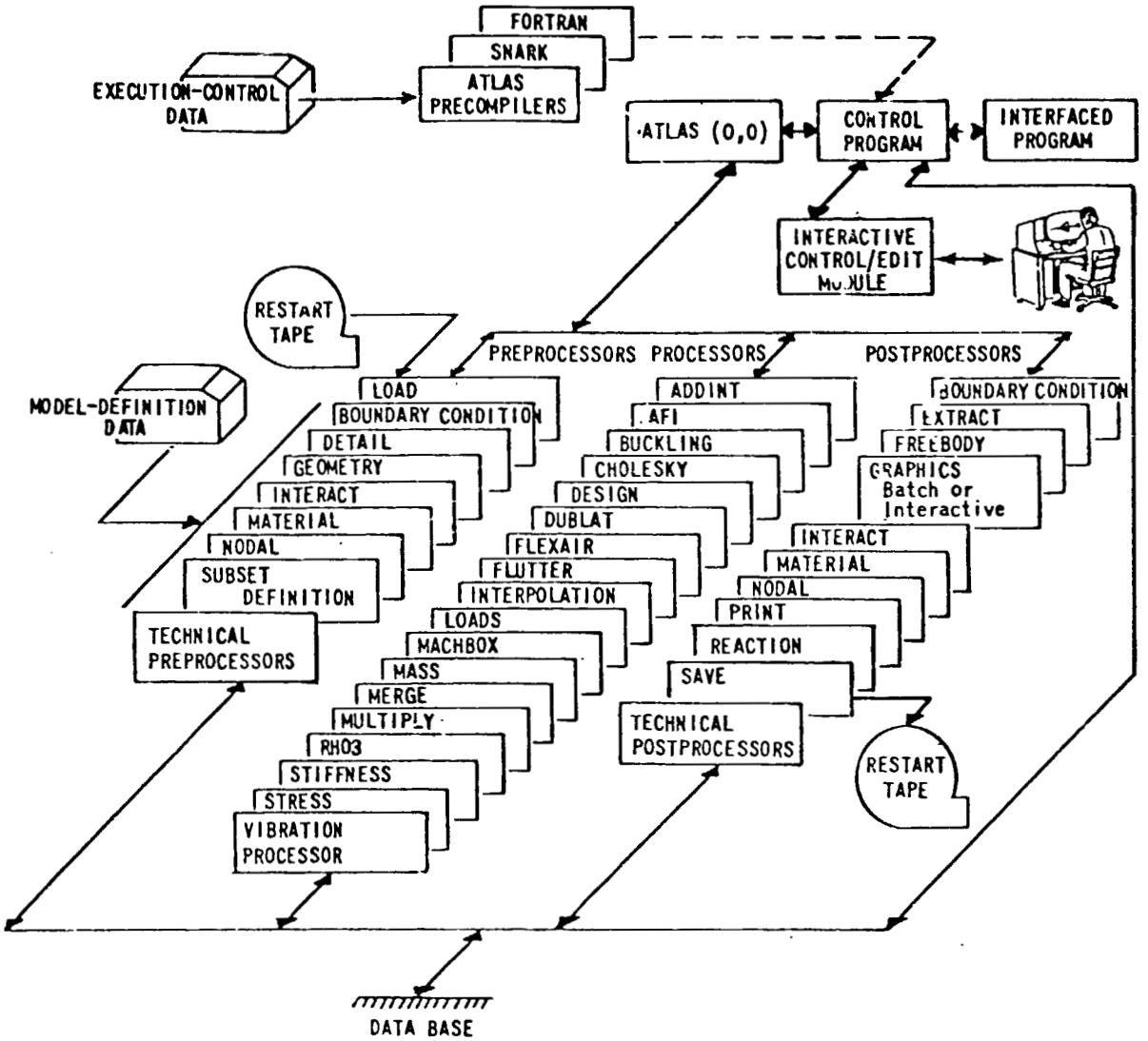


Figure 10-1. ATLAS System Modular Design

10.1.1 Classification

The ATLAS modules can be classified according to the general function they perform. These functions are indicated in figure 10-1 and are explained as follows:

a. Executive function - To monitor the sequence of module execution during a computer run. This function is carried out by the ATLAS main overlay and the Control module.

b. Technical function - A task which is related to a particular engineering theory or discipline. For instance, the Stiffness module contains the code which represents the finite element structural theory used in ATLAS. The AF1 module is based on a particular theory for calculating subsonic unsteady airloads.

c. Utility function - A task defining a major mathematical operation, normally involving large matrices. Examples of such tasks are solution of sets of linear equations, eigenvalue and eigenvector calculations, or performance of matrix additions and multiplications.

d. Preprocessing function - All input data for a problem is processed by one program module, the Preprocessor. The Preprocessor is organized in sub-modules which consist of one or more secondary overlays. Each sub-module will read and interpret the data for a particular technical module. Additional sub-modules are provided to handle general classes of input data.

e. Postprocessing function - All categories of output from ATLAS are generated by postprocessor modules. No results will be reported unless one of these modules has been executed. Currently there is one postprocessor module generating printed results, one preparing plot data and one which provides a program checkpoint capability. The print and graphics modules are organized similarily to the Preprocessor, i.e., they consist of sub-modules which generate results relative to a particular technical module.

10.1.2 Overview of Capabilities

An overview of the module capabilities is given below.

10.1.2.1 Executive modules

ATLAS(0,0)	Initializes loading of module overlays and closes system data files.
CØNTRØL	Defines the sequence of execution of system modules.
INTERACTIVE CØNTRØL	Supports interactive execution of a control program.

10.1.2.2 Technical and Utility modules

ADDINT	Adds and/or interpolates generalized air force matrices with respect to reduced frequency.
AF1	Generates subsonic incompressible-flow aerodynamic loads according to strip theory method for general planar surfaces.
BUCKLING	Calculates bifurcation buckling loads and mode shapes.
CHØLESKY	Solves systems of linear symmetric equations according to the Cholesky method.
DESIGN	Calculates margins of safety and resizes the structural model according to a fully stressed approach.
DUBLAT	Generates subsonic compressible-flow aerodynamic loads according to the Doublet Lattice method.
FLEXAIR	Generates the flexibility effects of truncated vibration modes for flutter.
FLUTTER	Solves the flutter equations.
INTERPØLATIØN	Establishes mode shape interpolation functions to be used by the aerodynamic airload generators.

LØADS	Prepares finite element static, inertial and thermal loads.
MACHBØX	Generates supersonic flow aerodynamic loads using the Machbox method with subdivision refinement for planar wings and tail surfaces with dihedral.
MASS	Generates finite element mass matrices for primary and secondary structure, fuel and payload.
MERGE	Assembles element matrices or substructure matrices to form gross structural matrices.
MULTIPLY	Performs matrix multiplication and addition of large matrices.
RHØ3	Generates subsonic compressible flow aerodynamic loads using a kernel function and assumed discrete pressure modes for planar surfaces with trailing edge controls.
STIFFNESS	Generates the finite element stiffness and stress matrices.
STRESS	Calculates the finite element stresses.
VIBRATIØN	Calculates frequencies and mode shapes of a structure undergoing free, undamped vibrations.

10.1.2.3 Preprocessor Modules

PREPRØCESSØR	Reads the ATLAS Problem Deck and loads the restart tape.

10.1.2.4 Postprocessor Modules

EXTRACT	Extracts data for processing by the Graphics module and some types of printing.
GRAPHICS	Supports all graphic and plotting capabilities of the ATLAS system.

PRINT	Prepares printed reports of results generated by technical and utility modules.
SAVE	Saves all problem data on a checkpoint file to enable subsequent problem restart.

10.2 USER INTERFACES

The user communicates with the ATLAS system in two ways: his Problem Deck defines his technical problem, while his Control Deck describes how the analysis shall be performed. These interfaces are described in the following two sections.

10.2.1 Problem Deck

The Problem Deck describes the technical problem to the program. The structure of the deck corresponds to the structure of the Preprocessor code, i.e., the deck is divided into sections, where each section is handled by a Preprocessor submodule. Each section is delimited by two cards of the type

```
BEGIN "name" DATA
END "name" DATA
```

where "name" is a unique identifier which coincides with the name of the corresponding Preprocessor submodule. For instance, 'STIFFNESS' is the "name" for input data to the Stiffness Preprocessor, and "SUBSETS" the "name" of the data to the preprocessor for subset definition. Each section may again be divided into subsections by delimiter cards

```
BEGIN "subsection" DATA
END "subsection" DATA
```

Subsection names need only be unique for each section.

The Problem Deck is processed by the input routine LØDAREC (sec. 900). LØDAREC is based on a free field input format. It also has the capability of performing internal data generation of the following types: 1) generate new data items within a record based on the original input data, and 2) generate one or more complete records from a parent record.

In addition to the data generating capabilities provided by LØDAREC, ATLAS input conventions allow for additional data generation to be performed as described in section 100.2.2, reference 1-1.

10.2.2 Control Deck

The Control Deck allows the user to:

a) define the sequence of module execution through use of the ATLAS Control Language.

b) add his own code, which may be either FØRTRAN or statements from the matrix interpretive SNARK language (sec. 10.4).

The Control Deck is translated into an overlay structure as described below. This structure will consist of at least a primary overlay containing the code for all of the ATLAS Control Language statements. It may also contain secondary overlays and subroutines which the user has added.

An ATLAS Control Language statement has the following general format:

Functional Descriptor (plist)

The "Functional Descriptor" will identify one, or several modules which will be executed as the statement is processed. The "plist" is a list of parameters which are passed onto the program modules as they are being executed. The parameters are used to achieve three purposes:

1) provide option control inside a module,

2) change default values set in the code, and

3) pass numeric or alphanumeric information.

The user may enrich the ATLAS language by defining his own "Functional Descriptors" based on the basic language capabilities provided by the ATLAS system (ref. 1-1, sec. 200.4).

The Control Deck is processed by a precompiler which translates the ATLAS Control Language statements into equivalent FØRTRAN code. Code which is not recognized as ATLAS statements will be transmitted unchanged to the precompiler output file. Thus, ATLAS statements may be intermixed with FØRTRAN and SNARK code.

10.3 PROGRAM FILES

Distinction must be made between four types of files: program overlay files, data communication files, special purpose files, and scratch files. The conventions which apply to these files are described in this section.

10.3.1 Overlay Files

Each program module, consisting of a primary/secondary overlay structure in absolute computer executable form, will have its own assigned overlay file name. This file name consists of 7 characters: the first 4 characters of the module name suffixed by the characters ØLF. Existing overlay file names are shown in table 10-1.

10.3.2 Data Communication Files

Data communication between program modules occurs on named random access disk files. Each module which generates global data, i.e., data to be used by other modules, will write this data on one specifically assigned file. The file name consists of 7 characters: the first 4 characters of the module name, suffixed by the characters RNF. Existing data communication file names are shown in table 10-1.

Data communication files are not available to a program module until they have been specifically "opened" by use of the ATLAS system routine FILEADD (sec. 900). This opening function will add the file name to the program RA+2 list and define a File Environment Table for the file. When a program module terminates execution the ATLAS system will 'close' all communication files that were opened. The closing function includes writing the random access tables out to the file and removing the file name from the RA+2 list. File closing is performed by the ATLAS system routine FILEDEL.

Table 10-1. Overview of Overlay Files and Communication Files

Module Name	Overlay File Name	Communication File Name
ADDINT	ADDIØLF	ADDIRNF
AF1	AF10ØLF	AF10RNF
ATLAS	ATLAS	
BUCKLING	BUCKØLF	BUCKRNF
CHØLESKY	CHØLØLF	CHØLRNF
DESIGN	DESIØLF	DESIRNF
DUBLAT	DUBLØLF	DUBLRNF
EXTRACT	EXTRØLF	EXTRRNF
FLEXAIR	FLEXØLF	FLEXRNF
FLUTTER	FLUTØLF	FLUTRNF
GRAPHICS	GRAPØLF	TAPE99
INPUT	INPTØLF	DATARNF
INTERPØLATIØN	INTEØLF	INTERNF
LØADS	LØADØLF	LØADRNF
MACHBØX	MACHØLF	MACHRNF
MASS	MASSØLF	MASSRNF
MERGE	MERGØLF	MERGRNF
MULTIPLY	MULTØLF	MULTRNF
PRINT	PRINØLF	
RHØ3	RHØ3ØLF	RHØ3RNF
SAVE	ØTPTØLF	
STRESS	STREØLF	STRERNF
VIBRATIØN	VIBRØLF	VIBRRNF

10.3.3 Special Purpose Files

ATLAS makes use of the following special purpose files:

TAPE5 = INPUT - Standard program input file

TAPE6 = ØUTPUT - Standard program output file

TAPE96 - Interactive message file

TAPE95
TAPE97
TAPE98
TAPE99
ØFFLINE - These five files are used by the plotting software

SYSRNF - File containing the User Matrix Index Catalog

SAVESSF - Sequential files used to
SAVESS1-SAVESS4 support the program checkpoint/
restart capability.

10.3.4 Scratch Files

Scratch files are defined and used locally inside a module. Data which have been generated on a scratch file will not be available to other modules. Scratch files are opened via the FILEADD subroutine. Scratch file names provided by the ATLAS system are:

SC00SSF - SC04SSF
SC00RIF - SC04RIF
SC00RNF - SC04RNF

After the module terminates execution, the ATLAS system will remove these scratch file names from both the program RA+2 list and the File Name Table of the operating system. Scratch files other than these must be specifically dropped

10.4 THE SNARK LANGUAGE - Core Management and I/Ø in ATLAS

SNARK is a matrix interpretive language which may be used to perform I/Ø functions, core management of data, and in-core matrix arithmetic. A description of the SNARK language capabilities is given in reference 10-1. SNARK statements are processed by a precompiler in the same way as described for the ATLAS language. This means that each SNARK statement will be replaced by equivalent FØRTRAN code. Also, SNARK and FØRTRAN code may be freely intermixed within the same program or subroutine.

In ATLAS the SNARK language is used:

1) to perform all I/Ø on the data communication files, and

2) to manage the core work area.

ATLAS uses the Blank Common concept of the CDC computer operating system to achieve dynamic core storage allocation. In this way the program field length may be adjusted to the requirements for each particular problem. The program work area is utilized to store arrays of problem dependent size. Administration of the area includes bookkeeping of existing arrays, allocation of space for new arrays, and purging of unneeded arrays. These functions are all taken care of by SNARK.

10.5 ERROR HANDLING

The ATLAS coding conventions distinguish between a warning condition and an error condition.

A warning condition occurs when the code detects an ambiguity in some data which can be resolved without user interaction. In this case a warning message will be issued, and the processing will proceed uninterrupted.

An error condition is entered either when the ATLAS code detects a fatal inconsistency in logic, or when an error is detected by the computer operating system. In the first case the module code will abort its execution and transfer back to the ATLAS main overlay. To provide a similar transfer in the second case, the system error table entries have been changed to provide the ATLAS main program as the recovery address. Not all error conditions detected by the operating system will allow return to the ATLAS code.

From the main overlay, transfer is made to the Control Module which either will perform a program exit, or branch to an error procedure defined by ATLAS statements. In this procedure the user may specify a number of tasks to be performed before terminating the run, such as executing the program checkpoint capability.

100. EXECUTIVE FUNCTION

The flow control function in ATLAS is performed by the main overlay and the Control Module. The Control Module defines the execution logic of a problem; it specifies the sequence in which the program modules will be processed. The primary function of the main overlay is to initiate loading of module overlays per instruction from the Control Module. Thus, the Control Module performs a task which is highly problem dependent, whereas the main overlay operates on a fixed and very limited logic. Dividing the program control function in this fashion serves the following important objectives:

a. The core requirements for the main overlay are kept to a minimum, thereby minimizing an important part of the system overhead cost.

b. Since the Control Module constitutes a primary overlay structure, the problem dependent part of the control function can be formed without reloading the rest of the system. Considering the size of the ATLAS system this feature becomes essential in assuring operating cost efficiency.

The following two subsections describe in more detail the flow control logic performed by the executive modules, including the contents of in-core tables which are part of the executive system.

100.1 PROGRAM FLOW

The ATLAS main overlay and the Control Module communicate via the common block MAINCAL described in section 100.2. Module execution is specified by using ATLAS statements in proper order in the Control Deck. Each statement will result in execution of one ATLAS module. When execution of an ATLAS statement is initiated in the Control Module, the following sequence of events will take place:

a. The name of the module overlay file is set in MAINCAL

b. The address of the next executable statement in the Control Module is set in MAINCAL.

c. Parameters to be passed to the module are set in common block CØNPARS according to the format described in section 200.3.

d. Flow returns to the main overlay, which initiates loading of the module overlay.

e. When the module terminates its processing the main overlay will close all communication files and drop all ATLAS scratch files. Communication file names and scratch file names will be deleted from the RA+2 list.

f. The main overlay will recall the Control Module to proceed with the next statement.

If a fatal error occurs during the execution the following additional events will take place:

g. A return to the main overlay will be processed with a normal exit from the module in the case of a module detected error or, to avoid program abort, to the beginning of the main overlay in the case of an operating system error. The latter exit is achieved by changing the system error tables with the CDC system routine SYSTEMC.

h. If the logical variable KERRDMP in common block ERRDMP has the value .TRUE. a complete core dump of the program field length will be provided on the output file.

i. The main overlay then recalls the Control Module which either will exit, or branch to the error procedure section when specified.

100.2 SYSTEM COMMON BLOCKS

100.2.1 /CØNPARS/ KCØNPAR, CØNPARS(30,2), KLABEL, LABEL(30)

CØNPARS is used to transmit parameters in the ATLAS statements from the control module to the other program modules.

KCØNPAR - The number of rows in CØNPARS utilized to transmit parameters associated with an ATLAS statement.

CØNPARS - Contains the parameters of an ATLAS statement. The format in which the parameters are stored in CØNPARS is described in section 200.3.

KLABEL - The number of words set in array LABEL

LABEL - The job identification specified on the PRØBLEM ID statement or the most recent CHANGE ID statement of the Control Module.

100.2.2 /ERRDMP/KERRDMP

ERRDMP is used to print core dumps from the main overlay in case of error exit. Thus, a core image may be provided before the Control Module overlay is loaded.

KERRDMP - Logical variable which is initialized to .FALSE. in the main overlay. A value of .TRUE. will result in the core dump when an error has occurred.

100.2.3 /KERRØR/KERRØR,KWARN,IPTU

KERRØR contains the system error counters

KERRØR - ATLAS system fatal error counter, initially set to zero in the main overlay. The variable will be increased by one for each error encountered by the ATLAS code.

KWARN - Warning counter, initially set to zero in the main overlay. The variable is increased by one each time a warning condition is encountered by the ATLAS code.

IPTU - Graphics file generation indicator. This variable is set by the Graphics module to indicate the combination of G/F, V/F, and P/F for a particular plotting device.

100.2.4 /KQBUFP/KQBUFP(2),SAVESSF,SAVESS1,SAVESS2, SAVESS3,SAVESS4,SC00SSF,SC01SSF,SC02SSF, SC03SSF,SC04SSF,DBASFIL

KQBUFP contains the file names of the system save files, sequential scratch files and the plot file. All names are set in the main overlay.

KQBUFP - Not in use

SAVESSF to SAVESS4 - System save files

SC00SSF to SC04SSF - System sequential scratch files

DBASFIL - Database file

100.2.5 /KQRNDM/KQINDX,KQRNDN,KQRNDI,KQRND,
ADDIRNF,AF10RNF,CHØLRNF,DATARNF,DESIRNF,
DUBLRNF,INTERNF,LØADRNF,MACHRNF,MASSRNF,
MERGRNF,MULTRNF,RHØ3RNF,STIFRNF,STRERNF,
CØNTRNF,VIBRRNF,FLUTRNF,FLEXRNF,EXTRRNF,
FLØRRNF,SEARRNF,SCALRNF,BUCKRNF,RESERVE(6)
SC00RNF,SC01RNF,SC02RNF,SC03RNF,SC04RNF,
SC00RIF,SC01RIF,SC02RIF,SC03RIF,SC04RIF

KQRNDM contains the file names of the data communication files and the random access scratch files. Values of all variables are set in the main overlay.

KQINDX - Random file index table space alloted for all random files

KQRNDN - Number of named random files

KQRNDI - Number of indexed random files

KQRND - KQRNDN + KQRNDI

ADDIRNF to BUCKRNF - System data communication files

RESERVE(6) - Reserve for future data communication files

SC00RNF to SC04RIF - System random scratch files

100.2.6 /MAINCAL/IFILE,IERRCT,ISAVERR,NEXTADR,KXINST,
CPTACC,ICYCLE

MAINCAL is used for communication between the main overlay and the Control Module

IFILE - Overlay file name of the current program module being executed. The name is set in Control Module.

IERRCT - Counter for operating system detected errors. The variable is set to zero as execution starts, and is incremented by one in the main overlay each time such an error occurs.

ISAVERR - Value of IERRCT upon last exit from the Control Module. Value is set in the Control Module.

NEXTADR - Statement number in Control Module where processing will resume when the module is recalled. Value is set in the Control Module.

KXINST - Identifier of the current ATLAS statement being executed. Value is set in the Control Module.

CPTACC - Accumulated central processor time. Value is updated in the Control Module routine ØVRRTN upon return from a program module.

ICYCLE - Frequency count of Control Module execution. Value is updated in the Control Module.

100.2.7 /MATTAB/NNBLK,NNLIM,NNPØS,NCBLK,NCLIM,NCPØS, MANNAM,MACNAM,SYSRNF,MAN(201), MAC(30,2),MACSUM

MATTAB is used to maintain the User Matrix Index Catalog. This catalog is stored in two parts, the Matrix Name Catalog and the Matrix Parameter Catalog, on the SYSRNF file as described below. The catalog is accessed by the ATLAS system routines GMISRT and GMLKUP.

NNBLK - Number of data blocks used to store the Matrix Name Catalog (MAN).

NNLIM - Size of MAN array minus one

NNPØS - Address of last entry in MAN array

NCBLK - Number of data blocks used to store the Matrix Parameter Catalog (MAC).

NCLIM - Row dimension of MAC array

NCPØS - Address of last entry in MAC array

MANNAM - Random index name of MAN array disc records

MACNAM - Random index name of MAN array disc records

SYSRNF - Name of file containing the User Matrix Index Catalog

MAN - Core resident part of the Matrix Name Catalog. This array contains the names of the matrices in the catalog and pointers to where the rest of the catalog can be found. Each entry of the array has the following format.

Bits 59-18: Matrix name (MN)

Bits 17-9: Data record number (RN) of MACNAM containing the catalog for MN.

Bits 8-0: Entry in record RN where catalog is stored

MAC - Core resident part of the Matrix Parameter Catalog. Each entry extends over two words, with the the following contents:

Word MAC(I,1):

Bits 59-36: The four leftmost characters of the file name where matrix is stored.

Bits 35-18: Matrix row dimension

Bits 17-0: Matrix column dimension

Word MAC(I,2):

Bits 59-57: Matrix type

Bits 56-54: Density type

Bits 53-36: Maximum data block size

Bits 35-18: Number of data blocks

100.2.8 /USERCØM/USERCØM(64)

USERCØM contains an array which is reserved for the user. The array is initialized to zero in the main overlay.

100.2.9 /FILE99/GRAFSQ1,GRAFSQ2,DIAGFIL,PLØTFIL

FILE99 is used to store the names of the files used by the GRAPHICS module.

100.2.10 /VIPER/VLEVEL,GENJØB,GENDATE,GENTIME

VIPER is used to store program version information

VLEVEL - Version number

GENJØB - Job name creating this version

GENDATE - Date of generation

GENTIME - Time of generation

100.2.11 /IACTIVE/IACTIVE,ØBJFILE,MØDE,EXTL,NPRØC,
PRØCLIM,PRØCFIL

IACTIVE is used to store control parameters while processing in the interactive control mode.

IACTIVE - Indicates whether a job is run in batch or online mode. Logical variable which is true while running in online mode.

ØBJFILE - Contains file name of the current command procedure.

MØDE - Contains the run mode for current command procedure (FULL or STEP).

EXTL - Logical variable which is set true when branching from the interactive control code to another ATLAS module.

NPRØC - Number of procedures currently in execution

PRØCLIM - Maximum number of nested procedures allowed

PRØCFIL - Array containing the names of procedures currently in execution.

200. ATLAS CONTROL LANGUAGE

The ATLAS Control Language constitutes the link between the program and the program user for transmitting processing requests to the various parts of the ATLAS system. This language is applicable to the Control Module only, to monitor the sequence of module execution and to pass parameter values. A description of the use of the language and the actual parameters which apply for the various computational modules is contained in reference 1-1, sections 200-258. This section describes the language syntax rules and the format for transmitting parameters to the program modules.

200.1 OVERVIEW OF ATLAS STATEMENTS

An ATLAS Control statement has the following general format:

Functional Descriptor (plist)

The "Functional Descriptor" identifies the blocks of code in the ATLAS system that will be executed. In most cases this will be one or more program modules. However, on occasion it can be code located in the Control Module itself. The "plist" represents the parameters passed to the code. Below is given a brief description of the Functional Descriptors defined for the ATLAS language. Terms which are enclosed in < > are optional. A term enclosed in { } indicates that several possible parameters are available. For a complete description see reference 1-1, section 200.

BEGIN CØNTRØL <MATRIX> PRØGRAM
END CØNTRØL PRØGRAM

These two statements will be the first and last statements, respectively, of the part of the Control Deck containing ATLAS statements. This block of code will be converted into the primary overlay of the Control Module.

PRØBLEM ID (text)

The "text" will appear as heading on all program printout. The text is stored in the common block /CØNPARS/ (sec. 100.2).

CHANGE ID (text)

The text defined by a PRØBLEM ID statement may be changed by this statement.

USER CØMMØN (Variable list)

This statement will generate the common block /USERCØM/ containing the parameters of the "variable list" (sec. 100.2).

ERRØR PRØCEDURE

This statement defines an entry point in the Control Module which will be executed if a fatal error condition has occurred.

READ INPUT (plist)
LØAD (type) (plist)

Both statements will generate code for branching to the preprocessor module. The READ statement is used to activate reading of the Problem Deck. The LØAD statement will cause the restart capability to be executed.

EXECUTE (module) (plist)

This statement sets up branching code to one of the Technical modules or one of the Utility modules. The name of the module is defined by "module".

PRINT (type) (plist)

This statement sets up code for branching to the print module.

SAVE (type) (plist)
INDEX FILES (plist)

Both statements will generate code for branching to the SAVE module. The SAVE statement will cause the restart capability to be executed. The INDEX statement will result in a printed report of the random access index tables.

PERFØRM {procedure} (list)

This statement causes a block of code identified by the name "procedure" to replace the statement itself. The replacement code can contain FØRTRAN, SNARK or ATLAS statements. The statement is nestable to any level. The parameter list of this statement has a different syntax than the "plist" of the statements above, as described in reference 1-1, section 200.

200.2 PARAMETER LIST SYNTAX

This section describes the syntax rules of the ATLAS statement parameter list identified by "plist" in the above section. Note that the parameter list of the PERFØRM statement does not belong to this category. The syntax definition is made in the Backus-Naur form.

<plist>::=<statement>|<complex statement>

<complex statement>::=<statement>|<statement>,
<complex statement>

<statement>::=<keyword>|<keyword>=<expression>|
<matrix expression>

<keyword>::=<alphanumeric literal>

<expression>::=<parameter>|<multiparameter>|
<list expression >|<matrix expression>

<parameter>::=<alphanumeric literal>|
<numeric literal>|FØRTRAN variable>

<alphanumeric literal>::=<character string other than
<FØRTRAN variable>>|
<n characters preceded by nH>|
<n characters preceded by nR>|
<n characters preceded by nL>

<numeric literal>::=<real scalar(F or E format)>|
<integer scalar>

<FØRTRAN variable>::=<simple or subscripted
variable name declared in
common block USERCØM>

<multiparameter>::=<(p_1,p_2,...,)>
where $p_{1,2}$::=<parameter>

<list expression>::=<n_1 TØ n_2>|<n_1 TØ n_2 BY n73>
where n_1,n_2::=<numeric literal>|
<alphanumeric string appended
by numeric characters>, and
n_3::=<numeric literal>

<matrix expression>::=<[<matrix name><operator>
<FØRTRAN variable><delimiter>
<numeric literal>]>

Two elements of the same type,
i.e., <matrix name>,<operator>
can not follow each other
immediately in the matrix
expression.

<matrix name>::=<ATLAS User Matrix name>|
<ATLAS User Matrix name appended by the
3-character string (char)> where
char is any FØRTRAN character

<operator>::= +|-|*|/

<delimiter>::= (|)|= Exceptions for ( ) are described
in the definition of <matrix
name> above.

200.3 FORMAT OF PARAMETER LIST IN CØNPARS

The ATLAS statements which are discussed in this section are: READ, LØAD, EXECUTE, PRINT, PLØT, SAVE, INDEX, PURGE and RENAME. All of these statements are decoded during the precompilation process and stored in the array CØNPARS of the common block of the same name (sec. 100.2). In the following, the format of the CØNPARS array is described.

200.3.1 General Rules

a. Using the notation of section 200.2 the general format of an ATLAS statement is

Functional Descriptor (<statement>$_1$,<statement>$_2$...)

The "Functional Descriptor" will always be a two-word combination, where "Functional" represents the first word and "Descriptor" the second. The general layout in CØNPARS is:

Row	Column 1	Column 2
1	FUNCTIONAL	0
2	DESCRIPTOR	0
3	<statement>$_1$	
4	<statement>$_2$	
-	---	

If the word represented by FUNCTIØNAL or DESCRIPTØR is more than 10 characters long, only the 10 leftmost characters will be stored.

The <statement>'s are stored in the sequence they appear in the ATLAS statement. The storage format depends on the type of <statement> and will be described in subsequent subsections.

b. <alphanumeric literal> is stored in the same way as FØRTRAN variables. Literals with more than 10 characters will be truncated.

c. <numeric literal> is stored as an integer or a real number as specified, except in matrix expressions where the type will be forced to real.

d. <matrix name> is stored in H-format

200.3.2 <statement>::=<keyword>=<multiparameter>

Example: SET=2

The format in CØNPARS is:

Row	Column 1	Column 2
-	--	--
i	SET	2
-	--	--

200.3.3 <statement>::=<keyword>=<multiparameter>

Example: KVALUE=(1,2,3,7)

The format in CØNPARS is:

Row	Column 1	Column 2
-	--	--
i	KVALUE	1
i+1	KVALUE	2
i+2	KVALUE	3
i+3	KVALUE	7
-	--	--

200.3.4 <statement>::=<matrix expression>

Example: [(A+L*B)X=0]

The format in CØNPARS is:

Row	Column 1	Column 2
-	--	--
i	*EXP	6
i+1	(	A
i+2	+	L
i+3	*	B
i+4	)	X
i+5	=	0
-	--	--

A matrix expression is notified in CØNPARS by setting of a keyword *EXP in the first column of the array. The second column of the same row will point to the relative row where next <statement> is stored.

200.3.5 <statement>::=<keyword>=<matrix expression>

Example: A=[-B+C (T) *D-2.4*E]

The format in CØNPARS is:

Row	Column 1	Column 2
-	--	--
i	*EXP	8
i+1	A	=
i+2	-	B
i+3	+	C (T)
i+4	*	D
i+4	-	*SCALAR
i+6	2.4	*
i+7	E	0 - - - 0
-	--	--

A numeric value in the expression is notified by preceeding it by the keyword *SCALAR.

200.3.6 <statement>::=<list expression>

Example: A=2 TØ 4, B=M3 TØ M9 BY 3

The format in CØNPARS is

Row	Column 1	Column 2
-	--	--
i	A	2
i+1	TØ	4
i+2	B	M3
i+3	TØ	M9
i+4	BY	3
-	--	--

300. ATLAS PROGRAMMING METHODS

This section describes the various ATLAS requirements to be satisfied when:

a. adding a new processor to the system, or

b. modifying existing code

300.1 ADDING A PROCESSOR

If the addition of the ABCDEFG Processor to the ATLAS system is hypothesized, the following steps must be taken:

a. System tapes and files, section 501, and catalogued procedures, section 503, must be modified to reflect the existence of files ABCDØLD, ABCDREL, and ABCDØLF.

b. The ATLAS precompiler must be changed to accept the EXECUTE ABCDEFG statement. The strings MNAMES and FNAMES in subroutine INITIAL must be modified to contain the names ABCD and ABCDRNF, respectively.

c. The system common block /KQRNDM/ must be updated to reflect the presence of ABCDRNF.

d. The dummy deck DUMABCD must be added to the DUMMØLD library. This deck will serve as the structural skeleton of the module, as described in section 502.3. Technical module primary overlays will be given as ØVERLAY(ABCDØLF,1,0). Secondary overlay numbers are assigned by the programmer.

e. The source library file ABCDØLD should be produced in an UPDATE creation run. Each subroutine must be fully contained in a deck of the same name. The organization of the ØLDPL is given in section 502.1. CØMDECKS for the system common blocks, and decks for routines NØBØØJM and NØFØRTN, should be formed.

In addition to the above specific steps, the new code must adhere to the general ATLAS guidelines of section 300.2 pertaining to documentation, coding practices, etc.

300.2 PROGRAMMING GUIDELINES

ATLAS guidelines are presented here for coding practices, error handling, updating, and documentation.

300.2.1 Coding Practices

No effort is made here to teach good programming technique, but certain practices have proven to be most effective in the ATLAS system.

a. SNARK usage: The greatest benefits from SNARK usage lie in its matrix I/Ø and core management functions. The most common structure for an ATLAS module is a FØRTRAN primary overlay, driving SNARK secondary overlays, which in turn drive the working routines, again usually in FØRTRAN. Because SNARK statements tend to generate inefficient code, effort should be made to avoid excessive use of SNARK language statements, such as SETELEM or SNARK CALL, to do the real work of the module. In particular, SNARK statements in DØ loops should be eliminated.

b. File manipulation: Random access files for SNARK matrix I/Ø should be opened using routine FILEADD (sec. 900). FØRTRAN I/Ø files should be opened with FETEDIT. Any scratch files other than the system scratch files should be dropped at the end of their usefulness using the system routine DRØPFIL.

c. Coding: Do not include any ØVERLAY cards in the code. This will be taken care of by the dummy deck.

Use standard ATLAS CØMDECKS for any system common blocks.

Comments should thoroughly describe the executable code. Variables in calling sequences and common blocks must be identified.

Do not use CALL EXIT, CALL FLUSH, or STØP statements.

300.2.2 Error handling

ATLAS attempts to trap all possible errors, and to provide a meaningful message to the user when one occurs. The following methods are directed to that end.

a. Common block /KERRØR/: Variable KERRØR is the count of fatal errors, KERRØRS (or KWARN), the count of non-fatal errors. They should be incremented each time an error of the appropriate type is encountered and a message written, containing the word "error" for fatal errors, "warning" for non-fatal errors. In preprocessors, fatal errors should not halt execution. In postprocessors, warnings only should be issued, and processing allowed to continue in the program. The reasoning behind these conventions is that by completing all preprocessing, more errors per run can be detected and corrected, while post processing errors do not necessarily mean an execution is without usefulness, and should not end it.

b. SNARK errors: Use the SNARK SETERR command to trap matrix errors. It is also often useful to use the INVENTØRY capability of SNARK.

System mode errors as the result of bad data or incorrect parameters should be avoided through the use of checks within the code.

300.2.3 UPDATE procedures

ATLAS system integrity is maintained via the UPDATE program. It is essential that consistent practices be used in this process.

a. Whenever making a system update, comments should be included in the update giving author, date, and purpose, and similar comments should go into the routines being updated.

b. When adding decks or CØMDECKs, *ADDFILE should be used. They should be inserted alphabetically per section 502.1. Deck names should match the name of the routine being added.

c. Cards submitted for system updates must have been run previously in checkout. Each deck of UPDATE directives submitted must be headed by the following two cards:

```
*IDENT   XXXXnnn
*COMPILE NOBOOJM,NOFORTN
```

where XXXX are the first 4 characters of the module name, and nnn is the number of the current system update. Any adding or purging of decks must follow the correction set, if any, and should be accompanied by appropriate updates to the Dummy OLDPL, headed by:

```
*IDENT  DXXXXnn
```

300.2.4 Documentation

ATLAS documentation is intended to clarify code, facilitate program usage, and describe code and theory developments. Each ATLAS programming task includes both code and documentation.

a. Source code comments: These comments have the basic purpose of making the code easier to read. In addition, each program or routine should have a set of comments giving purpose, [illegible] date, update and checkout history, file usage, and a description of variables in common blocks and calling sequences.

b. ATLAS documents: In part, these consist of the present document, the Users Manual, reference 1-1, and the File Catalog, reference 1-2.

c. Checkout documentation: This documentation occurs in the form of comments in the checkout decks, and is described in section 504.

400. DATA MANAGEMENT

This section describes data management of the program communication files, which constitute the data base for all ATLAS information. Since scratch files are defined by each program module according to need and then released before another module is entered, the data management of these files is left up to each module.

On the communication files, there exist three categories of information here described as table matrices, element or nodal matrices, and gross matrices.

a. The table matrices contain various types of reference data: coordinate tables, nodal correspondence tables, finite element tables, etc. They are of such limited size that they can be fully contained in the core memory. In order to keep core requirements to a minimum, the tables are designed in a compact format by packing information in each 60 bit word. This approach is based on the assumption that it is less expensive to unpack information than to execute on a larger field length.

b. Element or nodal matrices contain properties relevant to each structural element or node, i.e., finite element stiffness and mass matrices, nodal loads matrices, etc. These matrices normally require a subdivision into several logical records on disk. The core memory will only be able to contain one or a few records at a time. Word packing is used to a limited extent to store various control information related to each element or nodal matrix.

c. Gross matrices contain properties which relate to a complete analysis model, i.e., assembled stiffness, mass or loads matrices. The matrices can generally be very large and therefore, have to be subdivided into numerous logical records in order to be processed in core. Very little control information is required for these matrices; consequently, word packing is not used.

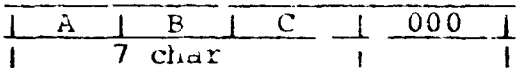
400.1 MATRIX NAMING

Each matrix on the data communication files is identified by a name which is either predetermined in the program code, or defined by the user at execution time. Generally all table matrices and the element and nodal matrices will have fixed names, whereas most gross matrices will have user defined names. Because of the

user's involvement in their naming, the gross matrices are called user matrices. The random index names associated with data records for fixed name matrices or user matrices are formed with the matrix name as a basis, as described in the following two subsections.

400.1.1 Fixed Name Conventions

The random index name for a fixed name matrix contains from 1 to 7 characters which are left adjusted in the word, followed by zero bits. In general the index name is composed of three fields A, B and C with contents as described below. The length of each may vary from one matrix to another. Unused parts of a field are zero filled.



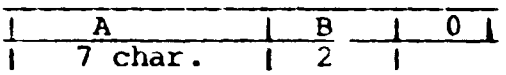
A - The matrix name consisting of an alphanumeric string starting with a letter. For the file DATARNF the first letter is associated with the Preprocessor submodule that generates the matrix, as described in table 600-1. For all other files, there are no restrictions as to choice of matrix name.

B - This field is used to distinguish between separate logical records of the same matrix. It contains the display code representation of the record number. This field is omitted if the matrix always consists of one logical record.

C - Each character of this field is used to index the matrix with respect to a certain condition; for instance, the data set number, the stage number, the Mach number, etc. With a display code representation of each condition (numbers and letters only are permitted), a maximum of 44 conditions may be accommodated.

400.1.2 User Name Conventions

The random index name for a user matrix consists of nine left adjusted characters followed by six zero bits. The name is composed of two fields A and B with contents as described below. Unused parts of a field are filled with zero bits.



A - The user defined matrix name consisting of from 1 to 7 alphanumeric characters.

B - This field is used to distinguish between separate logical records of the same matrix. It contains the binary representation of the record number.

400.2 MATRIX ACCESSING

Reading and writing of matrix records on the data communication files are performed by the SNARK statements RDMATRB and WTMATRB, reference 10-1. Before the actual transfer can be made the proper record index name and file name have to be identified. Depending upon the type of matrix to be accessed, i.e., fixed name or user matrix, the following approaches are used:

a. In the case of a fixed name matrix the name of the communication file must be known at programming time. The sub-fields A, B and C of the random access index are determined as follows: Field A must be defined at programming time. Field C is set by masking in the actual conditions that apply for the particular case. Field B is updated sequentially for each matrix record being transferred using the ATLAS library routine INCR. Existence of a particular record should be tested with the library routine IFRREC.

b. User matrix names are managed via an index catalog which is maintained on the file SYSRNF. The catalog consists of two parts: the matrix name catalog MAN, and the matrix parameter catalog MAC. At any point during processing, the matrix name catalog will contain one entry for each user matrix defined with the contents:

   1. User matrix name

   2. Pointer to associated entry in the parameter catalog

   The parameter catalog contains:

   1. File name where matrix is stored.

   2. Matrix row dimension.

3. Matrix column dimension.

4. Matrix type indicator specifying that the matrix is stored according to one of four possible formats - rectangular, symmetric, diagonal or null matrix format.

5. Maximum record size used.

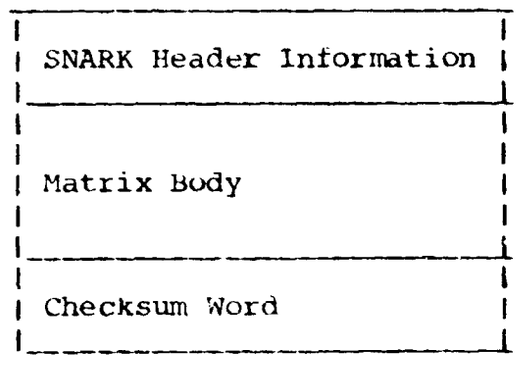
6. Number of records used.

Access to the two catalogs is performed by the ATLAS library routines GMISRT and GMLKUP (sec. 900) via the common block MATTAB (sec. 100).

Fields A and B of the random index name are determined as follows: Field A is supplied as input from the ATLAS statement for the appropriate program module. Field B is updated sequentially for each matrix record being transferred using the ATLAS library routine INCRB.

400.3 SNARK MATRIX STORAGE FORMAT

All tables and matrices on the ATLAS communication files are stored according to the SNARK matrix format. In this format each logical record has the following structure:



The characteristics of each of these record components are described in the subsequent subsections.

400.3.1 SNARK Header Information

The SNARK header information, as shown in table 400-1, is stored and retrieved by SNARK statements DEFID,EXTID,EXTAUX, and STØAUX (ref. 10-1).

Table 400-1 SNARK Matrix Header Information

Word	Contents
1	Contains MATRIX1 in H-format
2	SNARK matrix type indicator (H-format) which is one of the following: REAL - Matrix body contains floating point numbers only. Zero elements will not be stored on disk. MIXED - Matrix body contains floating point, integer and/or alphanumeric information. Zero words will be transferred to disk. DIAGØNAL - Matrix body contains a diagonal matrix stored as a vector. NULL - Matrix body contains zero elements only. No matrix elements are stored on disk.
3	Submatrix row dimension (=m)
4	Submatrix column dimension (=n) Note that the product m*n should be equal to the length of the matrix body. m and n will therefore often not be the true dimension of the matrix.
5	Pointer to word 6+k
6 : 5+k	Matrix name, with the first word being the date of the matrix generation (k words total) Note that matrix name is not the same, or associated with, the matrix index name (see ref. 10-1).
6+k : 15+k	Ten words containing the user's auxiliary ID. In ATLAS the first two words of the ID are reserved for the following information: ID1 - Contains the communication file name ID2 - Contains the matrix index name This information is generated and used by the checkpoint/restart capability.
16+k : 25+k	Not used

The SNARK matrix type specified in word 2 of the header is used by the I/Ø routines to determine whether or not the contents of the matrix body can be compressed when transferred to disk, or expanded when transferred to core storage. It has no relation to the matrix types defined for user matrices in section 400.2, which will be further described in the following section.

400.3.2 Matrix Body

The fixed name matrices do not have a common storage format for the matrix body. Reference 1-2 should be consulted in each particular case to explain the contents.

The type of user matrix, as identified in the user matrix catalog, reflects a special format of the SNARK matrix body. These formats for rectangular, symmetric, diagonal and null matrices are described below.

a. Rectangular Matrix

X_1	...	Xn	X_1	...	Xn	...	Xn

X_1-Xn: The elements of a row of a matrix. Rows are stored consecutively, each row being contained completely within a data record.

b. Symmetric Matrix

I	J	X_1	...	Xd	I	J	X_1	...	Xd

I: Row number in matrix

J: Column number of first non zero matrix element in row I

X_1-Xd: First non zero element in row I through the diagonal matrix element. Rows are stored consecutively, each row up to and including the diagonal element being completely contained within a data record.

c. Diagonal matrix

The diagonal matrix elements are stored as a vector.

d. Null Matrix

Each null matrix must be represented by one data record in order to be recognized by the checkpoint/restart facility. The contents of the matrix body are not interrogated.

400.3.3 Checksum Word

The checksum word is set and interrogated by the SNARK I/Ø routines to check the integrity of the data transfer. The word contains the logical sum of the complete matrix record, except word 6 (the date) of the SNARK header. The checksum is computed by the BCS library routine LCKSUM.

500. ATLAS SYSTEM OPERATION

The ATLAS system is maintained on both tape and disk. The following sections discuss the methods used to ensure system integrity, to standardize checkout procedures, and to gain access to either ATLAS system.

501. ATLAS SYSTEM FILES

The ATLAS 4.0 system is kept on two magnetic tapes, the relocatable tape and the absolute tape, and two sets of permanent files.

501.1 RELOCATABLE TAPE

The relocatable tape contains all of the files necessary for ATLAS code development (see sec. 502): the UPDATE program library (ØLDPL), relocatable files, and dummy routines for each processor, all system library routines, and the precompilers (for tape structure, see sec. 505.1).

501.2 ABSOLUTE TAPE

The absolute tape contains all of the files necessary for an ATLAS execution: the absolute code for each processor, the system libraries, and the precompilers (for tape structure see sec. 505.2).

501.3 ATLAS PERMANENT FILES

One set of ATLAS permanent files is a developmental version of the code from the latest absolute tape. The files are generally replaced after every update to the system but are subject to change and do not always reflect a single absolute tape. The other set of permanent files is from the released absolute tape and is intended for production use.

502. ATLAS SYSTEM MAINTENANCE

Modification of the ATLAS code is based on the files on the relocatable tape, which contains an UPDATE program library (ØLDPL), the relocatable decks in load order, and a dummy deck for each module.

502.1 ØLDPL

The UPDATE ØLDPL contains a source code deck for each routine and CØMDECKs for most of the common blocks in the processor. The deck names coincide with the names of the subroutines and the common blocks. Only one source deck is used for each routine, despite multiple uses of the routine in the processor. ØVERLAY cards are not needed on programs since the dummy routines supply all such cards.

The decks in the ØLDPL are ordered alphabetically in three groups: CØMDECKs, SNARK decks, and FØRTRAN routines. Two special decks, NØBØØJM, which follows the SNARK decks and NØFØRTN, which follows the FØRTRAN decks, are written to the CØMPILE file each time a module is updated. NØBØØJM contains a dummy SNARK program and a *CWEØR 15 card, which writes an end-of-file mark on the CØMPILE record. NØFØRTN is a dummy FØRTRAN routine. This procedure results in the CØMPILE file always having two files, one of SNARK routines, one of FØRTRAN routines.

502.2 RELOCATABLE FILES

Each relocatable file contains all of the ØVERLAY cards and relocatable code necessary to load a module (excluding code from the libraries). Thus, if a routine appears in two overlays of a processor, it appears twice in the relocatables.

502.3 DUMMY ROUTINES

Each module has a set of dummy routines associated with it consisting only of ØVERLAY, PRØGRAM, SUBRØUTINE, and END cards. This deck determines the overlay and subroutine structure of the module code. It is used in conjunction with the CØPYLL utility to build the relocatable file for the module.

502.4 MAINTENANCE PROCEDURE

Changes to an ATLAS processor code are made with updates. A system update, involving several processors, writes a new relocatable tape, loads the entire system and writes a new absolute tape.

Each system update is numbered sequentially (UD01, UD02,...). This number is reflected in the UPDATE IDENT cards. Dummy UPDATE IDENTS are prefixed by a D. The name portion of the UPDATE IDENT is taken from the name of the module, as are the names of the ØLDPL, relocatable and absolute files.

For processor XXXXaaaa:

XXXXØLD	-	Update library file
XXXXREL	-	Relocatable file
XXXXØLF	-	Absolute file
*IDENT XXXXnn	-	Update to code, system update nn
*IDENT DXXXXnn	-	Update to dummies, system update nn

502.5 ATLAS AND DUMMY

The (ATLAS,0,0) overlay of ATLAS is referred to as the real ATLAS. The relocatable file for this overlay is ATLASR and its source code is in ALIBØLD. A special procedure is used when loading ATLASR to insure that all CDC Record Manager routines needed throughout the ATLAS system are loaded in the (ATLAS,0,0) overlay. The resulting full load map is then scanned by a program called AUTØDUM which generates compass source code. This source code is compiled and forms DUMMYR. When DUMMYR is loaded to form the (DUMMY,0,0) overlay, the resulting absolutes duplicate the (ATLAS,0,0) overlay with identical common blocks, routines, and entry points. This enables overlays loaded with DUMMYR to execute with the (ATLAS,0,0) main overlay which was loaded previously. All ATLAS primary and secondary overlays including the CØNTRØL overlay are loaded with DUMMYR. Whenever an update is made to any routine which is loaded in the 0,0 overlay, DUMMYR must be remade to reflect changes in routine lengths and entry points.

502.6 VERSION IDENTIFICATION

Information identifying the version of ATLAS being executed is stored in the VIPER common block (sec. 100.2.10). The routine VERSIØN in VLIB sets up the values in VIPER. VLIB is loaded with the control program. Each time the ATLAS system is loaded using the LØADREL procedure on CATM (sec. 503.2.8), the program TRACE generates source code for VERSIØN. This source code is compiled and the VLIB library is made from the resulting relocatables of VERSIØN.

503. CATALOGUED PROCEDURES

Control card procedures are used with the CALL utility to simplify the running of a job. The ATLAS procedures are on file CATM, and are used for ATLAS program executions, and maintenance and checkout runs. Input and output refer to files required or produced by the procedure. Reserved names are those files or registers which will be modified in the procedure.

503.1 RUN PROCEDURES

ATØDISK, ATTACH, CØNTRØL, DRØPRNF, FILES, KØNTRØL, SHØRT

503.1.1 ATØDISK

PURPOSE - ATØDISK copies the absolute tape onto separate disk files.

USAGE - CALL(CATM,ATØDISK)

INPUT - ATLASMT, the ATLAS absolute tape.

OUTPUT - All files named in section 505.2 except CATM.

RESERVED NAMES - None

503.1.2 ATTACH

PURPOSE - ATTACH gets the permanent file version of ATLAS.

USAGE - CALL(CATM,ATTACH)

INPUT - None

OUTPUT - The ATLAS permanent files

RESERVED NAMES - None

503.1.3 CØNTRØL

PURPOSE - CØNTRØL compiles and loads an ATLAS control program containing no SNARK statements.

USAGE - CALL (CATM,CØNTRØL) or
CALL(CATM,CØNTRØL(INPUT=infile,
ØUTPUT=outfile))

INPUT - A file containing the user's ATLAS control program. Default is INPUT.

OUTPUT - a. Overlay file CØNTRØL containing the absolutes of the control program.
b. A file containing the listings from the compiling and loading. Default is ØUTPUT.

RESERVED NAMES - CREL, PEL, QX8, QX13, QX18, F0, F1, F2, F3, F4

503.1.4 DRØPRNF

PURPOSE - DRØPRNF returns the data communication files to facilitate multiple ATLAS runs in one job.

USAGE - CALL(CATM,DRØPRNF)

INPUT - None

OUTPUT - None

RESERVED NAMES - None

503.1.5 FILES

PURPOSE - FILES gets the development version of ATLAS from permanent file.

USAGE - CALL(CATM,FILES)

INPUT - None

OUTPUT - The ATLAS development version permanent files

RESERVED NAMES - None

503.1.6 KØNTRØL

PURPOSE - KØNTRØL compiles and loads an ATLAS control program containing SNARK statements.

USAGE - CALL(CATM,KØNTRØL) or
CALL(CATM,KØNTRØL(INPUT=infile,
ØUTPUT=outfile))

INPUT - A file containing the users ATLAS control program. Default is INPUT.

OUTPUT - a. Overlay file CØNTRØL containing the absolutes of the control program.

b. A file containing the listings from the compiling and loading. Default is ØUTPUT.

RESERVED NAMES - CREL,REL,QX8,QX13, QX18,F0,F1,F2,F3,F4,F5

503.1.7 SHØRT

PURPOSE - SHØRT gets a short portion of the permanent file version of ATLAS

USAGE - CALL(CATM,SHØRT)

INPUT - None

OUTPUT - The ATLAS permanent files for control programs, input, output, and print.

RESERVED NAMES - None

503.2 MAINTENANCE PROCEDURES

ATØFILE, ATØTAPE, DRØPABS, DRØPREL, DUMUDT, FICHE, LIBRARY, LØADREL, REWABS, REWREL, RTØDISK, RTØFILE, RTØLGØ, RTØTAPE, UPDATE, PRECUP, FULLUP

5.3.2.1 ATØFILE

PURPOSE - ATØFILE copies the absolute tape files to one file, ATLASMT.

USAGE - CALL(CATM,ATØFILE)

INPUT - ATLAS absolute files (sec. 505.2)

OUTPUT - ATLASMT

RESERVED NAMES - None

503.2.2 ATØTAPE

PURPOSE - ATØTAPE copies the absolute files to tape.

USAGE - CALL(CATM,ATØTAPE)

INPUT - ATLAS absolute files (sec. 505.2)

OUTPUT - File MT on tape and a catalog of MT on ØUTPUT

RESERVED NAMES - None

503.2.3 DRØPABS

PURPOSE - DRØPABS returns all of the ATLAS absolute files.

USAGE - CALL(CATM,DRØPABS)

INPUT - None

OUTPUT - None

RESERVED NAMES - None

503.2.4 DRØPREL

PURPOSE - DRØPREL returns all of the files from the relocatable tape except CATM

USAGE - CALL(CATM,DRØPREL)

INPUT - None

OUTPUT - None

RESERVED NAMES - None

503.2.5 DUMUDT

PURPOSE - DUMUDT is used to update the dummy decks.

USAGE - CALL(CATM,DUMUDT(REL=XXXXREL, INPUT-infile,ØUTPUT-outfile)) where XXXX are the first four letters of the module name.

INPUT - Update correction set. Default file - INPUT.

OUTPUT - a. Updated DUMMØLD and relocatable file and
b. Listing of the changed dummy deck on "outfile". Default file ØUTPUT).

RESERVED NAMES - ØLDPL, CØMPILE, CØR, ØUT, NEW, R1

503.2.6 FICHE

PURPOSE - FICHE writes the ØUTPUT file to tape for later use as microfiche input.

USAGE - CALL(CATM,FICHE)

INPUT - ØUTPUT file

OUTPUT - File FT on tape

RESERVED NAMES - F0

503.2.7 LIBRARY

PURPOSE - LIBRARY produces ATLASL from ALIBREL and SNKLREL, CLIB from CLIBREL, ILIB from ILIBREL, and SNKLIB from SNKLREL.

USAGE - CALL(CATM,LIBRARY)

INPUT - ALIBREL, CLIBREL, SNKLREL, ILIBREL

OUTPUT - ATLASL, CLIB, ILIB, SNKLIB

RESERVED NAMES - LIB

503.2.8 LØADREL

PURPOSE - LØADREL is used to load the ATLAS relocatables during an update.

USAGE - CALL(CATM,LØADREL)

INPUT - ATLAS relocatables on LGØ and ATLASL

OUTPUT - ATLAS absolute files

RESERVED NAMES - None

503.2.9 REWABS

PURPOSE - REWABS rewinds the files from the absolute tape.

USAGE - CALL(CATM,REWABS)

INPUT - ATLAS absolute files

OUTPUT - None

RESERVED NAMES - None

503.2.10 REWREL

PURPOSE - REWREL rewinds the files from the relocatable tape.

USAGE - CALL(CATM,REWREL)

INPUT - All files from the relocatable tape

OUTPUT - None

RESERVED NAMES - None

503.2.11 RTØDISK

PURPOSE - RTØDISK copies the relocatable tape onto separate files.

USAGE - CALL(CATM,RTØDISK)

INPUT - ATLASRT, the multi-file file from tape.

OUTPUT - All files in section 505.1 except CATM.

RESERVED NAMES - None

503.2.12 RTØFILE

PURPOSE - RTØFILE copies all of the relocatable tape files to one file, ATLASRT.

USAGE - CALL(CATM,RTØFILE)

INPUT - ATLAS relocatable tape files.

OUTPUT - ATLASRT

RESERVED NAMES - None

503.2.13 RTØLGØ

PURPOSE - RTØLGØ copies all of the relocatable files to LGØ in preparation for loading.

USAGE - CALL(CATM,RTØLGØ)

INPUT - ATLAS relocatable files

OUTPUT - LGØ

RESERVED NAMES - None

503.2.14 RTØTAPE

PURPOSE - RTØTAPE copies the relocatable tape files to tape.

USAGE - CALL(CATM,RTØTAPE)

INPUT - The ATLAS relocatable files

OUTPUT - File RT on tape and a catalog of RT on ØUTPUT

RESERVED NAMES - ATLASRT

503.2.15 UPDATE

PURPOSE - UPDATE is used to update the module ØLDPL's and relocatables.

USAGE - CALL(CATM,UPDATE(UDL=XXXXØLD, REL=XXXXREL,INPUT=intile, ØUTPUT=outfile)) where XXXX are the first four letters of the module name.

INPUT - UPDATE correction set. Default file - INPUT.

OUTPUT - a. Updated ØLDPL and relocatable files.
b. Listings on file "outfile" (default-ØUTPUT)

RESERVED NAMES - ØLDPL, CØMPILE, SNARKF, QX8, QX13, QX18, CØR, ØLD, NEW, R1

503.2.16 PRECUP

PURPOSE - PRECUP is used to update a precompiler's ØLDPL, relocatables, and absolutes

USATE - CALL(CATM,PRECUP,N(UDL=XXXØLD, REL=XXXREL,ABS=absfile, INPUT=intile,ØUTPUT=outfile))

where XXX are the 3 characters identifying the precompiler.

INPUT - Update correction set. Detault file=INPUT

OUTPUT - a. Updated ØLDPL and relocatable and absolute files
b. Listings on "outfile" - ØUTPUT

RESERVED NAMES - ØLDPL,CØMPILE,MACF, QX8,QX13,QX18,BMØD,BNEW

503.2.17 FULLUP

PURPOSE - FULLUP is used to make a full update of a module's ØLDPL and relocatables to get a complete program listing

USAGE - CALL(CATM,FULLUP,N(UDL=XXXXØLD, REL=XXXXREL,INPUT=infile, ØUTPUT=outfile)

where XXXX are the first four letters of the module name

INPUT - UPDATE correction set. Default file=INPUT

OUTPUT - a. Updated ØLDPL and relocatable files
b. Listing on file "outfile" (default-ØUTPUT)

RESERVED NAMES - ØLDPL,CØMPILE,SNARKF, QX8,QX13,QX18,CØR,ØLD,NEW,R1

503.3 CHECKOUT PROCEDURES

CHEKØUT, CHEKRUN, CØMMENT, DATAUP, DTØDISK, DTØTAPE

503.3.1 CHEKØUT

PURPOSE - CHEKØUT is used to write the ATLAS checkout decks to file CHEKØUT.

USAGE - CALL(CATM,CHEKØUT(INPUT=infile, ØUTPUT=outfile))

INPUT - File of UPDATE directives. Default-INPUT

OUTPUT - a. The checkout decks requested on file CHEKØUT
b. Listings on "outfile" (default - ØUTPUT)

RESERVED NAMES - None

503.3.2 CHEKRUN

PURPOSE - CHEKRUN is used to run the checkout decks

USAGE = CALL(CATM,CHEKRUN(G=n,INPUT=infile, ØUTPUT=outfile)) where n is the number of decks to be run.

INPUT - File of update directives. Default - INPUT

OUTPUT - Listings of n ATLAS executions on "outfile" (default-ØUTPUT)

RESERVED NAMES - All data files, F0, F1, F2, F3, F4, F5, QX8, QX13, QX18, CREL, REL, CØNTRØL, R1

503.3.3 CØMMENT

PURPOSE - CØMMENT is used to write the ATLAS checkout form in the dayfile.

USAGE - CALL(CATM,CØMMENT)

INPUT - None

RESERVED NAMES - None

503.3.4 DATAUP

PURPOSE - DATAUP is used to perform updates to the ATLAS checkout decks.

USAGE - CALL(CATM,DATAUP(INPUT=intile, ØUTPUT=outfile))

INPUT - UPDATE correction set. Default - INPUT.

OUTPUT - a. Updated ØLDPL (CHEKØLD).
b. Modified decks written to CHEKØUT.
c. Listings on "outfile". (default - ØUTPUT)

RESERVED NAMES - NEWPL

503.3.5 DTØDISK

PURPOSE - DTØDISK copies CHEKØLD from tape file IT.

USAGE - CALL(CATM,DTODISK)

INPUT - File IT on tape.

OUTPUT - CHEKØLD

RESERVED NAMES - IT

503.3.6 DTØTAPE

PURPOSE - DTØTAPE copies CATM and CHEKØLD to tape file ØT.

USAGE - CALL(CATM,DTØTAPE)

INPUT - CATM, CHEKØLD

OUTPUT - Tape file ØT and a catalog of ØT on ØUTPUT.

RESERVED NAMES - ØT

504. ATLAS CHECKOUT DECKS

ATLAS checkout decks are maintained on tape in an UPDATE ØLDPL format. The decks are maintained and run using procedures on file CATM (sec. 503.3).

The checkout runs conform to the following requirements:

a. data tape or file manipulations are not allowed;
b. each deck is executable as a standard ATLAS job;
c. matrix checksums are printed.

They are set up as follows:

```
$DECK XXXXnn (or YXXXXnn)

      BEGIN CØNTRØL MATRIX PRØGRAM XXXXNN (or YXXXXNN)
      PRØBLEM ID
C
C
C
C     comments describing deck
C
C
      CALL REQFL (field length)

      checkout control program

$WEØR
     ATLAS checkout data (if required)
$WEØR
```

Each deck name is the same as the control program name and is constructed from the module name and the deck number, e.g., MASS01. If the deck tests any of the following classifications, the prefix Y of the above example is replaced with the appropriate letter.

Preprocessor	-	I
Postprocessor	-	Ø
Plot	-	P
Library routine	-	L

CØMDECKs should start with a C, e.g., CSTIF03.

Every control program must be a SNARK program.

The PRØBLEM ID briefly describes the checkout deck.

Comments fully describing the purpose of the checkout deck follow the PRØBLEM ID statement. These comments supply the following information:

```
C
C     Module
C
C     Purpose
C
C     Author
C
C     Run History
C
C     Updates
C
C     Run Cost - CRUs, CP Time
C
C     Core
C
C     References
C
C     Method - include description of any checkout switches
C
C     Routines Checked
C
C
```

The call to REQFL requests the field length needed for execution. This call allows the economical running of the deck without exact knowledge of the core requirements.

The control program is terminated by a $WEØR card. Any checkout data present is terminated with a second $WEØR.

505. TAPE STRUCTURE

505.1 RELOCATABLE TAPE

File	Name	Contents
1	CATM	ATLAS control card procedures with random access table
2	CATN	ATLAS control card procedures without random access table
3	DUMMØLD	Dummy deck ØLDPL
4	ACIØLD	ACI ØLDPL
5	ACIREL	ACI relocatables
6	ACI	ATLAS Catalog Interpreter
7	SYSTPRØ	ACI statement file
8	APCØLD	ATLASPC ØLDPL
9	APCREL	ATLASPC relocatables
10	ATLASPC	ATLAS precompiler
11	MACØLD	MAC ØLDPL
12	MACREL	MAC relocatables
13	MAC	MAC precompiler
14	MACLIB	MAC library
15	SNKØLD	SNARK ØLDPL
16	SNKREL	SNARK relocatables
17	SNARK	SNARK precompiler
18	SNKLØLD	SNARK library ØLDPL
19	SNKLREL	SNARK library relocatables
20	SNKLIB	SNARK library
21	TRACØLD	TRACE ØLDPL
22	TRACREL	TRACE relocatables
23	TRACE	TRACE program
24	ATLASR	ØVERLAY(ATLAS,0,0) relocatables
25	ATLAS	ØVERLAY(ATLAS,0,0) absolutes
26	DUMMYR	ØVERLAY(DUMMY,0,0) relocatables
27	ALIBØLD	ATLAS library ØLDPL
28	ALIBREL	ATLAS library relocatables
29	CLIBREL	Control library relocatables
30	ILIBØLD	Interactive control library ØLDPL
31	ILIBREL	Interactive control library relocatables
32	ADDIØLD	Add/Interpolate ØLDPL
33	ADDIREL	Add/Interpolate relocatables
34	AF10ØLD	AF1 ØLDPL
35	AF10REL	AF1 relocatables
36	CHØLØLD	CHØLESKY ØLDPL
37	CHØLREL	CHØLESKY relocatables
38	DESIØLD	DESIGN ØLDPL
39	DESIREL	DESIGN relocatables
40	DUBLØLD	Doublet Lattice ØLDPL
41	DUBLREL	Doublet Lattice relocatables
42	EXTRØLD	EXTRACT ØLDPL

43	EXTRREL	EXTRACT relocatables
44	FLEXØLD	Res'dual Flexibility ØLDPL
45	FLEXREL	Residual Flexibility relocatables
46	FLUTØLD	V-G FLUTTER ØLDPL
47	FLUTREL	V-G Flutter relocatables
48	GRAPØLD	GRAPHICS ØLDPL
49	GRAPREL	GRAPHICS relocatables
50	INPTØLD	INPUT ØLDPL
51	INPTREL	INPUT relocatables
52	INTEØLD	INTERPØLATIØN ØLDPL
53	INTEREL	INTERPØLATIØN relocatables
54	LØADØLD	LØADS ØLDPL
55	LØADREL	LØADS relocatables
56	MACHØLD	MACHBØX ØLDPL
57	MACHREL	MACHBØX relocatables
58	MASSØLD	MASS ØLDPL
59	MASSREL	MASS relocatables
60	MERGØLD	MERGE ØLDPL
61	MERGREL	MERGE relocatables
62	MULTØLD	MULTIPLY ØLDPL
63	MULTREL	MULTIPLY relocatables
64	ØTPTØLD	ØUTPUT ØLDPL
65	ØTPTREL	ØUTPUT relocatables
66	PRNTØLD	PRINT ØLDPL
67	PRNTREL	PRINT relocatables
68	RHØ3ØLD	RHØ3 ØLDPL
69	RHØ3REL	RHØ3 relocatables
70	STIFØLD	STIFFNESS ØLDPL
71	STIFREL	STIFFNESS relocatables
72	STREØLD	STRESS ØLDPL
73	STREREL	STRESS relocatables
74	VIBRØLD	VIBRATIØN ØLDPL
75	VIBRREL	VIBRATIØN relocatables

505.2 ABSOLUTE TAPE

File	Name	Contents
1	CATM	ATLAS control card procedures
2	ACI	ATLAS Catalog Interpreter
3	ATLASPC	ATLASPC precompiler
4	SNARK	SNARK precompiler
5	SYSTPRØ	ACI statement file
6	ATLASL	ATLAS library
7	CLIB	Control library
8	ILIB	Interactive control library
9	VLIB	Version library
10	DUMMYR	ØVERLAY(DUMMY,0,0) relocatables
11	ATLAS	ØVERLAY(ATLAS,0,0) absolutes
12	ADDIØLF	Add/Interpolate absolutes
13	AF10ØLF	AF1 absolutes

14	BUCKØLF	BUCKLING absolutes
15	CHØLØLF	CHØLESKY absolutes
16	DESIØLF	DESIGN absolutes
17	DUBLØLF	Doublet Lattice absolutes
18	EXTRØLF	EXTRACT absolutes
19	FLEXØLF	Residual Flexibility absolutes
20	FLUTØLF	V-G Flutter absolutes
21	GRAPØLF	GRAPHICS absolutes
22	INPTØLF	INPUT absolutes
23	IN02ØLF	
24	IN02ØLF	
25	IN03ØLF	
26	IN04ØLF	
27	IN05ØLF	
28	IN06ØLF	
29	IN07ØLF	
30	INTEØLF	INTERPØLATIØN absolutes
31	LØADØLF	LØADS absolutes
32	MACHØLF	MACHBØX absolutes
33	MASSØLF	MASS absolutes
34	MERGØLF	MERGE absolutes
35	MULTØLF	MULTIPLY absolutes
36	ØTPTØLF	ØUTPUT absolutes
37	PRINØLF	PRINT absolutes
38	RHØ3ØLF	RHØ3 absolutes
39	STIFØLF	STIFFNESS absolutes
40	STREØLF	STRESS absolutes
41	VIBRØLF	VIBRATIØN absolutes

600. ATLAS PREPROCESSORS

600.1 PURPOSE

The ATLAS Preprocessors read the data for the various ATLAS Processors, and incorporate it into the appropriate matrices for subsequent use. This code also reads matrices from save files generated in previous ATLAS runs.

600.2 ACCESS

The preprocessors are all overlays on file INPTØLF, which is accessed by one of the following Control program statements: (sec. 200, ref. 1-1)

```
READ INPUT
LØAD FILES(plist)
LØAD MATRIX(plist)
```

600.3 READ INPUT COMMAND

ATLAS data is divided into sections beginning with "BEGIN name DATA" records. Upon encountering such records, control is transferred to the appropriate secondary overlay to process the ensuing records. Secondary overlays, matrix naming conventions and references to appropriate sections of reference 1-1 are given in table 600-1.

Table 600-1 Preprocessor Secondary Overlays

NAME	OVERLAY FILE	SECONDARY OVERLAY NUMBERS	LEADING MATRIX NAME LETTER	REF 1-1 SECT.
BC	IN02ØLF	1	K	106
ELEMENTKEY	IN01ØLF	2	K	
GEØMETRY	IN01ØLF	3,10B,14B	G	126
MATERIAL	IN01ØLF	4	K	140
SUBSET	IN05ØLF	5	SN,SE	156
STRESS	IN03ØLF	6	N	154
MASS	IN04ØLF	10B-17B	M	138
STIFFNESS	IN01ØLF	20B-27B	K	152
INTERACT	IN06ØLF	30B-37B	I	130
FLUTTER	IN07ØLF	40B	U	122
AF1	IN07ØLF	42B	A	104
DUBLAT	IN07ØLF	43B	D	116
MACHBØX	IN07ØLF	44B	B	136
RHØ3	IN07ØLF	45B	R	150
LØADS	IN02ØLF	50B-57B	L	134
DESIGN	IN03ØLF	60B-61B	N	112
DETAIL	IN03ØLF	62B	N	114

600.4 LØAD COMMANDS

These commands result in the calling of overlay (INPTØLF,1,77B). If the substructure options described in section 230, reference 1-1, are used, overlay (1,76B) is also called.

600.5 ADDITION OF PREPROCESSING CAPABILITY

a. Changes to ØVERLAY(INPTØLF,1,0)-PREPRØC

1. Change dimension of IDATA array.
2. Change DATA statement for IDATA array.
3. Change limit on DØ loop ending on statement 12.
4. If the new data capability involves the use of several secondary overlays, flow is controlled by variables LØVLY and NØVLY in which the numbers of the current and next overlays are stored. These are in common block /MØDUCØM/, which must also appear in the secondary overlays, and which contains a scratch area for communication between secondaries.
5. Change DATA statement for INØLF array to reflect overlay file.

700. ATLAS TECHNICAL PROCESSORS

Sections 701 through 716 discuss the code and structure of the ATLAS technical processors. The basic intent is to make it possible for programmers, unfamiliar with the code, to gain a fair understanding of the functioning of the module prior to actually reading the code. ATLAS communication files, general program method, subroutine and common usage are all presented with this intent.

701. ADDINT PROCESSOR

701.1 GENERAL INFORMATION

701.1.1 Purpose

1. Sum at corresponding reduced frequencies the sets of generalized air force matrices, [Q], for two or more lifting surfaces, and/or
2. interpolate over a set of generalized air force matrices for an array of reduced frequencies and K-values to find air force matrices at intermediate K-values.

701.1.2 Access

This module is called from the ATLAS control program by the EXECUTE ADDINT(plist) statement (sec. 202, ref. 1-1).

701.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	ACMij00	Aerodynamic module control matrix	AF10RNF DUBLRNF
	or		MACHRNF RF03RNF
	Sxxxx		FLEXRNF ADDIRNF
2	GFØijkl	Unsteady generalized air force matrices	AF10RNF DUBLRNF
	or		MACHRNF RHØ3RNF
	Sxxxxyy		ADDIRNF FLEXRNF
Output			
1	xxxxx	ADDINT control matrix	ADDIRNF
2	xxxxxyy	Unsteady generalized air force matrices (summed or interpolated)	ADDIRNF

A detailed description of the matrix formats is contained in reference 1-2.

701.3 PROGRAM METHOD

ADDINT consists of one primary overlay, named ADDINT, and four secondary overlays named DECPAR, RSPW, RCCIW, and RMW.

701.3.1 ØVERLAY (ADDIØLF,1,0) - ADDINT

ADDINT establishes labeled common blocks and directs program flow through secondary overlays. No calculations or input/output are performed in ADDINT.

701.3.2 ØVERLAY (ADDIØLF,1,1) - DECPAR

DECPAR is called to

a. Interpret the execution parameters passed to ADDINT through the ATLAS common block /CØNPARS/.

b. Read the case-condition control matrices written by the unsteady aerodynamics module(s) and extract the arrays of K-values.

c. Determine the storage allocation required by all subsequent ADDINT overlays. The array storage is problem dependent.

d. Any errors discovered during steps a - c will cause diagnostics to be printed and ADDINT execution to be terminated.

701.3.3 ØVERLAY (ADDIØLF,1,2) - RSPW

For each input K-value, RSPW reads and sums generalized air force matrices, [Q], over all surfaces, and writes the summed matrices on either -

a. ADDIRNF as a full matrix if no interpolation is desired, or

b. SCRAT1, in NPARTS vectors, if it is to be read later in RCCIW for interpolation. The size of the vectors, IPSIZE, is determined by available core in the next module, RCCIW, which will have to store two vectors for each of the NINK input [Q] matrices and one vector of the output [Q] matrix.

701.3.4 ØVERLAY (ADDIØLF,1,3) - RCCIW

RCCIW produces generalized air force matrices, [Q], for intermediate K-values by interpolating between the [Q] matrices calculated in RSPW. For each partition i of the [Q] matrices generated in RSPW repeat steps a., b., c., and d.

a. Read partition i of all NINK [Q] matrices from RSPW into the rows of the matrix [GFKIP] which is of size NINK x IPSIZE.

b. Using the subroutines CØMCUB fit a chain of cubics through each of the vectors formed by columns of [GFKIP] and the corresponding K-values. This will produce a matrix of slopes whose elements correspond to those of [GFKIP].

For each output K-value repeat steps c. and d.

c. Determine the appropriate K-value interval and interpolate for a partition of the output [Q] matrix.

d. If NPARTS = 1, write the [Q] matrix and the control matrix to ADDIRNF. If NPARTS>1, write the vectors to SCRAT1.

701.3.5 ØVERLAY (ADDIØLF,1,4) - RMW

RMW is called to read the NPARTS partitions of each interpolated [Q], merge the partitions, and write the final matrices on ADDIRNF. The control matrix is also written onto ADDIRNF.

701.4 COMMON BLOCK USAGE

	ØVERLAY: ADDIØLF				
NAME	(1,0) ADDINT	(1,1) DECPAK	(1,2) RSPW	(1,3) RCCIW	(1,4) RMW
CØNPARS		x			
KERRØR	x				
KORNDM	x	x			
ERRCØM	x	x	x	x	x
FILCØM	x	x	x	x	x
KINCØM	x	x	x	x	x
KØTCØM	x	x	x	x	x
ØPTCØM	x	x	x	x	x
SIZCØM	x	x	x	x	x
SURCØM	x	x	x	x	x

701.4.1 CØNPARS
KERRØR - ATLAS system common blocks, described
KORNDM in section 100.2.

701.4.2 ERRCØM - Contains a count of the number of fatal errors diagnosed by ADDINT.

FILCØM - Contains the names of the ADDINT scratch and output files and the names to be assigned to output matrices.

KINCØM - Contains the array of input reduced frequencies (K-values).

KØTCØM - Contains the array of output reduced frequencies.

ØPTCØM - Contains program options specifying action to be taken (add and/or interpolate) and the checkout level.

SIZCØM - Contains variables defining the problem size.

SURCØM - Contains variables pertinent to the input surfaces: generating program, Mach number, span, altitude, Bref, etc.

702. AF1 PROCESSOR

702.1 GENERAL INFORMATION

702.1.1 Purpose

The AF1 Processor generates pressure distributions, sectional forces, and generalized air forces for general three-dimensional configurations using modified strip theory (ref. 702-1). Execution options include:

1. Use of externally generated vibration mode shapes or internally generated rigid body modes
2. One plane of symmetry
3. Experimental correction data
4. Quasi-steady or unsteady analysis

702.1.2 Access

This module is called from the ATLAS control program by the EXECUTE AF1 statement (sec. 204, ref. 1-1).

702.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	AFCCi	Control Surface Correspondence	DATARNF
2	AFCFi	Experimental Data	DATARNF
3	AFCGi	Control Surface Geometry	DATARNF
4	AFCSi	Control and Size Data	DATARNF
5	AFMCi	Modal Control Data	DATARNF
6	AFMGi	Main Surface Geometry	DATARNF
7	AFSLi	Sectional Lift Data	DATARNF
8	AFPMi	Pitching Moment Data	DATARNF
9	AFRBi	Rigid Body Modes	DATARNF
10	AFTCi	Tab Surface Correspondence	DATARNF
11	AFTGi	Tab Surface Geometry	DATARNF
12	AFURi	Unit Rotations Control	DATARNF
13	AFYGi	Strip Cut Geometry	DATARNF
14	Cddd	Modal Interpolation Arrays	INTERNF
15	INTABLE	Interpolation Table	INTERNF

Output

1	ACMij	Control and Size Data	AF10RNF
2	CAyijAl	Component Forces	AF10RNF
3	CGCij	Control Surface Geometry	AF10RNF
4	CTCij	Geometry Correspondence Table	AF10RNF
5	GF0ijAl	Generalized Airforces	AF10RNF
6	M1Cij	Main Surface Geometry (Part 1)	AF10RNF
7	M2Cij	Main Surface Geometry (Part 2)	AF10RNF
8	SIØij	Static Induction Matrix	AF10RNF
9	SAyijAl	Sectional Forces	AF10RNF
10	TGCij	Tab Surface Geometry	AF10RNF
11	Wxxij	Mode Shapes	AF10RNF
12	XMØij	Lift Curve Slopes	AF10RNF

702.3 PROGRAM METHOD

The AF1 module includes the primary overlay and four secondary overlays on absolute file AF10ØLF.

Module Name	Overlay Level	Description
AF1	(1,0)	Controls flow of execution
AFGEØM	(1,1)	Generates geometry data
AFSI	(1,2)	Generates static induction matrix
AFMØDE	(1,3)	Generates mode shapes
AFGAF	(1,4)	Generates pressures, sectional forces, and generalized airforces.

702.3.1 ØVERLAY(AF10ØLF,1,0)-AF1

a. Retrieves and checks execution parameters from common block /CØNPARS/.

b. Reads control information from file DATARNF and stores this data in labeled common block /CØNTRL/ for communication with secondary overlays.

c. Each secondary overlay is preloaded to determine the length of available blank common storage. A summary of available and required blank common area is printed. The job is aborted if insufficient field length is available for the problem size.

d. Secondary overlays (1,1), (1,2), (1,3), and (1,4) are called.

e. Prints a job summary containing execution times for each secondary overlay.

702.3.2 ØVERLAY(AF10ØLF,1,1)-AFGEØM

Generates strip geometry data

a. Reads main surface, control surface, and tab surface geometry input data matrices from file DATARNF.

b. Computes strip centerline 1/2 chord, 3/4 chord and elastic axis coordinates; strip dihedrals, strip widths, and static induction geometry parameters for each strip.

c. Outputs strip geometry matrices to file AF10RNF.

702.3.3 ØVERLAY(AF10ØLF,1,2)-AFSI

Generates the static induction matrix

a. Reads static induction geometry matrix from file DATARNF.

b. Computes 2-D or 3-D static induction matrix, S, depending on the user assigned optional value of TWØD.

c. Outputs the static induction matrix to file AF10RNF.

702.3.4 ØVERLAY(AF10ØLF,1,3)-AFMØDE

Generates mode shapes.

a. Reads geometry data matrices from file AF10RNF.

b. If elastic modes have been specified, the modal control matrix is read from file DATARNF and the modal coefficient matrices are read from file INTERNF.

c. Displacements and relative rotations are computed for each strip to generate the mode shape matrix.

d. Partitions of the mode shape matrix are written on file AF10RNF.

e. If unit rotation modes have been specified, the unit rotations control matrix is read from file DATARNF.

f. Relative rotations are computed for each strip to generate the mode shape matrix.

g. Partitions of the mode shape matrix are written on file AF10RNF.

h. If rigid body modes have been specified, the rigid body modes matrix is read from file DATARNF.

i. Displacements and slopes are computed for each strip on each main surface to generate the mode shape matrix PHI.

j. Partitions of the mode shape matrix are written on file AF10RNF.

702.3.5 ØVERLAY(AF10ØLF,1,4)-AFGAF

Generates aeroelastic forces.

a Reads geometry, modal and option data from DATARNF and AF10RNF. Modifies lift and moment coefficients. Loops thru e, on Kvalue (or once) to compute unsteady forces (or quasi-steady forces).

b. Computes oscillatory derivative matrix, D.

c. Computes and outputs to AF10RNF sectional forces for each strip $(PHI)^{-1}D\ (PHI)$.

d. Computes and outputs to AF10RNF component forces for each strip, $(PHI)\ S,^{-1}D\emptyset$.

e. If quasi-steady forces are required then terminates, otherwise computes and outputs to AF10RNF generalized airforces.

702.4 COMMON BLOCK USAGE

	ØVERLAY: AF10ØLF				
NAME	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
AFILES	x	x		x	x
BCINDX		x	x	x	x
CØNPARS	x				
CØNTRL	x	x	x	x	x
ERRMSG	x	x	x	x	x
ERRSUM	x				x
KERRØR	x				
KQRNDM	x	x	x	x	x
LACØRE	x	x	x	x	x

702.4.1 /KERRØR/,/CØNPARS/,/KQRNDM/ are ATLAS system common blocks.

702.4.2 /BCINDX/ Contains blank common array indices.

/CØNTRL/ Contains program control and size information.

/ERRMSG/ Contains array of error messages.

/ERRSUM/ Contains error title and error count.

/AFILES/ Contain AF1 binary scratch file names.

/LACØRE/ Contains lengths of available core for each secondary overlay of AF1.

703. DESIGN PROCESSOR

703.1 GENERAL INFORMATION

703.1.1 Purpose

1. Produce margins of safety.
2. Update element properties.
3. Record and display history.
4. Optimize composite structure.
5. Smooth element properties.

703.1.2 Access

This module is called from the ATLAS control program by the EXECUTE DESIGN(plist) statement, (sec. 212, ref. 1-1).

703.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	NALLØWC	Compression allowables	DATARNF
2	NALLØWS	Shear allowables	DATARNF
3	NBI001a	Buckling interaction data matrix	DATARNF
4	NBUCTAB	Buckling table index	DATARNF
5	NC001ba	Design load control	DATARNF
6	NLLCRba	Design loadcases	DATARNF
7	NDP001a	Design control	DATARNF
8	ND001ba	Temperature data control	DATARNF
9	NITYPEa	Element type and partitions	DATARNF
10	NKS001a	Element control	DATARNF
11	NL001ba	Design loads	DATARNF
12	NMATERa	Material code reference	DATARNF
13	NMS001a	Margins of safety	DATARNF
14	NØCNTRa	Optimization Control matrix	DATARNF
15	NØDVCCa	Variable constraint control	DATARNF
16	NØD001a	Optimization data	DATARNF
17	NPARAMa	Parameters	DATARNF
18	NPB001a	Boundary data	DATARNF
19	NPD001a	Design data	DATARNF
20	NSMCNTa	Smoothing property control	DATARNF
21	NSMKEYa	Smoothing problem key	DATARNF
22	NSP001a	Smoothing property data	DATARNF
23	NST001a	Restrain sizing data	DATARNF
24	NVAR1Aa	Variable constraints data	DATARNF
25	LCØØRba	Loadcase correspondence	DATARNF

26	KPARMS1	Stiffness parameters	DATARNF
27	KSF001a	Element data matrix	DATARNF
28	KM00001	Material properties	DATARNF
29	SEKddda	Element subsets	DATARNF
30	SCN01ba	Stress control data	STRERNF
31	SLCSTba	Stress loadcase table	STRERNF
32	ST001ba	Stress matrix	STRERNF

	Name	Description	File	
Output				
1	DESPARa	History parameters	DESIRNF	
2	HISTRYa	History minimum margins	DESIRNF	
3	KSF001a	Element data matrix	DESIRNF	
4	MIN01ca	Resize minimum margins	DESIRNF	
5	MPARcba	Strength parameters	DESIRNF	1)
6	MPØ001a	Pointers for minimum margins	DESIRNF	
7	MTARcba	Thermal design parameter	DESIRNF	
8	M001cba	Strength minimum margins	DESIRNF	(1)
9	N001cba	Thermal design minimum margins	DESIRNF	
10	S001cba	Strength margin of safety	DESIRNF	(1)
11	SMIMcba	Strength min-max margins	DESIRNF	
12	TMIMcba	Thermal min-max margins	DESIRNF	
13	T001cba	Thermal margin of safety	DESIRNF	

(1) Also on SC00RNF

A detailed description of the matrix formats is contained in reference 1-2.

703.3 PROGRAM METHOD

The DESIGN module consists of a primary overlay DESIGN and ten secondary overlays, KØNTRØL, STRNGTH,HISTØRY,RESIZE,FLUTTER, DATAMØV,ØPTIMUM,SMØØTHG,THERMLI and THERMLX.

703.3.1 ØVERLAY(DESIGN,1,0) - DESIGN

a) Controls execution of secondary overlays by using array in common block /LØCCØNT/. This array is established by the KØNTRØL overlay, and contains an entry for each overlay that is to be executed. If multiple stage executions are requested, the overlay finds the additional information in /LØCPARS/.

b) Common blocks /CØMMASK/ and /NCØRES/ are loaded by data statements in this overlay.

703.3.2 ØVERLAY(DESIØLF,1,1) - KØNTRØL

This overlay performs the managing function of the DESIGN module and does the following:

a) Sets defaults.

b) Reads execution parameters in /CØNPARS/ and translates them to module directives, which are stored in common blocks /DALECØM/, /LØCCØNT/, /LØCFLTR/, /LØCHIST/, /LØCPAR/ and /RECØNTR/. Produces diagnostics for unrecognized parameters.

c) Sets variables, file names and overlay names.

703.3.3 ØVERLAY(DESIØLF,1,2) - STRNGTH

This overlay produces strength margins of safety.

a) It reads material allowables (produced by the material preprocessor), buckling allowables (produced by the design preprocessor), element properties (produced by the stiffness preprocessor), design data, loads data and temperature data (all produced by the design preprocessor) and finally stresses (produced by the stress processor). Based on these data it produces output margins of safety for strength.

b) The overlay also contains buckling criteria for plate-like elements which can be applied either in one step or by iterating to convergence as indicated by execution parameters.

c) The output from the overlay is a set of margins of safety and minimum margins of safety which can be either "pure strength" margins or a combination of strength and buckling.

703.3.4 ØVERLAY(DESIØLF,1,3) - HISTØRY

a) Writes all margins of safety on DESIRNF at time of execution if requested by user.

b) Produces matrix containing history of margins of safety from previous cycles for output print.

703.3.5 ØVERLAY(DESIØLF,1,4) - RESIZE

This overlay updates the element properties in the KSF-matrices on DATARNF.

a) It reads bound data, design data, restraining data, input margin data, element property data (all on DATARNF) and calculated margins of safety (on DESIRNF).

b) Updates element properties in the KSF matrices with regard to calculated margins of safety subject to the user constraints which can be input, calculated or introduced as execution parameters.

703.3.6 ØVERLAY(DESIØLF,1,5) - FLUTTER

a) This overlay receives input data through the execution parameters (subsets and associated factors). These are converted to margins of safety for updating of element properties. This serves two purposes: to update structure for simple factors and to produce "flutter constraints" (equivalent to calculated lower bounds on element proper[illegible].

703.3.7 ØVERLAY(DESIØLF,1,6) - DATAMØV

The function of this overlay is purely administrative. It saves "old" element properties on DESIRNF in order to guarantee data integrity and to assure an alternate restart point for a new resizing.

703.3.8 ØVERLAY(DESIØLF,1,11) - ØPTIMUM

This overlay performs optimizations of composite elements (CPLATE and CCØVER). It is built around the concept of local optimization and operates on a set of problems defined in the input as element subsets. The constraints are of strength type and the design variables can be coupled by directives in the input stream.

703.3.9 ØVERLAY(DESIØLF,1,12) - SMØØTHG

This overlay changes (smooths) element properties according to the specifications provided by the user in the smoothing data.

703.3.10 ØVERLAY(DESIØLF,1,13) - THERMLI

This overlay initializes data for thermal design margins of safety.

It reads element properties (produced by the stiffness preprocessor), design data and thermal loads data

(produced by the design preprocessor), and stresses (produced by the stress processor). These data are collected and stored in a single block of data for each element.

The output from the overlay is a partitioned scratch matrix containing all element data (excluding material allowables) required for thermal design margins of safety.

703.3.11 ØVERLAY(DESIØLF,1,14) - THERMLX

This overlay produces thermal design margins of safety.

It reads material allowables (produced by the material preprocessor) and element data (produced by the design preprocessor). Based on these data it produces output margins of safety for thermal design.

The output from the overlay is a set of margins of safety and minimum margins of safety for thermal design.

703.4 COMMON BLOCK USAGE

NAME	OVERLAY										
	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,11	1,12	1,13	1,14
CONPARS	X	X		X							
KERROR	X	X	X	X	X	X		X	X	X	X
KQBUFP	X	X		X		X					
KQRNDM	X	X	X	X	X	X		X	X	X	X
CNMNI								X			
COMMASK	X	X	X		X	X		X			
COMSCR			X		X	X					
DALECOM	X	X	X					X			
DATAMS			X								
DESSCRT	X	X	X					X			
INIT								X			
LOCCONT	X	X	X	X	X	X	X	X			
LOCFLTR	X	X				X					
LOCHIST	X	X		X							
LOCPARS	X	X	X	X	X	X	X	X	X	X	X
MACOM								X			
NCORES	X	X	X	X	X	X		X		X	X
OLFLAG			X								
OPTCON								X			
OPTCORR								X			
OPTDDD								X			
OPTPOS								X			
OPTPROB								X			
OPTRES								X			
OPTSOL	X	X						X			
OPTVARI								X			
RECONTR	X	X			X						
REVECT					X						
THERCMI										X	
THERCMX											X
UFROP									X		

703.4.1 System Common Blocks

/CØNPARS/:
/KERRØR/:
/KQBUFP/:
/KQRNDM/:

ATLAS system common blocks detailed in sec. 100.2.

703.4.2 Design Common Blocks

/CNMN1/: Contains information for the optimizer in the composite optimization.

/CØMMASK/: Contains bit information for masking

/CØMSCR/: Scratch common block

/DALECØM/: Contains information regarding iteration for solution of nonlinear margins of safety in the STRNGTH overlay.

/DATAMS/: Contains information required to produce margins of safety for each element in the STRNGTH overlay.

/DESSCRT/: Scratch common block.

/INIT/: Information concerning the initiation of an optimization for the composite elements.

/LØCCØNT/: Contains vector defining execution file names and overlay names.

/LØCFLTR/: Contains information pertaining to the flutter execution of DESIGN.

/LØCHIST/: Contains information pertaining to the history execution of DESIGN.

/LØCPARS/: Contains execution variables for DESIGN.

/MACØM/: Information for execution of the subroutine MAMULT

/NCØRES/: Contains element description for all element types and size of output partitions.

/ØLFLAG/: Contains pointers and flags for STRNGTH overlay.

/ØPTCØN/:	Contains pointers and flags for execution of the composite optimization.
/ØPTCØRR/:	Contains scratch pointers for the composite optimization.
/ØPTDDD/:	Contains scratch matrices for conversion of strains to stresses and transformation to element frames in the composite optimization.
/ØPTPØS/:	Position numbers for active positions in the composite optimization.
/ØPTPRØB/:	Flags and pointers for output from the composite optimization.
/ØPTRES/:	Information describing critical laminae and related information.
/ØPTSØL/:	Criterion and convergence flags for composite optimization.
/ØPTVARI/:	Constraint information for the composite optimization.
/RECØNTR/:	Contains flags and pointers for the RESIZE overlay.
/REVECT/:	Contains information required for one element in order to update element properties in the RESIZE overlay.
/THERCMI/:	Contains flags and pointers for the THERMLI overlay.
/THERCMX/:	Contains flags and pointers for the THERMLX overlay.
/UPRØP/:	Contains flags and pointers for the SMØØTHG overlay.

704. DUBLAT PROCESSOR

704.1 GENERAL INFORMATION

704.1.1 Purpose

The DUBLAT Processor generates pressure distributions, stability derivatives, and generalized air forces for general three-dimensional configurations using the doublet lattice method (ref. 704-1). Execution options include:

(1) Use of externally generated vibration mode shapes or internally generated rigid body modes.

(2) One or two planes of symmetry.

(3) Pressure and normal wash corrections.

(4) Reuse of a previously generated kernel matrix at selected Mach numbers and reduced frequency values.

704.1.2 Access

This module is called from the ATLAS control program by the EXECUTE DUBLAT(plist) statement (sec. 216, ref. 1-1).

704.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	DLCSi00	Control and size data	DATARNF
2	DLPGi00	Lifting surface geometry	DATARNF
3	DLBGi00	Body interference surface geometry	DATARNF
4	DLDIi00	Body doublet geometry	DATARNF
5	DLVIi00	Velocity profiles	DARARNF
6	DLPIi00	Pressure scaling data	DATARNF
7	DLMCi00	Modal control data	DATARNF
8	DLRBi00	Rigid body modes	DATARNF
9	DLSSi00	Subset data	DATARNF
10	Cddd	Modal interpolation arrays	INTERNF
11	INTABLE	Interpolation table	INTERNF

Output

1	B1Cij00	Box geometry (part 1)	DUBLRNF
2	B2Cij00	Box geometry (part 2)	DUBLRNF
3	ACMij00	Control and size data	DUBLRNF
4	DBCij00	Body doublet geometry	DUBLRNF
5	GF0ijkl	Generalized air forces	DUBLRNF
6	M1Øij00	1/4 chord displacements	DUBLRNF
7	M3Øij00	3/4 chord slopes and displ.	DUBLRNF
8	PDØijkl	Pressure differences	DUBLRNF
9	PSCij00	Pressure scaling data	DUBLRNF
10	Q00xxkl	Quasi-inverse matrix	DUBLRNF
11	SBCij00	Strip/box corresp. table	DUBLRNF
12	SDØijkl	Stability derivatives	DUBLRNF
13	SFBijkl	Body sectional forces	DUBLRNF
14	SFØijkl	Sectional forces	DUBLRNF
15	SGCij00	Strip geometry	DUBLRNF
16	VPCij00	Velocity profile data	DUBLRNF

A detailed description of the matrix formats is contained in reference 1-2.

704.3 PROGRAM METHOD

The DUBLAT module includes the primary overlay and seven secondary overlays on absolute file DUBLØLF.

Program Name	Overlay Level	Description
DUBLAT	(1,0)	Controls flow of execution
INPUTG	(1,1)	Generates geometry and option data
MØDEB	(1,2)	Generates 1/4 chord box displacements
MØDEW	(1,3)	Generates 3/4 chord box displacements and slopes
GENDW	(1,4)	Generates kernel and wash matrices
QUASII	(1,5)	Generates quasi-inverse of kernel matrix and solution vectors
FUTSØL	(1,6)	Generates solution vectors from a previously generated quasi-inverse
MØDFIN	(1,7)	Generates pressures, sectional forces, stability derivatives, and generalized forces

704.3.1 ØVERLAY (DUBLØLF,1,0) - DUBLAT

a. Retrieves and checks execution parameters from common block /CØNPARS/.

b. Reads control information from file DATARNF and stores this data in labeled common block /CØNTRØL/ for communication with secondary overlays.

c. Each secondary overlay is preloaded to determine the length of available blank common storage. A summary of available and required blank common area is printed. The job is aborted if insufficient field length is available for the problem size.

d. Secondaries (1,1), (1,2) and (1,3) are called only once for each execution of the DUBLAT (1,0) overlay, while the (1,4) - (1,7) secondary overlays are called for each combination of reduced frequency and Mach number to be executed.

e. Prints a job summary containing execution times for each secondary overlay.

704.3.2 ØVERLAY (DUBLØLF,1,1) - INPUTG

a. Reads lifting surface and interference surface geometry input data matrices from DATARNF.

b. Computes box coordinates of sending (centerline 1/4 chord) and receiving (centerline 3/4 chord) points, box dihedral, and box inboard and outboard coordinates for each box.

c. Outputs box geometry matrices to file DUBLRNF.

d. Reads doublet geometry input data matrix from file DATARNF.

e. Computes doublet element radii and centerline coordinates for each body doublet.

f. Outputs body doublet geometry matrix to file DUBLRNF.

g. Reads pressure correction input data matrix from file DATARNF.

h. Determines the operation (pressure scaling or pressure replacement) to be made on each box.

i. Outputs pressure correction data matrix to file DUBLRNF.

j. Reads velocity profile input data matrix from file DATARNF.

k. Computes the velocity ratio, v (local)/v (free stream), to be applied to each box.

l. Outputs the velocity profile matrix to file DUBLRNF.

704.3.3 ØVERLAY (DUBLØLF,1,2) - MØDEB

a. Reads modal control matrix from file DATARNF (if elastic modes are to be used) or rigid body matrix from file DATARNF (if rigid body modes are to be used).

b. If modes are to be applied to lifting surfaces, then

(1) the box geometry matrix is read from file DUBLRNF.

(2) if elastic modes are to be used, the box subset matrix is read from file DATARNF and each modal coefficient array matrix is read from file INTERNF.

(3) box quarter chord displacements are computed using subroutine AINTG for each mode to be applied to each box.

c. If elastic modes are to be applied to body interference panels then

(1) the strip/box correspondence table is read from file DUBLRNF.

(2) each modal coefficient array matrix is read from file INTERNF.

(3) box quarter chord displacements are computed using subroutine AINTG for each mode to be applied to each box.

d. If modes are to be applied to body doublets then

(1) the doublet geometry matrix is read from file DUBLRNF.

(2) if elastic modes are to be used, each modal coefficient matrix is read from file INTERNF. Otherwise, the rigid body modes matrix is read from file DATARNF.

(3) doublet centerline displacements are computed using subroutine AINTG.

e. The quarter chord displacements matrix is output to file DUBLRNF.

704.3.4 ØVERLAY (DUBLØLF,1,3) - MØDEW

a. Reads modal control matrix from file DATARNF (if elastic modes are to be used) or rigid body matrix from file DATARNF (if rigid body modes are to be used).

b. If modes are to be applied to lifting surfaces then

(1) the box geometry matrix is read from file DUBLRNF and the subset data matrix is read from file DATARNF.

(2) if velocity profiles are to be applied to strips the velocity profile matrix is read from file DUBLRNF.

(3) if elastic modes are to be used, each modal coefficient array matrix is read from file INTERNF.

(4) box three-quarter chord displacements and slopes are computed using subroutine AINTG for each mode to be applied to each box.

(5) wash values (modified by the velocity profile effect) are computed for each box.

c. If elastic modes are to be applied to body interference panels then

(1) the strip/box correspondence table is read from file DUBLRNF.

(2) each modal coefficient array matrix is read from file INTERNF.

(3) box three-quarter chord displacements and slopes are computed using subroutine AINTG for each mode to be applied to each box.

(4) wash values are computed for each box.

d. The wash matrix is output to file DUBLRNF.

e. If body doublets were defined, then

(1) the body doublet geometry matrix is read from file DUBLRNF.

(2) if elastic modes are to be used, each modal coefficient matrix is read from file INTERNF (otherwise the rigid body modes matrix is read from file DATARNF).

(3) doublet axis displacements and slopes are computed using subroutine AINTG.

(4) the derivative of the doublet axis slopes is computed.

(5) using displacements, slopes, and derivatives of slopes, the incremental pressures due to doublets are computed and output to scratch files.

704.3.5 ØVERLAY (DUBLØLF,1,4) - GENDW

Generates the kernel matrix, [D], and the wash matrix [W].

a. If an existing quasi-inverse matrix is to be used, skips to step e.

b. The box geometry matrices and strip/box correspondence matrix are read from file DUBLRNF.

c. Loops on boxes to compute elements of the kernel matrix, [D], using subroutines INCRØ, KERNEL, and SNPDF.

d. Outputs kernel matrix to file DUBLRNF.

e. If modal data has been defined, the box geometry matrices and strip/box correspondence matrix are read from file DUBLRNF in preparation for generating the wash matrix.

f. Loops on boxes to generate a single K value wash vector for each mode. Each wash vector is output to scratch file SC4FIL.

704.3.6 ØVERLAY (DUBLØLF,1,5) - QUASII

a. Reads rows of the kernel matrix [D] into blank common until either all rows have been read or the available blank common has been filled.

b. Performs Gauss elimination on the partioned [D] matrix.

c. Outputs the upper and lower parts of the eliminated partition to two different scratch files (the lower part contains the elimination multipliers).

d. If modal data is present, the wash vectors are read and the elimination multipliers are applied to each vector.

e. Back substitution is performed to solve the system of complex equations $D^{-1}W$. The solution vector is output to a scratch file.

f. The quasi-inverse matrix is output to file DUBLRNF in row partition form.

704.3.7 ØVERLAY (DUBLØLF,1,6) - FUTSØL

Solves the system of complex equations

[D][DELCP=[W]

when a quasi-inverse matrix is available from a prior execution of QUASII. The capability to save the quasi-inverse allows the GENDW overlay to be bypassed, saving a substantial portion of the job cost. This feature applies only when the wash matrix is modified to generate a new solution.

a. Reads the quasi-inverse matrix into blank common in partitioned form.

b. Reads wash vectors into blank c on.

c. Applies elimination multipliers to the wash vectors.

d. Back substitution is performed to solve the system of complex equations $[D]^{-1}[W]$. The solution vectors are output to a scratch file.

704.3.8 ØVERLAY (DUBLØLF,1,7) - MØDFIN

Generates final pressures, sectional forces, stability derivatives and generalized forces.

a. Solution vectors (pressure coefficients) are read from scratch file and modified by the body doublet effect if doublets were defined.

b. If pressure correction data are defined, pressures are modified accordingly.

c. Loops on strips to compute sectional forces and stability derivatives.

d. Outputs sectional forces to file DUBLRNF.

e. If body doublets exist, computes body sectional forces and modifies stability derivatives.

f. Computes generalized forces.

g. Outputs pressures, stability derivatives, and generalized forces to file DUBLRNF.

704.4 COMMON BLOCK USAGE

	ØVERLAY: DUBLØLF							
NAME	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
KERRØR	x	x	x	x	x	x	x	x
CØNPARS	x							
KØRNDM	x	x	x	x	x	x	x	x
CØNTRØL	x	x	x	x	x	x	x	x
ERRMSG	x	x	x	x	x			x
ERRSUM	x							
FILES	x	x	x	x	x	x	x	x
LACØRE	x	x	x	x	x	x	x	x
MATNAM	x				x			x
SMIDØ	x							
BCINDX		x	x	x	x			x
DLM				x				

704.4.1 /KERRØR/,/CØNPARS/,/KØRNDM/ are ATLAS system common blocks, described in section 100.2.

704.4.2 /CØNTRØL/ Contains program control and size information.

/ERRMSG/ Contains array of error messages.

/ERRSUM/ Contains error title and error count.

/FILES/ Contains DUBLAT binary scratch file names.

/LACØRE/ Contains lengths of available core for each secondary overlay of DUBLAT.

/MATNAM/ Contains the portion of the matrix name which is common to all matrices.

/SMIDØ/ SNARK matrix output auxiliary ID.

/BCINDX/ Contains blank common array indices.

/DLM/ Contains kernel matrix parameters.

705. FLEXAIR PROCESSOR

705.1 GENERAL INFORMATION

705.1.1 Purpose

To calculate generalized airforce matrices that include the effects of the flexibility of vibration modes that are otherwise omitted from the mathematical model of a structure.

705.1.2 Access

This module is called from the ATLAS control program by the EXECUTE FLEXAIR statement (sec. 220, ref. 1-1)

705.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	xxxxxxx	Stiffness or Flexibility Matrix	MULTRNF
2	xxxxxxx	Mass Matrix	MULTRNF MERGRNF MASSRNF
3	MØDESvs	Mode Shapes	VIBRRNF
4	GSTIFvs	Generalized Stiffness	VIBRRNF
5	Mxxx0AB	Partitioned (Subset) Mode Shapes	VIBRRNF
6	AMC or xxxxx	Aerodynamic module Control Matrix	AF10RNF DUBLRNF MACHRNF RHØ3RNF ADDIRNF
7	GF0ghij or xxxxij	AIC Matrix (Unsteady generalized airforce matrices)	AF10RNF DUBLRNF MACHRNF RHØ3RNF ADDIRNF
Output			
1	xxxx	FLEXAIR control matrix	FLEXRNF
2	xxxxajj	Generalized force matrices which include the residual flexibility effects	FLEXRNF

705.3 PROGRAM METHOD

FLEXAIR consists of one primary overlay, named FLEXAIR, and six secondary overlays named EXPAR, DATPREP, RESFLXN, RESFLXS, GENCBAR, and GENFØR.

705.3.1 ØVERLAY (FLEXØLF,1,0) - FLEXAIR

FLEXAIR is the main program (primary overlay) of this ATLAS Utility Module. FLEXAIR establishes labeled common blocks and directs program flow through secondary overlays.

705.3.2 ØVERLAY (FLEXØLF,1,1) - EXPAR

EXPAR is called to establish the values of the /EXPARS/ common block which contains the controlling variables for the execution of the FLEXAIR utility module by:

a. Setting variables to default values.

b. Deciphering and editing the execution parameters passed to FLEXAIR through the ATLAS labeled common block /CØNPARS/. Any errors detected will cause diagnostics to be printed as they are discovered and FLEXAIR execution to be terminated.

c. Modifying values of variables in /EXPARS/ complying with the execution parameters picked up.

705.3.3 ØVERLAY (FLEXØLF,1,2) - DATPREP

DATPREP is called to establish the working storage area and to read, prepare, and compute intermediate matrices needed for computing the residual flexibility and generalized force matrices. DATPREP goes through the following steps:

a. Establish working storage area.

b. Read the control matrix from the named random file written by the unsteady aerodynamic module for the K-values, BREF, SPAN, and Mach Number.

c. Loop on number of K-values to read AIC matrices [a] from named random file written by the unsteady aerodynamic modules, to check available working storage area, and write AIC matrix (full or partitioned depending on working storage available) on SC00RNF.

a. Write K-value/Name of AIC matrix, [KVAL], on SC00RNF.

e. Read mode shape matrix [Ø] from VIBRRNF, extract user node and freedom numbers forming matrix [NØDH], reposition mode shapes forming matrix [H], and write matrices [NØDH] and [H] on SC00RNF.

f. Read generalized stiffness [KSTIFF] from VIBRRNF, check for rigid body modes, and write matrix [GSTIFF] on SC00RNF. The next 5 steps are performed for the singular stiffness case only.

g. Reorder and partition mode shapes [H] into

$$\left[\begin{array}{c|c} H_{11} & H_{12} \\ \hline H_{21} & H_{22} \end{array}\right]$$

such that $[H_{21}]$ is non-singular.

h. Write partition $[H_{12}]$ on SC00RNF.

i. Compute $[L] = [H_{11}][H_{21}]^{-1}$ and write matrix [L] on SC00RNF

j. Read mass matrix [M] from MERGRNF, MULTRNF, or MASSRNF and partition into

$$\left[\begin{array}{c|c} M_{11} & M_{12} \\ \hline M_{21} & M_{22} \end{array}\right]$$

reflecting the partitioning of mode shapes [H].

k. Compute $[U] = ([L]^T[M_{12}]+[M_{22}])^{-1}\ ([L]^T[M_{11}]+[M_{21}])$ and write [U] on SC00RNF.
Continue non-singular and singular stiffness case from here.

l. Read stiffness matrix [K] from MERGRNF or MULTRNF and write stiffness [K] on SC00RNF. Perform the next step for the singular stiffness case only.

m. Establish partitions $\left[\begin{array}{c|c} K_{11} & K_{12} \\ \hline K_{21} & K_{22} \end{array}\right]$ reflecting the partitioning of mode shapes and write partition $[K_{11}]$ on SC00RNF. Perform the next step only when the subset indicator is on, otherwise equate index name of [H] to [HB].

n. Read partitioned (subset) mode shape matrix [ØS] from VIBRRNF; extract user node and freedom numbers forming matrix [NØDS], reposition mode shapes forming matrix [HB], and write matrices [NØDS] and [HB] on SC00RNF.

705.3.4 ØVERLAY (FLEXØLF,1,3) - RESFLXN

RESFLXN, called to compute the residual flexibility matrix $[C_2]$ for the non-singular stiffness case, goes through the following steps:

a. Read stiffness matrix [K] from SC00RNF and invert to $[K]^{-1}$.

b. Read generalized stiffness matrix [KSTIFF] and mode shapes [H] from SC00RNF.

c. Compute residual flexibility matrix $[C_2]$ by routine GENHKH:

$$[C_2] = [K]^{-1} - [H][GSTIFF]^{-1}[H]^T$$

d. Write residual flexibility matrix $[C_2]$ on SC00RNF.

705.3.5 ØVERLAY (FLEXØLF,1,4) - RESFLXS

RESFLXS is called to compute the residual flexibility matrix $[C_2]$ for the singular stiffness case. RESFLXS goes through the following steps:

a. Read matrix [L], matrix [U], and matrix $[K_{11}]$ from SC00RNF.

b. Compute intermediate result matrix [KT] by routine CALCK:

$$[KT] = ([I]+[L][U])^T[K_{11}]([I]+[L][U])$$

c. Invert [KT] to $[KT]^{-1}$.

d. Read generalized stiffness [GSTIFF] and matrix $[H_{12}]$ from SC00RNF.

e. Compute intermediate result matrix $[CT_2]$ by routine GENHKH:

$$[CT_2] = [GSTIFF]^{-1} - [H_{12}][GSTIFF]^{-1}[H_{12}]^T$$

f. Read matrix [U] from SC00RNF.

g. Compute residual flexibility matrix $[C_2]$ by routine CALCC2:

$$[C_2] = \left[\begin{matrix} I \\ -U \end{matrix}\right][CT_2]\left[I \mid -U^T\right]$$

h. Write residual flexibility matrix $[C_2]$ on SC00RNF.

705.3.6 ØVERLAY (FLEXØLF,1,5) - GENCBAR

GENCBAR is called to form the residual flexibility matrix $[RF_2]$ associated with the given partitioned (subset) mode shape node and freedom numbers. GENCBAR goes through the following steps:

a. Read user node and freedom number matrices [NØDH] and [NØDS] for mode shapes and partitioned mode shapes, respectively, from SC00RNF.

b. Form an array of row numbers of the residual flexibility matrix relating to the partitioned (subset) mode shapes by routine RØWNUM.

c. Read residual flexibility matrix $[C_2]$ from SC00RNF.

d. Form the residual flexibility matrix $[RF_2]$ relating to the partitioned mode shapes by routine FRMCBAR.

e. Write partitioned residual flexibility matrix $[RF_2]$ on SC00RNF over writing $[C_2]$ and using $[C_2]$ index name.

705.3.7 ØVERLAY (FLEXØLF,1,6) - GENFØR

GENFØR is called to compute the generalized airforces with residual flexibility effects. GENFØR goes through the following steps:

a. Establish sequential scratch files FLEXSC1, FLEXSC2, FLEXSC3, and FLEXSC4 for partitiond matrices.

b. Calculate density of air, ρ, for all given altitudes by routine AT62.

c. Read matrix [KVAL] for K-values and name of AIC matrices.

d. Setup the FLEXAIR control matrix [CØNTRØL] and establish its values. Then write FLEXAIR control matrix [CØNTRØL] on FLEXRNF.
Loop on the number of K-values, NKVALUE.

e. Read mode shapes [HB] and AIC matrix [a] (full or partitioned) from SC00RNF.

f. Compute intermediate result:

$$[V] = [a][HB]$$

g. Write matrix [V] on FLEXSC2

h. Read residual flexibility matrix [C_2] or [RF_2] and AIC matrix [a] from SC00RNF.

i. Compute intermediate result:

$$[W] = [a][C_2]$$

j. Write matrix [W] on FLEXSC1
Loop on number of Altitudes, NALT

k. Compute constant:

$$Y = \frac{\frac{1}{2}\rho b^2 (2\pi f)}{K_{rf}^2}$$

l. Compute [E] then decompose to upper and lower triangles:

$$[E] = ([I] + Y[W])^{-1}$$

$$= [L/U] \text{ decomposed by the routine CDECØM.}$$

m. Read matrix [V] from FLEXSC2

n. Compute intermediate result:

$$[D] = [E][V]$$

o. Read mode shapes [HB] from SC00RNF

p. Calculate generalized force matrix:

$$[D_2] = -[HB]^T[D]$$

q. Write generalized force matrix $[D_2]$ on FLEXRNF.
END of Altitudes loop.
End of K-values loop.

r. Drop and delete scratch files FLEXSC1, FLEXSC2, FLEXSC3, and FLEXSC4.

705.4 COMMON BLOCK USAGE

Labeled common blocks are used to pass options, constants, counters, and small arrays between secondary overlays of FLEXAIR. The block names and the overlays in which they are defined are shown below.

NAME	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)
CØMPARS	x	x	x				
EXPARS	x	x	x	x	x	x	x
KERRØR	x	x	x	x	x	x	x
KQBUFP	x						
KQRNDM	x	x	x	x	x	x	x
PARTI	x	x	x				x

705.4.1 CØNPARS, KERRØR, KQBUFP, and KQRNDM are ATLAS system common blocks described in section 100.2

705.4.2 /EXPARS/ contains the controlling parameters for the execution of the FLEXAIR utility module. The parameters are first set to default values, and then modified to comply with the parameters in the EXECUTE FLEXAIR statement.

/PARTI/ contains the number of words available in working storage, the number of partitions, and the size of the partitions of the various matrices that need to be partitioned.

706. FLUTTER PROCESSOR

706.1 GENERAL INFORMATION

706.1.1 Purpose

This module computes the eigensolution of a flutter matrix from stiffness, structural damping, air forces, and mass or inertia input matrices. These matrices may be modified by the user "changeset" instructions. The solution is based on the parametric method of reference 706-1. By solving the root-sorting problem, interpolation for flutter crossings and automatic plotting can be accomplished. Flutter in fluids other than air may also be simulated. The velocity-damping, V-g, and velocity-frequency, V-f, curves can be optionally plotted for each altitude investigated. When requested, the normal and adjoint eigenvectors are found at flutter and at specified reduced frequencies.

706.1.2 Access

This technical module is called from the ATLAS Control program by the EXECUTE FLUTTER(plist) statement. (sec. 222, ref. 1-1).

706.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	ULCSi	Input data for cases	DATARNF
2	xxxxx	Air force control	ADDIRNF
3	xxxxxyy	Unsteady air forces	ADDIRNF
4	GMASSxx	Generalized mass matrix	VIBRRNF
5	GSTIFxx	Generalized stiffness matrix	VIBRRNF
	xxxxxxx	Generalized mass or stiffness	
Output			
1	Fkiupvj	Print output matrix	FLUTRNF
2	FPiupvjx	V-g plot matrix	FLUTRNF
3	FLBCij	Flutter Solution Control	FLUTRNF
4	Fiupvjw	Eigensolution matrices	FLUTRNF
5	FPiupvj	Plot control matrix	FLUTRNF

A detailed description of the matrix formats is contained in reference 1-2.

706.3 PROGRAM METHOD

The FLUTTER processor consists of a primary overlay, FLUDETN, and two secondary overlays, FLUTRD and FLUSØL.

706.3.1 ØVERLAY(FLUTØLF,1,0)-FLUDETN

This program opens the required named random files: ADDIRNF, VIBRRNF, DATARNF and FLUTRNF. It calls FLUTRD to interpret the EXECUTE FLUTTER statement. For each flutter data case identified, it calls FLUSØL.

706.3.2 ØVERLAY(FLUTØLF,1,1)-FLUTRD

a. Reads and interprets the input data specified in the EXECUTE FLUTTER statement.

b. Checks data validity.

c. Reads the control matrix of the generalized air force matrix on file ADDIRNF and extracts data constants: reference length, Mach number, number of degrees of freedom, altitude for residual flexibility option and the number of reduced frequencies for which air force matrices exist.

d. Checks the utility matrix table for the presence of the generalized mass and stiffness matrices on VIBRRNF.

e. Confirms the presence of the general flutter data matrix on DATARNF for each case identifier, and the presence of the last air force matrix on ADDIRNF.

f. Prints the number of fatal errors found and appropriate diagnostics.

706.3.3 ØVERLAY(FLUTØLF,1,2)-FLUSØL

This program performs the flutter eigensolution for each flutter case defined.

a. Reads the general flutter data matrix created by the input preprocessor. It then extracts the nominal case data and the pointers to the changeset data.

For each changeset

b. Reads the generalized mass and stiffness matrices from VIBRRNF.

c. Performs element changes as specified by the changeset instructions, on the generalized mass, stiffness and structural damping matrices.

For each retention vector set

d. Forms the complex generalized stiffness matrix according to the retention vector set. Finds the number of rigid body modes and rearranges the retention vector set.

e. Forms the inverse of the complex stiffness matrix.

For each altitude

f. Computes the air density and finds the still air coupled modes when requested.

g. For each reduced frequency, reads the airforce matrix from ADDIRNF and forms the flutter matrix according to the current retention vector set. Solves for eigenvalues of the flutter matrix using the method of Laguerre iteration.

h. Obtains flutter roots and crossings by linear interpolation on the V-g curve. When requested, computes the normal and adjoint eigenvectors at flutter points and/or at specified reduced frequencies.

i. Writes matrices FRiupvj, Fiupvjw and FPiupvj on FLUTRNF, for printing, eigensolution data and plotting, respectively.

j. If a matched point solution is requested, the next altitude is computed from the lowest flutter speed found.

706.4 COMMON BLOCK USAGE

706.4.1

/KERRØR/
/CØNPARS/
/KQRNDM/

ATLAS System common blocks, described in section 100.?

706.4.2

/XQCØM1/ Input data for title and headings

/XQCØM2/ Data from EXECUTE FLUTTER statement

/XQCØM3/ Flutter roots data

/XQCØM4/ Flutter Matrix Pointers and Options

/DACØM1/ Eigensolution data constants

/EIGCØM/ Laguerre iteration eigensolution data

	ØVERLAY: FLUTØLF		
NAME	(1,0)	(1,1)	(1,2)
CØNPARS		x	
KERRØR	x	x	x
KQRNDM	x		
XQCØM1		x	x
XQCØM2		x	x
XQCØM3		x	x
XQCØM4			x
DACØM1		x	x
EIGCØM			x

707. INTERPOLATION PROCESSOR

707.1 GENERAL INFORMATION

707.1.1 Purpose

Generates an array of interpolating function coefficients for one of four types of representations:

a. Surface spline - where the interpolating function is the equation of an infinite pinned plate.

b. Motion axis - where the interpolating functions are cubic splines along a continuous planar curve.

c. Polynomial - where the interpolating function is represented by an nth degree polynomial.

d. Motion point - where the interpolating function is a motion transformation from a single point.

e. Beam spline - where the interpolating functions are cubic splines in arc length along a continuous planar curve called a beam. A set of these beams form a planar or nearly-planar surface.

707.1.2 Access

This module is called from the ATLAS control program by the EXECUTE INTERPØLATIØN(plist) statement (sec. 2.2, ref. 1-1).

707.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	SMdddvs	Subset mode matrix	VIBRRNF
2	KNØALTa	Nodal data matrix	DATARNF
3	KNC100a	Nodal correspondence table	DATARNF
4	KLØCØØa	Local coordinate system matrix	DATARNF
5	SNKddda	Nodal subset matrix	DATARNF
6	SFdddvs	Subset freedoms and node numbers	VIBRRNF

Output

1	CdddØ	Array of coefficients	INTERNF
2	INTABLE		INTERNF

A detailed description of the matrix formats is contained in reference 1-2.

707.3 PROGRAM METHOD

The INTERPØLATIØN module consists of a primary overlay, INTEPRG, and seven secondary overlays, DIEPAR, DATPREP, DATPRE2, SURFSPL, MØTIØNA, PØLY, and MØTIØNP.

707.3.1 ØVERLAY(INTEØLF,1,0) - INTEPRG

Controls the flow of execution.

a. Establishes module common blocks and sets parameters to default values.

b. Controls the logical flow of the secondary overlay calls for the desired types of representation as indicated on the EXECUTE INTERPØLATIØN statement.

707.3.2 ØVERLAY(INTEØLF,1,1) - DIEPAR

Sets execution parameters.

a. Establisnes the values of local variables.

b. Interrogates /CØNPARS/ and transmits the parameters to the other overlays via the /PRGPAR/ common block.

c. If errors are found in an execution parameter, DIEPAR writes a diagnostic message and prohibits the interpolation to which it applies.

707.3.3 ØVERLAY(INTEØLF,1,2) - DATPREP

Prepares mode shapes and coordinates.

a. Reads the subset mode shapes, subset freedom and node numbers, nodal correspondence table, and nodal data matrix into core.

b. Establishes the degrees of freedom as specified by the user or as indicated by the mode shapes matrix.

c. If nodes are defined in an analysis frame, DATPREP reads the analysis frame description from the local coordinate system matrix.

d. Picks up the user node number from subset freedom and node numbers matrix and calls NCUIR for associated internal node number and pointer to nodal data matrix.

e. Using this pointer, DATPREP picks up the node's global coordinates.

f. If an analysis frame indicator is on, DATPREP transforms the node coordinates from global frame to analysis frame.

g. Extracts the row of subset mode shapes associated with this retained freedom and places it in matrix [Φ],

h. Steps d-g are repeated for all freedoms.

i. Writes the mode shapes, [Φ], coordinates, and associated user node numbers onto SC00RNF.

707.3.4 ØVERLAY(INTEØLF,1,3) - SURFSPL

a. Reads mode shapes, [Φ], and increases row dimension by three (required by PLATEI routine).

b. Reads global coordinates, [XYZ], and extracts X-coordinates, and Y-coordinates.

c. Calculates the length of the coefficient matrix and allocates core for it.

d. Calls PLATEI to generate the surface spline array of coefficients, using as the interpolating function the small deflection equation of an infinite pinned plate.

e. If the analysis frame indicator is on, SURFSPL appends the rotation matrix and origin location to the coefficient matrix.

f. Writes the coefficient matrix on random file INTERNF.

g. Appends to the interpolation table matrix the coefficient matrix index name, size, and number of modes, and writes the table on random file INTERNF.

707.3.5 ØVERLAY(INTEØLF,1,4) - MØTIØNA

a. Reads mode shapes, [ϕ], coordinates, [XYZ], and user node number array.

b. Deletes identical node numbers and coordinates from node and coordinate arrays.

c. Establishes the out-of-plane axis and picks up the in-plane coordinates.

d. Orders y-coordinate values in monotonic increasing sequence using routine ØRDER.

e. Establishes the node location and motion arrays.

f. Establishes the definition point location and imaginary reference line slope arrays.

g. Calculates the length of the array of coefficients and allocates core for it.

h. Calls MØTAXI to generate the motion axis array of coefficients using as the interpolating function a cubic spline along a continuous planar curve.

i. If the analysis frame indicator is on, MØTIØNA appends the rotation matrix and origin location to the array of coefficients.

j. Writes the coefficient matrix on random file INTERNF.

k. Appends to the interpolation table matrix the coefficient matrix index name, size, and number of modes for this type of representation and writes the table on random file INTERNF.

707.3.6 ØVERLAY(INTEØLF,1,5) - PØLY

a. Calculates the length of the coefficient matrix and allocates core for it.

b. Inserts the following information in the coefficient matrix

1. number of elements in the array
2. 10HPØLYNØMIAL for type of representation
3. pointer to transformation matrix (analysis frame)

4. number of modes
5. degree of polynomial
6. coefficients for the polynomial

c. Writes the coefficient matrix o random file INTERNF.

d. Appends to the interpolation table matrix the coefficient matrix index name, size, and number of modes for this type of representation and writes the table on random file INTERNF.

707.3.7 ØVERLAY(INTEØLF,1,6) - MØTIØNP

a. Reads mode shapes, [Φ], and coordinates, [XYZ].

b. Establishes motion arrays and moves values from the mode shapes matrix, [Φ].

c. Calculates the length of the coefficient matrix and allocates core for it.

d. Calls MØTIØNP routine to generate the motion point coefficients, using as the interpolating function a rigid transfer of motions of a datum point.

e. If the analysis frame indicator is on, MØTIØNP appends the rotation matrix and origin location to the coefficient matrix.

f. Writes the coefficient matrix on random file INTERNF.

g. Appends to the interpolation table matrix the coefficient matrix index name, size, and number of modes for the type of representation and writes the table on random file INTERNF.

707.3.8 ØVERLAY(INTEØLF,1,7) - DATPRE2

Prepares the node coordinates and generates the identity mode shapes for the AIC option.

a. Reads the nodal subsets, nodal correspondence table, and nodal data matrix into core.

b. If nodes are defined in an analysis frame, DATPRE2 reads the local coordinate system matrix and picks up the analysis frame description.

c. Establishes the degrees of freedom either as specified by the user or by default.

d. Establishes the size of coordinate and identity modes matrices.

e. Loops on the number of nodes, steps f through h.

f. Picks up the user node number and pointer to nodal data matrix.

g. Picks up the node's global coordinates.

h. If the analysis frame indicator is on, DATPRE2 transforms the node location from global frame to analysis frame.

i. Writes coordinates and nodes on SC00RNF.

j. Generates identity modes matrix and writes it to SC00RNF.

707.3.9 ØVERLAY(INTEØLF,1,10) - BEAMSPL

This overlay generates the coefficient matrix for the beam spline method. The program will perform the following operations:

a) Read mode shapes, coordinates, node numbers, and nodal correspondence table.

b) Establish valid DØF and out-of-plane indicator.

c) Establish x-coordinate, y-coordinate, z-translation, x-rotation, and/or rotation arrays (position numbers) with its values.

d) Establish pointer array of 1) pointer to row number where first node starts, 2) number of nodes on Ith beam, and 3) indicator for extrapolating inboard and/or outboard.

e) Loop on number of beams.

f) Read Ith beam subset.

g) Establish user node number array, number of nodes, and pointer to Ith beam location in modes matrix.

h) Pick up x- and y-coordinates for each node in the Ith beam.

i) Order y-coordinates in ascending order.

j) Loop on number of nodes to pick up Tz, Rx, and/or Ry by searching modes matrix for a matching user node number.

k) End of beam loop.

l) Calculate length of coefficient array.

m) Call subroutine BEAMI generating all information on the coefficient array.

n) Establish table matrix and write on INTERNF

o) Establish auxiliary ID and identification of coefficient array and write on INTERNF.

707.4 COMMON BLOCK USAGE

NAME	OVERLAY: INTEGLF								
	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,10
KERROR	x	x	x	x	x	x	x	x	
COMPARS	x	x	x	x	x	x	x	x	
KQBUFP	x	x	x	x	x	x	x	x	
KQRNDM	x	x	x	x	x	x	x	x	
PRGPAR	x	x	x	x	x	x	x	x	
POLYCOM	x	x	x			x		x	
BEAMCOM	x	x						x	

707.4.1 /KERROR/,/COMPARS/,/DQBUFP/,/KQRNDM/ are ATLAS system common blocks, described in section 100.2.

707.4.2 /PRGPAR/ - Contains parameters which control the execution of the module. Values are established by defaults and the parameters on the EXECUTE INTERPOLATION statement.

1. ISET - Stiffness data set number
2. ISTAGE - Boundary condition stage number
3. NAMSUB - index name for mode subset
4. IMETHØD - indicator for method of interpolation
   = 1, surface spline
   = 2, motion axis
   = 3, polynomial
   = 4, motion point
5. ICHECK - indicator for checkout printing
   = 0, no printing
   = 1, checkout printing
6. IVSET - Vibration set number
7. ITX - freedom indicators-x,y,z translation,
8. ITY x,y,z rotation
9. ITZ =0, not selected
10. IRX =1, selected
11. IRY
12. IRZ
13. NDEF - number of definition points and angles (for motion axis only)
14. IDEFPTS(40) -user node number defining the Ith motion axis definition point (for motion axis only)
15. ANGLES(40) -angle of reference line at Ith definition point (for motion axis only)
16. IKRIN -error return indicator from DIEPAR
   = 0, no error
   = 1, error detected in execution parameters for this subset, continue on with next subset data.
17. ICØN -pointer to the last element of /CØNPARS/ that was picked up for the current subset
18. ITRAN -transformation matrix indicator
   = 0, no transformation
   = 1, transformation matrix available in matrix RØTRAN
19. RØTRAN(3,4) -rotation and translation transformation matrix. Transformation is from local to global. The last column is the translation matrix.

/PØLYCØM/ - contains the polynomial representation coefficients which are established by the EXECUTE INTERPØLATIØN specifications.

1. NMØDES - the number of polynomial mode shapes requested.
2. CØEF(6,6,5) - the polynomial coefficients for up to 5 mode shapes. CØEF(I+1,J+1) is the coefficient of the x**I,y**J term (0≤I,J≤4)

/BEAMCØM/ - contains the beam spline variable parameters of which the values are established by the EXECUTE INTERPØLATIØN statement.

1. NBEAM - Number of beams (or nodal subsets forming beam nodes)
2. IBEAM(100) - An array of information for each beam. A typical word contains the following information pertaining to the Ith beam.

   Bits 59-18 - Index name of nodal subset associated with the Ith beam.

   Bits 17-2 - Not used.

   Bit 1 - Indicator for inboard extrapolation:
   = 0, Do not extrapolate
   = 1, Extrapolate inboard

   Bit 0 - Indicator for outboard extrapolation:
   = 0, Do not extrapolate
   = 1, Extrapolate outboard

708. LOADS PROCESSOR

708.1 GENERAL INFORMATION

708.1.1 Purpose

1. Generate complete loadcase correspondence table for all requested user ID's.
2. Produce direct nodal loads in analysis coordinate system.
3. Produce specified nodal displacements from all sources.
4. Calculate statically equivalent nodal loads (consistent nodal loads for BRICKs) from element pressure loads, and output these in the node analysis system.
5. Calculate consistent nodal loads from nodal and element thermal loading, and output these in the node analysis system.
6. Generate loads and displacements for combined loadcases.
7. Calculate the resultants for all loadcases requested (basic or combined).
8. Calculate nodal accelerations due to rotational inertia loads and multiply them by element mass matrices to compute nodal forces.

708.1.2 Access

This module is called from the ATLAS Control program by the EXECUTE LØADS(plist) statement (sec. 234, ref. 1-1).

708.2 Matrix Activity

	Name	Description	File
Input			
1	LCØØRba	Load Case Correspondence	DATARNF
2	LN001ba	Direct Nodal Loads	DATARNF
3	LE001ba	Element Distributed Loads	DATARNF
4	LT001ba	Nodal Thermal Loads	DATARNF
5	LTLCCba	Thermal Load Case Correspondence	DATARNF
6	LNTLTba	Nodal Thermal Load Index	DATARNF
7	LEDIRba	Element Load Directions	DATARNF
8	LD001ba	Specified Displacements	DATARNF
9	LCØMBba	Combined Load Cases	DATARNF
10	KCØØRba	Load Case Corr.(from BC)	DATARNF

11	KD001ba	Specified Displacements (from BC)	DATARNF
12	KLCT00a	Element Correspondence	DATARNF
13	KSF001a	Element Data	DATARNF
14	KNC100a	Nodal Correspondence	DATARNF
15	KNØALTa	Nodal Data	DATARNF
16	KLØCØØa	Local Coordinate Data	DATARNF
17	KMELNØa	Element Nodal Data	DATARNF
18	LRØTNba	Rotational Inertia Loads Matrix	DATARNF
19	MA0001a	Element Mass Matrices	MASSRNF
20	LU000ba	Element Thermal Loads	DATARNF
21	GKS001a	Detailed Geometry	DATARNF
22	GP0001a	Element Stress	STIFRNF
23	KA0001a	Element Stiffness	STIFRNF

Output

1	RSULTba	Resultant Matrix	LØADRNF
2	LA001ba	Nodal Loads Matrix	LØADRNF
3	DCØØRba	Load Case Correspondence	LØADRNF
4	ISC01ba	Initial Stress Control	LØADRNF
5	IS001ba	Initial Stress	LØADRNF
6	DA001ba	Specified Displacements	LØADRNF
7	ELCØNba	Element Temperature Control	LØADRNF
8	ELØØØba	Element Temperature	LØADRNF
9	LFAVØØba	Load Freedom Activity Vector	LØADRNF

A detailed description of the matrix formats is contained in reference 1-2.

708.3 PROGRAM METHOD

The LØADS module consists of a primary overlay, MUTHALD, and nine secondary overlays, BEGIN, CØØR, NØDE, DISP, ELEM, THERM, RØT, MUITØT and FIN.

708.3.1 ØVERLAY (LØADØLF,1,0) - MUTHALD (FØRTRAN)

a) loops through secondary overlays

b) carries module common blocks

708.3.2 ØVERLAY (LØADØLF,1,1) - BEGIN (SNARK)

a) establishes values of the /NUGGETS/ common block. This is done by interrogating the /CØNPARS/ block.

b) establishes the values of the /SETCØM/ common block by determining requested SETs and STAGEs from /CØNPARS/

c) prints opening comment

d) puts a list of requested loadcases on SC00RNF if specific loadcases are requested through /CØNPARS/. This list goes out with the name 2HLC.

e) When recalled for each SET and STAGE, resets the values of the appropriate variables in common blocks /NUGGETS/ and /SETCØM/.

f) In the event of erroneous information being passed in /CØMPARS/, BEGIN prints a message, increments KERRØR and terminates processing.

708.3.3 ØVERLAY(LØADØLF,1,2) - CØØR (SNARK)

a) Produces the loadcase correspondence table for the current SET and STAGE. This table is the intersection of the requested loadcases with the union of the loadcase correspondence tables from the boundary condition data (B.C.) preprocessor and the LOADS preprocessor.

b) Produces a scratch combined loadcase matrix which is written on SC00RNF with index name DCØMBXX. The format of this matrix is explained in comments in the code of subroutine DCØMBUP.

c) Prints a list of loadcases being processed for this SET and STAGE.

708.3.4 ØVERLAY(LØADØLF,1,3) - NØDE (SNARK)

a) Checks if there are any nodal loads requested for this SET/STAGE.

b) Reads in nodal data, local coordinate data, loadcase correspondence, RSULT matrix, and the scratch combined loadcase matrix.

c) Loops through the sources of nodal loads and reads the source loadcase correspondence table and nodal loads data.

d) Calls DUST to calculate the nodal loads.

708.3.4.1 DUST (FØRTRAN subroutine)

a) DUST reads the incoming nodal loads and converts the loadcase to the internal loadcase number. If this loadcase is not a component of a requested loadcase, these loads are ignored.

b) The loads are rotated to the global system and the resultant is incremented, for this loadcase and any combined cases of which it is a component.

c) Unless user requested resultants only, the load is rotated to the node analysis system.

708.3.5 ØVERLAY(LØADØLF,1,4) - DISP (SNARK)

a) Checks to see if there are any specified displacements for this SET/STAGE.

b) Reads in loadcase correspondence and scratch combined loadcase matrix.

c) Loops through the sources of specified nodal displacements (LØADS preprocessor and B.C. data preprocessor) and reads in source loadcase correspondence table and displacements.

d) Calls FRIG to move displacements to output matrix.

708.3.5.1 FRIG (FØRTRAN subroutine)

a) FRIG reads the nodal displacements and converts the loadcase names to internal load numbers. If the loadcase has not been requested and is not a component of a requested combined case, this displacement is skipped. Otherwise, the displacement is output for this case and any combined load cases.

708.3.6 ØVERLAY(LØADØLF,1,5) - ELEM (SNARK)

a) Checks if there are any pressure loads for this SET/STAGE.

b) Reads in element data, nodal data, local coordinate data, loadcase correspondence, resultant matrix, and scratch combined loadcase matrix.

c) Loops through sources of pressure loads and reads in source loadcase correspondence table, element pressure data, and element pressure directions.

d) Calls SUBEL to calculate the equivalent nodal loads.

708.3.6.1 SUBEL (FØRTRAN subroutine)

a) Loops through the element pressure data until it comes to a valid loadcase (either requested or a component of a requested case).

b) Sets up the element geometry in common block /USUEL/. BRICKs require subroutine BRKGEØM, and common block /BRICKLE/.

c) Reads element pressures and pressure direction vector from input matrices into /USUEL/.

d) Calls individual element routines to compute statically equivalent nodal loads (for BRICK, consistent nodal loads; see comments in GLØBPR and MUG). These routines also compute the element's contribution to the loads resultant.

e) Increments resultant for appropriate loadcases.

f) Unless the user requested resultants only, rotates nodal loads to analysis system and writes them out.

708.3.7 Thermal Loads

Thermal loads are calculated in a series of four overlays.

OVERLAY(LØADØLF,1,20) - THERMEL (SNARK)

This overlay reads in both nodal and element thermal loads matrices form DATARNF and combines and reformats the information to:

1. Produce ELCØNba and EL000ba for use by design.

2. Produce scratch matrices with element temperature data for use by the following three overlays.

ØVERLAY(LØADØLF,1,21) - THERMV (SNARK)

Computes initial stress and equivalent nodal loads for RØD and BEAM elements using element stress and stiffness matrices from stiffness.

ØVERLAY(LØADØLF,1,22) THERMU (SNARK)

Computes initial stresses and equivalent nodal loads for the SPAR, GPLATE, BRICK, AND SRØD elements.

ØVERLAY(LØADØLF,1,23) THERMU2 (SNARK)

Computes initial stresses and equivalent nodal loads for the PLATE, CØVER, CPLATE and CCØVER elemen

708.3.7.1 CALØRIE(FØRTRAN subroutine)

a) Loops through elements calling BTU to set up geometry for each BRICK.

b) Loops through requested thermal loadcases for each BRICK calling STRAIN to set up vector of thermal strains.

c) Calls TBRICK to generate consistent nodal loads and initial stresses.

d) Increments resultant and, unless user requested resultants only, rotates loads to analysis system.

708.3.8 ØVERLAY(LØADØLF,1,7) - RØT (SNARK)

Calculates nodal accelerations and writes them on scratch

a) Sets up a local scratch matrix LCRALab which contains all information needed from LRØTNba for the calculation of nodal accelerations. The subroutines TØTLRØT and RØTCNTR set up this scratch matrix.

b) Calculates the nodal accelerations in the FØRTRAN subroutine ACCRØT where an acceleration control matrix ACCØNba and acceleration matrices AC001ba are created. Subroutines AGLLIN and AGLANG are used to compute linear and angular accelerations, respectively.

c) Processes combined loadcases in the routine CØMBRØT.

708.3.9 ØVERLAY(LØADØLF,1,10) - MULRØT (SNARK)

Calculates nodal forces by multiplying nodal accelerations and element masses.

a) Modify LCRALab in the routine FMALC for use later on.

b) Set up pointers for multiplication in the routine MULSET. Also set up loadcase ID data.

c) Compute nodal forces in the routine MULDØ, using MAMULT as the multiplication tool and MAEXPND as a tool to expand the lower triangular mass matrices into rectangular matrices.

d) Combine loadcase ID data and nodal forces in the routine LARØT. Also update the resultant matrix RSULTba.

708.3.10 ØVERLAY(LØADØLF,1,77) - FIN (SNARK)

a) Outputs the final resultant matrix to LØADRNF and prints it.

708.4 COMMON BLOCK USAGE

NAME	OVERLAY; LOADOLF (1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,7)
/CONPARS/	x	x	x	x	x	x	
/KERROR/	x	x	x	x	x	x	x
/KQBUFP/	x	x	x	x	x	x	
/KQRNDM/	x	x	x	x	x	x	x
/NUGGETS/	x	x	x	x	x	x	x
/SETCOM/	x	x					
/USUEL/						x	
/BRICKLE/						x	
/MACOM/							
/HOTS/							
/ARMYROT/							x
/MULPARS/							

NAME	OVERLAY: LOADOLF (1,10)	(1,20)	(1,21)	(1,22)	(1,23)	(1,77)
/CONPARS/		x	x	x	x	x
/KERROR/	x	x	x	x	x	x
/KQBUFP/		x	x	x	x	x
/KQRNDM/	x	x	x	x	x	x
/NUGGETS/	x	x	x	x	x	x
/SETCOM/						
/USUEL/						
/BRICKLE/						
/MACOM/	x	x	x	x	x	
/HOTS/		x	x	x	x	
/ARMYROT/						
/MULPARS/	x					

708.4.1 /KERROR/,/CONPARS/,/KQBUFP/,/KQRNDM/ are ATLAS system common blocks, described in section 100.2.

708.4.2 LOADS Processor Common Blocks

/NUGGETS/ is used throughout the module. It contains buffer sizes, matrix names, and control parameters. It is initialized by BEGIN.

/SETCØM/ preserves information for BEGIN as to which SETs and STAGEs have been requested but not yet executed.

/USUEL/ is used by ELEM to carry element pressure loading information such as nodal geometry, pressure values, and nodal loads.

/BRICKLE/ is used by the BRICK pressure loading routines BRKGEØM, GLØBPR, and MUG, to carry additional BRICK information not kept in /USUEL/.

/MACØM/ is used by the ATLAS matrix multiply routine MAMULT.

/HØTS/ is used by THERM and its subroutines to carry element geometry, thermal strains, nodal temperatures, nodal loads, and other information needed to compute thermal loads.

/ARMYRØT/ is used by RØT, ACCRØT, and CØMERØT to carry information on the number of loadcases and nodes and matrix row indices.

/MULPARS/ is used by MULRØT, MULSET, and MULDØ to pass data on the number of loadcases, nodes, matrix indices and partition numbers, and input/output control.

709. MACHBOX PROCESSOR

709.1 GENERAL INFORMATION

709.1.1 Purpose

The MACHBØX module calculates unsteady aerodynamic loadings on coplanar or non-coplanar surfaces in supersonic flow. The loadings are represented by a generalized air force matrix which is used by other ATLAS modules to provide a flutter solution (ref. 709-1).

709.1.2 Access

This module is called from the ATLAS control program by the EXECUTE MACHBØX(plist) statement (sec. 236, ref. 1-1).

709.2 Matrix Activity

	Name	Description	File
Input			
1	BØXi	Machbox input data	DATARNF
2	Cddd	Interpolation coefficient array	INTERNF
Input/Output			
1	AICCeee	Velocity potential AIC array	MACHRNF
2	AICCINDX	AIC index array	MACHRNF
3	AICMeee	Spatial AIC pointer array	MACHRNF
4	AICPeee	Planar AIC array	MACHRNF
5	AICWeee	Upwash AIC ARRAY	MACHRNF
6	AICVeee	Sidewash AIC array	MACHRNF
Output			
1	ACMij	Control matrix	MACHRNF
2	ACNijkl	AIC index pointers	MACHRNF
3	BLnijkl	Box lifts	MACHRNF
4	BØXijkT	Box code array for nonplanar tail if present	MACHRNF
5	BØXijkW	Box code array for wing (and coplanar tail if present)	MACHRNF
6	CMnijkl	Sectional moments	MACHRNF
7	DWPijkl	Normal wash pointers	MACHRNF

8	EXPij	Common block array containing all input data and execution parameters	MACHRNF
9	GACijkl	Generalized aerodynamic coefficients (real)	MACHRNF
10	GCIijkl	Generalized aerodynamic coefficients (imaginary)	MACHRNF MACHRNF
11	GFOijkl	Generalized forces	MACHRNF
12	ISPijk	Planform column pointers	MACHRNF
13	LNnijkl	Wing (Wing-Tail) lower surface normal wash	MACHRNF
14	LTnijkl	Non-coplanar tail lower surface normal wash	MACHRNF
15	MØnijkl	Mode slopes and deflections	MACHRNF
16	MPTijk	Planform row pointers	MACHRNF
17	PCnijkl	Pressure difference coefficients	MACHRNF
18	PSTijkl	Subdivided normal wash pointers for tail	MACHRNF
19	PSWijkl	Subdivided normal wash pointers for wing	MACHRNF
20	SACijkl	Smoothed generalized aerodynamic coefficients (real)	MACHRNF
21	SBnijkl	Smoothed box lifts	MACHRNF
22	SCIijkl	Smoothed generalized aerodynamic coefficients (imaginary)	MACHRNF
23	SFOijkl	Smoothed general forces	MACHRNF
24	SLnijkl	Sectional lifts	MACHRNF
25	SMnijkl	Smoothed sectional moments	MACHRNF
26	SPnijkl	Smoothed pressure differential coefficients	MACHRNF
27	SSnijkl	Smoothed sectional lifts	MACHRNF
28	STnijkl	Non-coplanar tail subdivided normal wash	MACHRNF
29	SUnijkl	Wing (Wing-Tail) subdivided normal wash	MACHRNF
30	SVnijkl	Smoothed velocity potentials	MACHRNF
31	UNnijkl	Wing (Wing-Tail) upper surface normal wash	MACHRNF
32	UTnijkl	Non-coplanar tail upper surface normal wash	MACHRNF
33	VPnijkl	Velocity potentials	MACHRNF
34	WSnijkl	Off planform wash samples	MACHRNF

A detailed description of the matrix formats is contained in reference 1-2.

709.3 PROGRAM METHOD

The MACHBØX module consists of a primary overlay and eight secondary overlays.

709.3.1 ØVERLAY (MACHØLF,1,0) - CØNTRØL

CONTRØL's function is to control and direct the calling sequence of the secondary overlay. In addition, it checks the error return from each secondary overlay and determines what recovery is to be used if an error is encountered.

709.3.2 ØVERLAY (MACHØLF,1,1) - DATAPP

DATAPP first examines the /CØNPARS/ common block for the execution parameters. The data case input information specified on the execution card is then read from DATARNF. A comprehensive error analysis is completed. If no errors are found, execution continues.

709.3.3 ØVERLAY (MACHØLF,1,2) - GEØMBX

Program GEØMBX processes all of the geometry data. Leading and trailing edge data is transformed to a non-dimensional coordinate system. Planform and diaphragm box code patterns are determined in subroutines BXCDPF and BXCDI. The fractional on-planform portion of all boxes cut by a planform edge is determined by subroutine GMAREA, which in turn calls subroutines ALPHAC and NTRCEP. If spatial AIC's are necessitated by non-zero dihedral angles or vertical separation of wing and tail, integer array (MUAIC) is determined for each AIC set required. It serves as a map, so that only those AIC values needed will be calculated. The MUAIC arrays are computed in subroutines PWWAIC and PWTAIC. All resulting arrays are written on scratch file IGEØSC.

709.3.4 ØVERLAY (MACHØLF,1,3) - MØDES

The primary function of program MØDES is to determine mode shapes (slopes and deflections) for the problem. The planform information is read from a scratch file created in program GEØMBX. Subroutine RØPER is called to compute row pointers for rowwise storage of box center modal values. If rigid body modal input was specified in the input data, an interpolation coefficient array composed of that data is assembled in core. If rigid body input was not specified, an

interpolation coefficient array specified by the user is read from INTERNF. In either case the interpolation coefficient array, with the box center locations, is passed to subroutine AINTL which returns the slopes and deflections at the box centers.

Program MØDES also has an option to use an input array of thickness slope function values at box centers derived from "Piston Theory." This array is stored as part of the data case by the preprocessor, and is valid only for a user specified Mach number. These values are used in an equation that computes the thickness correction factor.

$$Z(x,y) = 1 + \frac{GAMMA + 1}{2} * M * \frac{dz}{dx} \qquad \text{eq. (1)}$$

where

GAMMA	is the ratio of specific heats for a perfect gas (1.40)
M	is the Mach number
$\frac{dz}{dx}$	is the thickness slope function values
z(x,y)	is the thickness correction factor.

709.3.5 ØVERLAY (MACHØLF,1,4) - VICMAIN

Program VICMAIN determines whether aerodynamic influence coefficients must be calculated or retrieved from MACHRNF for each reduced frequency.

A parameter array is read from the geometry scratch file for each spatial AIC that is needed. The program then determines if an array already exists on MACHRNF. If it exists the array is read, expanded if necessary, and stored on scratch file IAICSC if spatial, or in labeled common if planar. If calculations are necessary, subroutine KERNEL is called to control the actual computations. KERNEL, in turn, calls RØMBER to do the integrations of FUNCT and VFUNC.

If AIC's are expanded or calculated, they are added to those already on MACHRNF.

709.3.6 ØVERLAY (MACHØLF,1,5) - NWVPMBX

Normal washes and velocity potentials at the box centers are calculated for each mode shape in program NWVPMBX. The necessary box patterns and geometry data are first read from the scratch file IGEØSC. The mode shape and velocity potential pointer array IPNTRM is read from scratch file MØDESC, and a pointer array for normal-washes, IPNTDW, is generated by subroutine PØINTER. These pointer arrays serve to associate a box location in a sparsely filled rectangular array with the corresponding mode, velocity potential or normal wash value in a singly dimensioned, densely filled array. A loop on mode shapes is entered next. The deflections and slopes at box centers are read from scratch file MØDESC into array DEFSL. Subroutine VELPØT is called to compute wing upper and lower surface normal washes and velocity potentials at box centers, and trailing edge velocity potentials. If a tail is being analyzed as well, the contributing wing normal-washes are determined and VELPØT is called again for the tail. If off-planform sampling was specified in the input data case, subroutine SMPLW is called to compute the sampling results. The velocity potential array and the trailing edge velocity potential array are written on scratch file.

709.3.7 ØVERLAY (MACHØLF,1,6) - SMØØTH

Program SMØØTH uses a least squares surface fitting technique to smooth the velocity potentials generated by program NWVPMBX. The velocity potentials are read from scratch file and fitted with a least squares polynomial by subroutine FITTER. The polynomial equation derived from the fitting is then used to compute an interpolated array of velocity potentials at planform box centers. The smoothed values are then written on scratch file IAICSC.

709.3.8 ØVERLAY (MACHØLF,1,7) - CHØRDF

Program CHØRDF uses a linear least squares fitting technique to smooth the velocity potentials along each chord. After the velocity potential array is read from scratch file, it is separated by chord. The values for each chord are changed to the numerical slope between the midpoint average values, and subroutine CURVE is called to fit a least squares polynominal curve through

these slopes. The polynominal equation is integrated at each box on the chord. The integral value becomes the smoothed velocity potential at that box. The smoothed values are reassembled and written on a scratch file IAICSC.

709.3.9 ØVERLAY (MACHØLF,1,10) - FØRCES

Program FØRCES calculates a generalized air force matrix for each reduced frequency. Planform information is read from the geometry and modes scratch files. The outermost loop on thickness slope functions is then entered. One set of thickness slope functions, defined at box centers by equation (1), is read from the thickness slope scratch file. If thickness slopes were not specified in the input data case, a unit array is used. Next a loop on mode shapes, used as weighting functions for the generalized force calculations, is entered. Each mode shape is read from the mode shape scratch file. In the inner loop the velocity potentials and trailing edge velocity potential arrays are read from scratch file. The box pattern for each surface is then examined row by row. Arrays for box lifts, sectional lifts, sectional moments, pressure difference coefficients, velocity potentials, generalized aerodynamic coefficients, and generalized air forces are calculated and written on MACHRNF as requested by the user.

709.4 COMMON BLOCK USAGE

Name	ØVERLAY: MACHØLF								
	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,10)
/KERRØR	x								
/CØNPARS/	x	x							
/KQRNDM/	x	x	x	x	x	x	x	x	x
/KQBUFP/	x								
/MATRNAM/	x	x	x	x	x	x	x	x	x
/GEØMTY/	x	x	x	x	x	x	x	x	x
/GEØM2/	x	x	x	x	x	x	x	x	x
/PLANXY/	x	x	x	x	x	x	x	x	x
/ARRAYS/	x	x	x	x	x	x	x	x	x
/SAMPLW/	x	x	x	x	x	x	x	x	x
/MØDES/	x	x	x	x	x	x	x	x	x
/BØX/	x	x	x	x	x	x	x	x	x
/TSLØPE/	x	x	x	x	x	x	x	x	x
/EXEC/	x	x	x	x	x	x	x	x	x
/MACH/	x	x	x	x	x	x	x	x	x
/KVAL/	x	x	x	x	x	x	x	x	x
/LEVEL/	x	x	x	x	x	x	x	x	x
/WRITE/	x	x	x	x	x	x	x	x	x
/KERN/	x	x	x	x	x	x	x	x	x
/FILES/	x	x	x	x	x	x	x	x	x
/TAPEIØ/	x	x	x	x	x	x	x	x	x
/NERRØR/	x	x	x	x	x	x	x	x	x
/CHECKØUT/	x	x	x	x	x	x	x	x	x
/RANDIØ/	x	x	x	x	x	x	x	x	x
/MASK/	x	x	x	x	x	x	x	x	x
/RWBUFF/	x					x	x	x	x
/AICCØM/	x				x	x			
/MUAICS/			x			x			
/EDGES/			x						
/LAREA/			x						
/INDEX/			x				x	x	
/VICPAR/					x				
/TØC/					x				
/BESFUN/					x				
/PAICS/						x			
/AICS/						x			
/DELTAP/						x			
/NWASHES/						x			
/SNWASH/						x			
/SWASH/						x			
/LRØT/						x			

709.4.1 /KERRØR/,/CØNPARS/,/KQRNDM/,/KQBUFP/ are ATLAS system common blocks described in section 100.2.

709.4.2 Module common blocks

/MATRNAM/ - contains data case title.

/GEØMTY/ - contains geometry data for first surface.

/GEØM2/ - contains geometry data for second surface.

/PLANXY/ - contains numbers of edge definition points and arrays of edge definition points.

/ARRAYS/ - contains size limits of arrays used within MACHBØX.

/SAMPLW/ - contains data for off-planform wash sampling.

/MØDES/ - contains modal data needed for execution of a data case.

/BØX/ - contains input variables determining box size and location.

/TSLØPE/ - contains information for applying thickness slopes correction.

/EXEC/ - contains parameters and option indicators from execution card.

/MACH/ - contains Mach number index, number of Mach numbers, array of Mach numbers, and current Mach number being used.

/KVAL/ - contains information about K values being used.

/LEVEL/ - contains the array of retention level flags (logical).

/WRITE/ - contains flags for SNARK or READTP formatted I/Ø.

/KERN/ - contains data for generation of AIC's.

/FILES/ - contains names of ATLAS scratch files stored in local scratch file name variables.

/TAPEIØ/ - contains variables used for reading and writing of scratch files by routines READMX and WRTEMX.

/NERRØR/ - contains counter and flag for local error analysis

/CHEKØUT/ - contains print flags.

/RANDIØ/ - contains variables used for SNARK reading and writing on random files.

/MASK/ - contains masking constants.

/AICCØM/ - contains planar AIC array to be communicated between VICMAIN and NWVPMBX modules.

/RWBUFF/ - contains buffer array used by READMX and WRTEMX routines.

/MUAICS/ - contains MUAIC arrays (AIC maps) and other parameters needed for the generation of AIC arrays.

/EDGES/ - contain the leading and trailing edge x locations of the planform.

/LAREA/ - contains y coordinates of chord, and box flag.

/INDEX/ - contains planform column and row pointers.

/VICPAR/ - contains MUAIC array and other parameters use for generating an AIC array.

/TØC/ - contains the AIC index array.

/BESFUN/ - contains variables used in the evaluation of Bessel function.

/PAICS/ - contains variables and pointers associated with AIC arrays written on scratch file.

/AICS/ - contains the AIC arrays.

/DELTAP/ - contains velocity potential arrays and pointer.

/NWASHES/ - contains arrays used in normal wash calculation.

/SNWASH/ - contains arrays and variables used in subdivided normal wash calculations.

/SWASH/ - contains arrays of off-planform samples.

/EXCDES/ - contains box code array.

/IRØT/ - contains a subdivision pointer.

710. MASS PROCESSOR

710.1 GENERAL INFORMATION

710.1.1 Purpose

1) Compute element and total model mass properties
2) Generate diagonal and/or non-diagonal lumped mass matrices
3) Generate panel weight matrices
4) Generate the element mass matrices
5) Generate fuel and payload distribution data

710.1.2 Access

This module is called from the ATLAS control program via the EXECUTE MASS(plist) statement. (sec. 238, ref. 1-1)

710.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	ISSCsss	Substructure definition vector	DATARNF
2	ISSSCØR	Set/stage-substructure correspondence vector	DATARNF
3	KlØCØØa	Local coordinate data	DATARNF
4	KM00001	Material data	DATARNF
5	KMELNØa	Element nodal data	DATARNF
6	KNC100a	Nodal correspondence table	DATARNF
7	KNØALTa	Nodal data matrix	DATARNF
8	KPARMS1	Data set parameter matrix	DATARNF
9	KRFV0ba	Retained freedom vector	DATARNF
10	KSF001a	Element data	DATARNF
11	MCMASga	Concentrated mass data	DATARNF
12	MCMNØDa	Concentrated mass nodes	DATARNF
13	MCØNDTa	Condition data	DATARNF
14	MFATUDa	Fuel altitude data	DATARNF
15	MFCØNDa	Fuel condition data	DATARNF
16	MFLØADa	Fuel management data-loading	DATARNF
17	MFMUSEa	Fuel management data-usage	DATARNF
18	MFULffa	Fuel element data	DATARNF
19	MHØLDSa	Cargo hold geometry data	DATARNF
20	MMELNØa	Element nodal data	DATARNF
21	MLØDppa	Payload element data	DATARNF
22	MLUMP0a	Mass lumping data	DATARNF
23	MPANLha	Auxiliary panel data matrix	DATARNF
24	MPARMS1	Data set parameter matrix	DATARNF
25	MPCØNDa	Payload condition data	DATARNF

26	MPLØADa	Payload loading data	DATARNF
27	MPLØCLa	Seat data - local coordinates	DATARNF
28	MPNØCTa	Seat data - correspondence table	DATARNF
29	MPNØDMa	Seat data - coordinates	DATARNF
30	MPSETha	Panel subset data	DATARNF
31	MSF001a	Element data	DATARNF
32	MTANKSa	Fuel tank geometry data	DATARNF
33	MWTFACa	Weight factors	DATARNF
34	MWTFTta	Weight factor tables	DATARNF
35	SEKddda	Stiffness subset data	DATARNF
36	SEMddda	Mass subset data	DATARNF

Output

1	MA000 1a	Element mass matrices	MASSRNF
2	Cg0001a	Concentrated mass matrices	MASSRNF
3	Mxxxxxx	Element weight data	MASSRNF
4	Gxxxxxx	Element geometry data	MASSRNF
5	IDxx00a	Element index matrix	MASSRNF
6	GFxxxxx	Fuel tetrahedron geometry data	MASSRNF
7	GPxxxxx	Payload tetrahedron/scalar data	MASSRNF
8	TØTLWTa	Weight summary matrix	MASSRNF
9	TAPLWTa	Condition summary matrix	MASSRNF
10	MDCqqqa	Mass matrices	MASSRNF
11	MREDsss	Mass matrices (substructure)	MASSRNF
12	MFAV00a	Freedom activity vector	MASSRNF
13	FTxxxxx	Fuel tables	MASSRNF
14	FTINDxx	Fuel table index matrix	MASSRNF
15	FVECffa	Fuel vectors	MASSRNF
16	CVECppa	Cargo vector	MASSRNF
17	PVECppa	Passenger vectors	MASSRNF
18	CNMASga	Concentrated mass data	SC00RNF/ MASSRNF
19	AXxxxxx	Auxiliary panel data	SC00RNF/ MASSRNF
20	LMxxxxx	Lumped mass data	SC00RNF/ MASSRNF

A detailed description of the matrix formats is contained in reference 1-2.

710.3 PROGRAM DESCRIPTION:

The MASS module consists of a primary overlay and twenty secondary overlays.

710.3.1 ØVERLAY (MASSØLF,1,0) - MASSCTL

a. Controls execution of the MASS module.

710.3.2 ØVERLAY (MASSØLF,1,1) - GEØMTRY

a. Reads the stiffness, mass, fuel and payload element matrices and multiplies the densities by the requested weight factor.

b. Extracts the coordinates of the element node .

c. Sets up the element geometry matrices. (Gxxxxxx)

710.3.3 ØVERLAY (MASSØLF,1,2) - TØTALWT

a) Computes the weight, center of gravity, and weight moments of inertia for the stiffness, mass, fuel and payload elements.

b) Sets up the element mass data matrices (Mxxxxxx).

c) Sets up the element index matrices (IDxx00a).

d) Extracts the concentrated mass nodal coordinates and sets up the concentrated mass data matrices (CNMASga).

e) Computes the total weight, center of gravity, and weight moments of inertia for each element set and for each concentrated mass subset.

f) Sets up the weight summary matrix (TØTLWTa).

710.3.4 ØVERLAY (MASSØLF,1,4) - PANELWT

a) Sets up the array of cutting planes corresponding to the auxiliary weight panels.

b) Calls PANEL3 to assemble the stiffness, mass, fuel and payload element weight data matrices for each element subset.

c) PANEL3 calls PANEL2 which sums the weights of each element and/or part of element within a polygonal cylinder to get the panel weight.

d) Weight data matrices corresponding to each auxiliary panel subset are written on scratch file for use by the ASSMBLY program.

710.3.5 ØVERLAY (MASSØLF,1,5) - GENMASS

a) Assembles the stiffness, mass, fuel and payload element weight data matrices by computing the weight within a polyhedron defined around each retained node such that each infinitesimal weight is lumped at the closest retained node.

b) Weight data matrices corresponding to each set of retained nodes are written on scratch file for use by the ASSMBLY program.

710.3.6 ØVERLAY (MASSØLF,1,6) - ASSMBLY

a) Reads the weight data matrices from scratch file for the element subsets and assembles a weight data matrix corresponding to each defined mass condition.

b) Adds the concentrated mass data to each weight data matrix as required.

c) Sums the terms in each weight data matrix and compares the total weight to the total weight of the element subsets that make up the mass condition.

d) Writes the assembled weight data matrices on scratch file for use by the MRGMASS program.

e) Sets up the final panel weight matrices corresponding to the panel weight conditions and writes them on MASSRNF.

f) Sets up the condition summary matrix (TAPLWTa) and writes it on MASSRNF.

710.3.7 ØVERLAY (MASSØLF,1,7) - MRGMASS

a) Reads the assembled weight data matrices from scratch file.

b) Converts weight to mass and assembles the diagonal and/or nondiagonal mass matrix.

c) The diagonal matrices are generated by extracting the required diagonal terms from the weight data matrices. The resulting matrix is a user matrix written on MASSRNF.

d) The nondiagonal matrices are generated by transferring the mass from the center of gravity to the retained node. All resulting off-diagonal terms are retained. The resulting matrix is a user matrix written on MASSRNF.

710.3.8 ØVERLAY (MASSØLF,1,10) - LUMPGEN

a) Computes the element lumped mass matrices for stiffness and mass elements.

b) Computes lumped mass matrices for each defined concentrated mass subset.

c) The lumped mass matrices (MA0001a and Cg0001a) are written on MASSRNF.

d) The total weight of each element/concentrated mass combination is computed and added into the condition summary matrix (TAPLWTa)

e) The mass freedom activity vector (MFAV00a) for the elements and each concentrated mass subset is created and written on MASSRNF.

710.3.9 ØVERLAY (MASSØLF,1,11) - CØNSGEN

a) This is a dummy overlay reserved for consistent mass matrix generation.

710.3.10 ØVERLAY (MASSØLF,1,12) - SUBMASS

a) Reads the mass matrices for each lower level substructure and writes them out as user matrices with the names MREDsss, where "sss" is the substructure number.

b) Computes the total weight, center of gravity and weight moments of inertia for the top level substructure.

c) Sets up the condition summary matrix (TAPLWTa) for the top level substructure.

710.3.11 ØVERLAY (MASSØLF,1,13) - FUELTET

a) Forms the fuel tank tetrahedron scratch matrices.

710.3.12 ØVERLAY (MASSØLF,1,14) - FUELTAB

a) Forms the fuel table matrices and their index matrix. (FTxxxxx, FTINDxx)

710.3.13 ØVERLAY (MASSØLF,1,15) - FUELMAN

a) Forms all scratch matrices related to fuel management.

710.3.14 ØVERLAY (MASSØLF,1,16) - FUELELM

a) Forms the fuel geometry and index matrices for each condition. (GFxxxxx, IDxxxxx)

710.3.15 ØVERLAY (MASSØLF,1,17) - PAYTET

a) Forms the cargo hold tetrahedron scratch matrices.

710.3.16 ØVERLAY (MASSØLF,1,20) - PAYTAB

a) Forms the cargo table scratch matrices and their index matrix.

710.3.17 ØVERLAY (MASSØLF,1,21) - PAYLØD

a) Forms the cargo and passenger loading scratch matrices.

710.3.18 ØVERLAY (MASSØLF,1,22) - PAYELM

a) Forms the payload geometry and index matrices for each condition (GPxxxxx, IDxxxxx)

710.3.19 ØVERLAY (MPESØLF,1,23) - PAYVEC

a) Forms the fuel, cargo, and passenger loading vectors for the GRAPHICS module. (FVecffa, CVECppa, PVecppa)

710.3.20 ØVERLAY (MASSØLF,1,24) - PAYBAL

a) Forms the payload geometry and index matrices for the generated balance conditions. (GPxxxxx, IDxxxxx)

710.3.21 ØVERLAY (MASSØLF,1,77) - SETCNTL

a) Sets up the MASS Processor control common block CØNMAT.

710.4 COMMON BLOCK USAGE

710.4.1 /CØNPARS/,/KERRØR/,/KQBUFP/,/KQRNDM/ - ATLAS system common blocks, described in section 100.2.

710.4.2 /CØNMAT/ contains control and problem size information for use by the MASS module.

710.4.3 /TETCØM/ passes control information between FUELTET and TETDRIV.

/CØØR20/ has the coordinate points for a 20 node BRICK.

/CØRNER/ has the corner points for a tetrahedron.

/TABCØM/ has control information for use in FUELTAB and TABDRIV.

/ELMCØM/ passes control information between FUELELM and FCØNTET.

/PELMCØM/ passes control information between PAYELM and PCØNPAS, PCØNTET.

/PBALCØM/ contains balance data and control information.

/PBGPCØM/ passes control information between PAYBAL and PBALGP, PBALPAS.

NAME	ØVERLAY: MASSØLF (1,0)	(1,1)	(1,2)	(1,4)	(1,5)	(1,6)	(1,7)	(1,10)
CØNPARS	x							
KERRØR	x	x	x	x	x	x	x	
KQBUFP	x	x	x	x	x	x	x	x
KQRNDM	x	x	x	x	x	x	x	x
CØNMAT	x	x	x	x	x	x	x	x

NAME	ØVERLAY: MASSØLF (1,12)	(1,13)	(1,14)	(1,15)	(1,16)	(1,17)	(1,20)
KERRØR	x	x	x	x	x	x	x
KQBUFP	x						
KQRNDM	x	x	x	x	x	x	x
CØNMAT	x	x	x	x	x	x	x
TETCØM		x				x	
CØØR20		x				x	
CØRNER		x				x	
TABCØM			x				
ELMCØM					x		

NAME	ØVERLAY: MASSØLF (1,21)	(1,22)	(1,23)	(1,24)	(1,77)
CØNPARS					x
KERRØR	x	x	x	x	x
KQBUFP					x
KQRNDM	x	x	x	x	x
CØNMAT	x	x	x	x	x
PELMCØM		x			
PBALCØM				x	
PBGPCØM				x	

711. MERGE PROCESSOR

711.1 GENERAL INFORMATION

711.1.1 Purpose

Merge element stiffness, mass, nodal loads and displacement matrices for regular and substructured models. Produce tables of node and freedom activities.

711.1.2 Access

This processor is called from the ATLAS control program by the EXECUTE MERGE(plist) statement (sec. 242, ref. 1-1).

711.2 MATRIX ACTIVITY

711.2.1 Set/Stage Mode

	Name	Description	File
Input			
1	KACV0ba	Assembly Control Vector	DATARNF
2	KFAV01a	Stiffness Freedom Activity Vector	STIFRNF
3	KRFV0ba	Retained Freedom Vector	DATARNF
4	KUFRT0a	User Freedom Reference Table	DATARNF
5	KFAT00a	Freedom Assignment Table	MERGRNF
6	KNC100a	Nodal Correspondence Table	DATARNF
7	LFAV0ba	Loads Freedom Activity Vector	LØADRNF
8	MFAV00a	Mass Freedom Activity Vector	MASSRNF
9	DCØØRba	Loadcase Correspondence Table	LØADRNF
10	KA0001a	Element Stiffness Matrix	STIFRNF
11	MA0001a	Element Mass Matrix	MASSRNF
12	Cg0001a	Element Concentrated Mass Matrix	MASSRNF
13	DA001ba	Nodal Displacement Matrix	LØADRNF
14	LA001ba	Nodal Loads Matrix	LØADRNF
15	KNØDCØN	Nodal Connectivity	DATARNF
16	BSETCØN	Bset Control	STIFRNF

	Name	Description	File
Output			
1	Kij b a or XXXXXXX	i-j Partition of Stiffness Matrix	MERGRNF
2	Mij b a or XXXXXXX	i-j Partition of Mass Matrix	MERGRNF
3	Li1 or XXXXXXX	i-1 Partition of Loads Matrix	MERGRNF
4	D31	3-1 Partition of Displacement Matrix	MERGRNF
5	KRTC0ba	Retained Freedom Correspondence Table	MERGRNF
6	KUFRT0a	User Freedom Reference Table	MERGRNF
7	KFAT00a	Freedom Assignment Table	MERGRNF

711.2.2 Substructure Mode

Input			
1	IACVsss	Assembly Control Vector	DATARNF
2	IRFVsss	Retained Freedom Vector	DATARNF
3	IRTCsss	Retained Freedom Correspondence Table	DATARNF
4	IFATsss	Freedom Assignment Table	DATARNF
5	IFAVsss	Stiffness Freedom Activity Vector	DATARNF
6	IUFRsss	User Freedom Reference Table	DATARNF
7	INC1sss	Nodal Correspondence Table	DATARNF
8	ILFAsss	Load Freedom Activity Vector	DATARNF
9	ILCØsss	Loadcase Correspondence Table	DATARNF
10	ILRCsss	Loads Runcodes Matrix	DATARNF
11	ISRCsss	Stiffness Runcodes Matrix	DATARNF
12	INDMsss	Nodal Data, Matrix	DATARNF
13	IDLCsss	Downward Loadcase Runcodes Matrix	DATARNF
14	IELCsss	Loadcase Expansion Runcode Matrix	DATARNF
15	LA001ba	Nodal Loads Matrix	LØADRNF
16	DA001ba	Nodal Displacement Matrix	LØADRNF

User Matrices

17	KREDsss	Reduced Stiffness Matrix
18	MREDsss	Reduced Mass Matrix
19	LREDsss	Reduced Loads Matrix
20	DBS1sss	Free Partition Substructure Displacement Matrix
21	DBS2sss	Retained Partition Substructure Displacement Matrix
22	DBS3sss	Support Partition Substructure Displacement Matrix
23	GREDsss	Reduced Geometric Stiffness Matrix

Output

1	KijSsss	i-j Partition of Substructure Stiffness Matrix	MERGRNF
2	MijSsss	i-j Partition of Substructure Mass Matrix	MERGRNF
3	Li1Ssss	i-1 Partition of Substructure Loads Matrix	MERGRNF
4	D31Ssss	3-1 Partition of Substructure Displacement Matrix	MERGRNF
5	DBS2sss	Retained Partition Substructure Displacement Matrix	MERGRNF
6	IRTCsss	Retained Freedom Correspondence Table	MERGRNF
7	IFATsss	Freedom Assignment Table	MERGRNF
8	IUFRsss	User Freedom Reference Table	MERGRNF
9	GijSsss	i-j Partition of Sub-structure Geometric Stiffness Matrix	MERGRNF

The format of the user matrices is described in section 400.3. The format of the rest of the matrices is described in reference 1-2.

711.3 PROGRAM METHOD

The MERGE module consists of a primary overlay, MERGE, and five secondary overlays, MRGSET, MRGMAP, TØURSRT, PØLYMRG and LØDER.

711.3.1 ØVERLAY(MERGE,1,0)-MERGE

a. Establishes processor's file environment

b. Controls execution through secondary overlays.

c. Writes summary printout for each overlay if checkout flag is on.

d. Handles error returns and error printing.

711.3.2 ØVERLAY(MERGØLF,1,1)-MRGSET

Control parameter interpretation and generation

a. CØNPARS(3,1) is read for basic merge option.

b. Rest of /CØNPARS/ is examined for control parameters and a matrix list.

c. Parameters not specified are set to default.

d. List of common matrix names is constructed.

e. List of output matrices is set up.

f. If not already in existence, the freedom assignment table (KFAT) is constructed from the assembly control vector (KACV) and stiffness freedom activity vector (KFAV).

g. List of free freedoms is printed out by user node number and freedom index.

h. If requested, list of inactive retained freedoms is printed out by user node number and freedom index.

i. For set/stage mass merge, compare stiffness and mass freedom activity vectors and indicate deviations.

j. Produce retained freedom correspondence table (KRTC) from retained freedom vector (KRFV) and freedom assignment table (KFAT).

k. Update user freedom reference table (KUFRT) with final partition population counts.

l. For loads merge, obtain number of loadcases in problem.

m. For loads merge, compare stiffness and loads freedom activity vectors (LFAV) and indicate deviations.

n. Set up matrix merging information and print a message from matrices of zero order.

o. Check to see if a requested matrix already exists and print the appropriate message (see ØPTIØN description, sec. 242, ref. 1-1).

711.3.3 ØVERLAY(MERGØLF,1,2)-MRGMAP

This overlay reads bulk matrix data and assigns sort keys to each datum. The main program controls the execution of the overlay according to one of the ten merge options:

a. Read matrix, produce sort key elements for:

1.	Stiffness	Set/Stage
2.	Mass	Set/Stage
3.	Loads	Set/Stage
4.	Displacements	Set/Stage
5.	Stiffness	Substructure
6.	Mass	Substructure
7.	Loads	Substructure
8.	Displacements	Substructure
9.	Gstiffness	Set/Stage
10.	Gstiffness	Substructure

b. Decompose a higher level substructure displacement matrix into its component substructure matrices.

c. Transform a substructure displacement matrix from one loadcase system to another.

Convert sort key elements into a sort key and add datum to file to be sorted. Each type of matrix is handled by a separate routine.

Each block of sort keys and datums is sorted and an in-place merge is performed before the block is written out.

711.3.4 ØVERLAY(MERGØLF,1,3)-STRING

In the event multiple blocks are written out by MRGMAP, STRING performs the string merge on these blocks. Blocks are maintained on numbered random files so that blocks may be sorted by sorting indices. Thus only overlapping blocks are ever called in for string merge.

711.3.5 ØVERLAY(MERGØLF,1,5)-LØDER

Matrix data contained on a sorted sequential file is written to the random access file, MERGRNF, in utility matrix format. If a row is missing, at least one zero will be produced for the row. For a triangular matrix, the zero is put on the diagonal; for a rectangular matrix, the zero is put in the first column. At the end of the process, zero and null matrices are inserted into the user matrix catalog.

711.3.6 ØVERLAY(MERGØLF,1,6)-KMTEST

Provides a consistency check on the global stiffness (1,1) partition by determining the eigenvalues of the submatrix for each node. The ratio of largest to smallest is used as a test and error and warning messages are printed. A second pass check may be requested which uses the larger submatrix of nodes connected to each node.

711.4 COMMON BLOCK USAGE

711.4.1 /CØNPARS/,/KERRØR/,/KQRNDM/,/KQBUFP
are ATLAS Common Blocks described in section 100.2.

711.4.2 MERGE Common Blocks

/ARRAY1/
- MFIRST - Switch indicating if first call to MAP routine
- INTEN - Parameter indicating type of matrix
  - = 1 Stiffness
  - = 2 Mass
  - = 3 Geometric Stiffness
- MYSTAT(I,J,K) - Array of matrix status codes
  - J = Matrix Row Partition Number
  - J = Matrix Column Partition Number
  - K = Matrix Storage
    - = 1 lower triangular matrix by row
    - = 3 rectangular matrix by row

/BFSIZE/
- MAXREC - Maximum record size of output matrices.

/CHKPRNT/
- ICHK1 - Switch for printing input records in octal.
- ICHK2 - Not used.

ICHK5 - Switch for printing output records in octal.

/CHKPT2/

ICHK3 - Switch for printing records of unsorted file
= 1 in octal
= 2 in floating point

ICHK4 - Switch for printing records of sorted file (same as ICHK3)

/ERRØR/

KRTRN - Contains address in main overlay for error return.

/FRMCNT/

NFRDMS - Array containing partition counts

NFRDMS(1) - Number of elements in Partition 1

NFRDMS(2) - Number of elements in Partition 2

NFRDMS(3) - Number of elements in Partition 3

NFRDMS(4) - Number of elements in Partition 4

NFRDMS(5) - Number of columns in deflection matrix

NFRDMS(6) - Number of columns in loads matrix

/ØPTIN10/

IMTXLS - Name of substructure matrix to be expanded

IPARTP - Partition number of substructure matrix to be expanded

/MGCØST/

MGCØST - Switch to print CRU information

/MRGØPT/

MRGØPT - Contains basic merge switch
=1 Set/stage elemental stiffness matrices
=2 Set/stage elemental mass matrices
=3 Set/stage nodal loads matrices
=4 Set/stage specified displacement matrices
=5 Substructure stiffness matrices
=6 Substructure mass matrices
=7 Substructure loads matrices

=8 Substructure displacement matrices
=9 Higher level substructure displacement matrix partition
=10 Substructure loadcase dependent matrix expansion

/MXFILE/
NØFILE - Number of matrices in array MXFILE
MXFILE - Array containing matrix names and codes
MXFILE(I,1) - Name of matrix I in list
MXFILE(I,2) - Row partition code of matrix I
=1 free
=2 retain
=3 support
=4 not used

MXFILE(I,3) - Column partition code of matrix I
=1 free
=2 retain
=3 support
=4 not used
=5 displacements
=6 loads

/NECØN/
DASET - Contains set number in display code equivalent in bits 23-18
ISTGE - Contains stage number in display code equivalent in bits 29-24

/NEIGH/
NAMHLD - Array containing matrix file names

/ØPTIØN9/
NUROWS(I) - Number of rows in substructure I
NUCLS(I) - Number of columns in substructure I

/PFDATA/
IDATA(1) - Number of bulk records read
IDATA(2) - Number of items in bulk records
IDATA(3) - Number of data mapped
IDATA(4) - Number of unsorted records
IDATA(5) - Number of items in unsorted records
IDATA(6) - Number of items in sorted records
IDATA(7) - Number of sorted records
IDATA(8) - Number of output matrix records
IDATA(9) - Number of items in output matrix records

/RNAMES/

NACV	-	Name of assembly control vector
NFAV	-	Name of stiffness freedom activity vector
NRFV	-	Name of retained freedom vector
NRTC	-	Name of retained freedom correspondence table
NUFR	-	Name of user freedom reference table
NFAT	-	Name of freedom assignment table
MNC1	-	Name of nodal correspondence table
NLFA	-	Name of loads freedom activity vector

/SDATA/

KCNMS	-	Contains concentrated mass data subset number in bits 53-48
KRETN	-	Retained freedom handling option = 1 Inactive retains are flagged as error = 2 Inactive retains are left in problem
KSBNØ	-	Substructure number in display code in bits 35-18
KSBTP	-	Structure type = 1 set/stage = 2 first level substructure = 3 upper level substructure
KSWEN	-	Matrix list switch = 1 no list = 2 list present

/SNAMES/

NØSUB	-	Number of names in array
ISNAME	-	Array of substructure names

/SRTFILS/

NWØRDS	-	Number of words in sort group
LF(1)	-	Input file
LF(2)	-	Sort file 1
LF(3)	-	Sort file 2
LF(4)	-	Sort file 3

	OVERLAY: MERGØLF				
NAME	(1,0)	(1,1)	(1,2)	(1,3)	(1,5)
CØNPARS		x			
KERRØR	x	x			
KQBUFP	x				
KQRNDM	x				x
ARRAY1	x	x	x		
BFSIZE	x	x			x
CHKPRNT	x	x	x		x
CHKPT2	x	x	x		x
EKRØK	x				
FRMCNT	x	x	x		x
ØPTIN10	x	x	x		
LGØFAT			x		
MGCØST	x	x			x
MKGØPT	x	x	x		x
MTXSPEC					x
MXFILE	x	x			x
MXTSPEC					x
NECØN	x	x	x		x
NEIGH	x	x			x
NULLIN			x		
ØPTIØN		x			
ØPTIØN9	x		x		x
PFDATA	x		x		x
RNAMES	x	x	x		
SDATA	x	x	x		
SDATA1		x			
SNAMES	x	x	x		x
SFLFET			x		
SRTFILS	x		x		
STRINGS	x			x	

712. MULTIPLY PROCESSOR

712.1 GENERAL INFORMATION

712.1.1 Purpose

The MULTIPLY processor evaluates the matrix expression

$$[D] = \{\pm[C]\} \pm [A(t)]_1[B]_1 \pm [A(t)]_2[B]_2 \pm \ldots$$

where [A], [B], [C] and [D] are ATLAS user matrices, and {} indicates optional terms. A series of from one to five matrix products can be processed by each call to the program. The first matrix in a product may be specified as transposed. If any one of the factor matrices is null the corresponding matrix multiplication will not be carried out.

712.1.2 Access

The processor is called from the ATLAS control program by the EXECUTE MULTIPLY(plist) statement. The matrix expression to be evaluated is specified in plist, which also may contain other optional specifications. (sec. 244, ref. 1-1).

712.2 MATRIX ACTIVITY

Input

From 2 to 11 user matrices, depending on the length of the matrix expression. Any user matrix residing in the ATLAS data base at the time of the call to the multiply processor may be used as input. Formats of the user matrices are described in section 400.3.

Output

A user matrix containing the evaluated matrix expression.

712.3 PROGRAM METHOD

The MULTIPLY processor is organized as an overlay program consisting of one primary overlay, MULTPAC, and two secondary overlays, SETPARS and MATMULT. The overlay structure, including supporting subroutines, is described in section 712.4. The internal processing of the program is divided into three subtasks which are defined as follows:

a) Matrix loading - The input matrices are re-formatted and copied onto numbered random scratch files. The primary purpose of the re-formatting is to change the record size of the matrices in order to 1) utilize all core storage available to perform the matrix operations, and 2) to achieve overlapping of CP operations and data transfer to/from disk storage. The principles used to select the matrix partitioning sizes will be discussed in section 712.3.2. A secondary effect of re-formatting is to extract information about matrix sparseness which is used to increase processing efficiency. The routines performing matrix loading are RTLØAD, TFLØAD and TBLØAD.

b) Matrix multiplication - The matrix operations specified by the EXECUTE MULTIPLY statement in the control program are carried out. The program distinguishes between five different multiplication "schemes" as described in section 712.3.1. The matrix multiplication routines are BTBMLT, BWDMLT, FWDMLT, RCRMLT and RRRMLT.

c) Matrix unloading - The result matrix is re-formatted and copied from scratch file to the random named file MULTRNF. Matrix partition size on MULTRNF is either input via the control program statement or a default value is used. The matrix unloading routines are RTBLØD and TBRLØD.

712.3.1 Matrix Multiplication Formats

A matrix multiplication scheme is defined by the types of matrices involved in a matrix product. The program distinguishes between the five multiplication schemes shown in table 712-1. In this table, [A] denotes the pre-multiply matrix, [B] the post-multiply matrix, [C] the add matrix, and [D] the result matrix. Where "triangular" is entered, the lower triangular portion of a banded, symmetrical matrix is meant.

Table 712-1. Multiplication schemes used by the MULTIPLY Processor

Multiplication Scheme	[A] Matrix	[B] Matrix	[C] Matrix	[D] Matrix	Subroutine Used
1	rectangular	rectangular	rectangular	rectangular	RRRMLT
2	rectangular transpose	rectangular	rectangular	rectangular	RCRMLT
3	[B(t)]	rectangular	triangular	triangular	BTBMLT
4	triangular	rectangular	rectangular	rectangular	FWDMLT BWDMLT
5	rectangular transpose	rectangular	triangular	triangular	RCRMLT

712.3.2 Overlapped Processing - Core Strategy

In performing the various types of matrix multiplications the program will attempt to optimize the processing in the following ways:

(a) Overlap central processor operations and data transfer to and from disk storage.

(b) Select matrix partitioning sizes such that a minimum of core work area is needed to satisfy requirement (a) above.

In the following the arrangement for each matrix multiplication scheme will be discussed separately. The following notation will be used:

t(c) - The Central Processor time required to perform d=c(t)a*b, where a, b, c and d are all floating point numbers.

t(a) - Disk access time.

t(t) - The time required to transfer one word from disk to core.

T(c) - The Central Processor time required to perform a matrix multiplication with one set of [A], [B] and [C] partitions.

T(t) - The transfer time required to copy one partition from disk to core.

W - The size of the core work area.

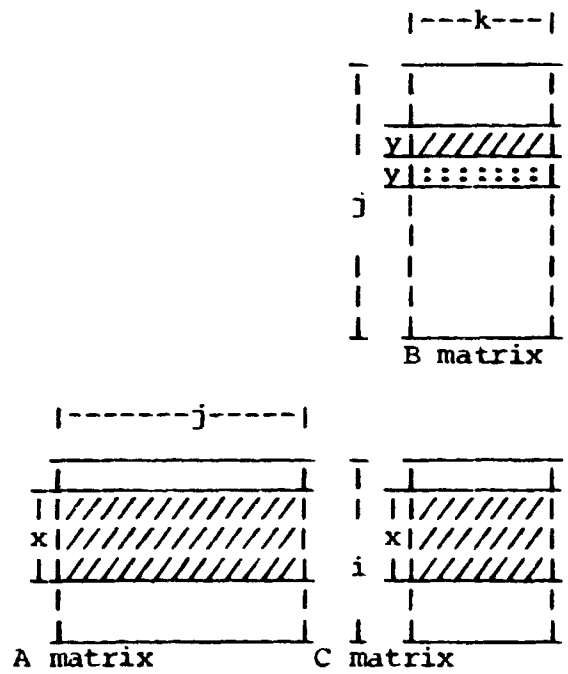
i,j,k - Dimension parameters of the [A], [B] and [C] matrices such that [A] = A(i,j), [B] = B(j,k), [C] = C(i,k).

|////| - Core resident matrix partitions

|::::| - Matrix partition being transferred between disk and core.

|____| - Disk resident matrix partitions

712.3.2.1 Multiplication scheme 1



The matrix multiplication is illustrated in the figure above. The core work area is organized into four buffers containing matrix partitions. Three of the buffers contain the [A], [B], and [C] partitions which are currently being multiplied. The fourth buffer is used to store the next [B] matrix partition to be

processed. This buffer is loaded while multiplication is carried out on the other partitions. The [A] and [C] partitions will contain x rows, and the two [B] partitions y rows each. Equations for W, T(c) and T(t) can then be set up as follows:

$$W = x(j+k)+2yk$$

$$T(c) = xykt(c)$$

$$T(t) = ykt(t) + t(a)$$

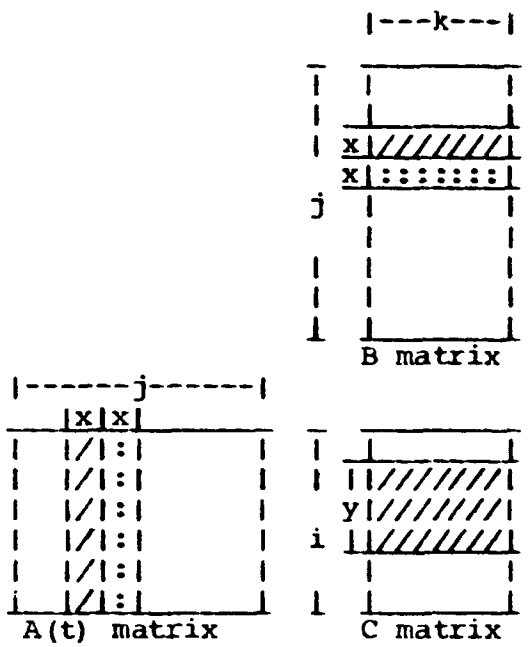
The requirement for overlapping of central processor operations and data transfer is obtained from the last two equations:

$$T(c) = T(t)\ ;\ x = \frac{t(t)}{t(c)} + \frac{1}{yk}\,\frac{t(a)}{t(c)}$$

Substituting for x in the first equation and evaluating dW/dy gives the condition for minimum work area:

$$\frac{dW}{dy} = 0\ ;\ y = \frac{1}{k}\ \mathrm{SQRT}\ \frac{(t(a)\,(j+k)}{2t(c)}$$

712.3.2.2 Multiplication scheme 2



The work area in this case contains five matrix partitions: two [A] and [B] partitions and one [C] partition. The two extra [A] and [B] buffers are loaded while the other partitions are multiplied. The [A] and [B] partitions contain x rows, the [C] partition y rows. In setting up the transfer time for the [A] and [B] partitions it is assumed that the corresponding files were accessed through different channels so that the loading may occur simultaneously. The equations for W, T(c) and T(t) will be:

$$W = 2x(i+k)+yk$$

$$T(c) = yxkt(c)$$

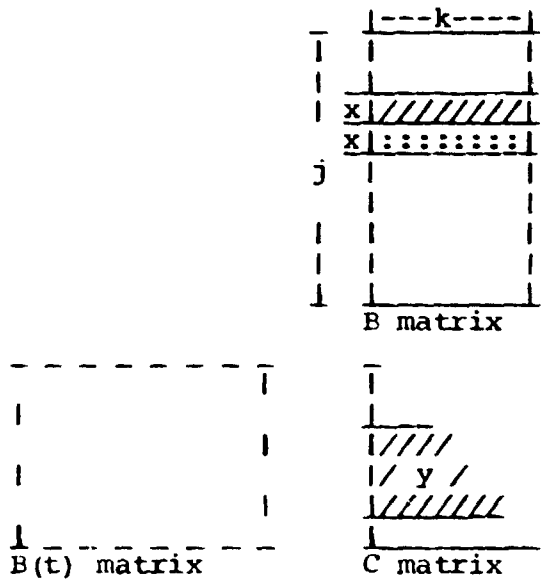
$$T(t) = xit(t)+t(a)$$

The optimal buffer sizes are then found to be:

$$T(c)=T(t) : y=\frac{i}{k}\frac{t(t)}{t(c)} + \frac{1}{xk}\frac{t(a)}{t(c)}$$

$$\frac{dW}{dx}=0 : x=SQRT\left(\frac{t(a)}{2t(c)(i+k)}\right)$$

712.3.2.3 Multiplication scheme 3



The work area contains three matrix partitions; two [B] partitions and one [C] partition. One [B] partion is transferred to core while the other partitions are multiplied. The B partitions contain x rows and the C partition y matrix elements. The equations for W, T(c), T(t), x and y are:

$$W = 2xk+y$$

$$T(c) = xyt(c)$$

$$T(t) = xkt(t) + t(a)$$

$$T(c) = T(t);\ y=k\frac{t(t)}{t(c)} + \frac{1}{x}\frac{t(a)}{t(c)}$$

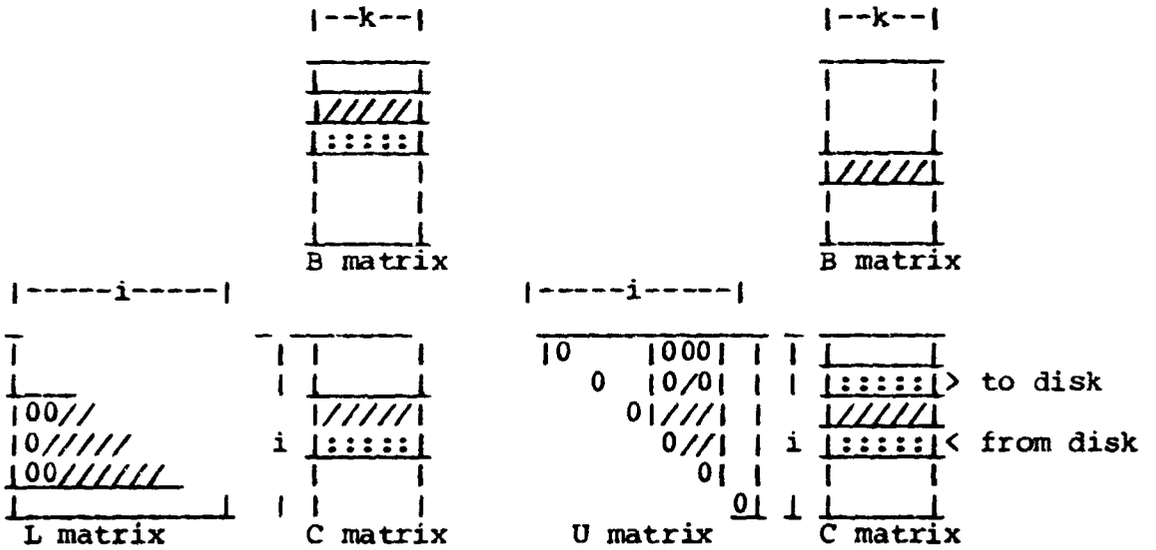
$$\frac{dW}{dx} = 0;\ x = \mathrm{SQRT}\ \left(\frac{1}{2k}\frac{t(a)}{t(c)}\right)$$

712.3.2.4 Multiplication scheme 4

This matrix product is evaluated as

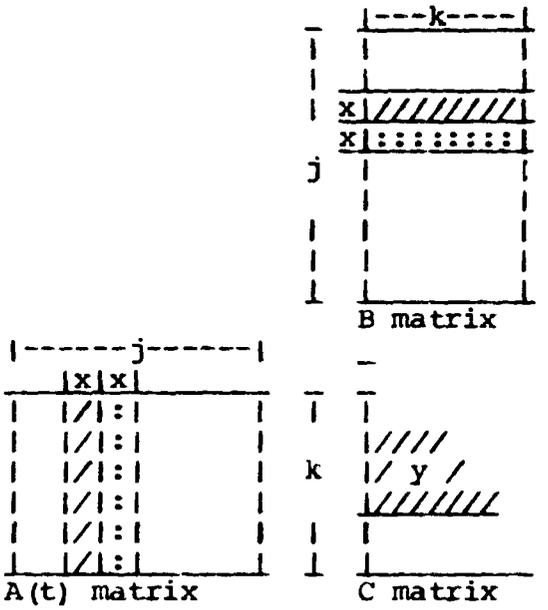
$$[D] = \pm[C] \pm ([L]+[U]) * [B]$$

where L is the lower triangular half of a banded symmetric matrix, and [U]=[L(t)] without the diagonal terms of [L]. The matrix product is illustrated in the figures below.



Because of the variable bandwidth of the [L] and [U] matrices, a partitioning of the matrices which allows full overlapping becomes impractical. In both of the products described above the available core area is divided into five equal partitions. Overlapped processing is employed as fully as possible.

712.3.2.5 Multiplication scheme 5



The same algorithm as for multiplication scheme 2 is used, except that in this case the [C] matrix is lower triangular. The [C] matrix partition contains y elements. The equations for optimal core usage are:

$$W = 4xk+y$$

$$T(c) = xyt(c)$$

$$T(t) = xkt(t) + t(a)$$

$$T(c) = T(t);\ y=k\frac{t(t)}{t(a)} + \frac{1}{x}\frac{t(a)}{t(c)}$$

$$\frac{dW}{dx} = 0;\quad x = \frac{1}{2}\ \mathrm{SQRT}\ \frac{(t(a))}{kt(c)}$$

712.3.3 Scratch file usage

The MULTIPLY Processor uses the following numbered random files for internal data storage: MULTF1, MULTF2, MULTF3, MULTF4, TABFL1 and TABFL2. The files are accessed by the I/Ø routines REDER and WRTER. The contents of the files are illustrated below. The storage formats referenced in this table are described subsequently.

Multiplication Scheme	MULTF1	MULTF2	MULTF3	MULTF4	TABFL1	TABFL2
1	A Matrix Format 1	C Matrix Format 1	B Matrix Format 1		Partition Sizes Format 4	
2	A Matrix Format 1	B Matrix Format 1	C Matrix Format 1		Partition Sizes Format 4	
3	B Matrix Format 1	C Matrix Format 2			Partition Sizes Format 4	
4	C Matrix Records 1,2,5,6,9, 10,... Format 1	C Matrix Records 3,4,7,8, ... Format 1	B Matrix Format 1	A Matrix Format 3	Partition Sizes Format 4	Additional A Matrix info. Format 5
5	A Matrix Format 1	B Matrix Format 1	C Matrix Format 2		Partition Sizes Format 4	

712.3.3.1 Storage format 1

This format is used for rectangular matrices stored by rows. Each file record contains a partition of the matrix consisting of full matrix rows.

Word	Contents
1	Record number
2	Record length
3	Pointer to first matrix element of record
4	Row number of first matrix row in record
5	Number of matrix rows in record
6	Column number of first column in record containing a nonzero element
7	Distance between first and last column in record containing nonzero elements
8-16	Not used
17-n	Matrix elements stored sequentially by rows
n+1	Integer checksum of elements 1 through n.

712.3.3.2 Storage format 2

This format is used for full (non-banded) symmetric matrices which are stored lower triangular by rows. Each row is fully contained in one record.

Word	Contents
1	Record number
2	Record length
3	Pointer to first matrix element in record
4	Row number of first row in record
5	Number of matrix rows contained in record
6-16	Not used
17-n	Matrix elements stored sequentially by rows
n+1	Integer checksum of elements 1 through n

712.3.3.3 Storage format 3

This format is used for banded symmetric matrices. The matrix is stored lower triangular by rows. Each row is fully contained in one record. In the description below references are to the [B] and [C] matrices of multiplication format 4.

Word	Contents
1	Record number
2	Record length

3	Pointer to first matrix element in record
4	Row number of first row in record
5	Number of matrix rows contained in record
6-7	Not used
8	Let NCØL designate the left-most nonzero column in the record. This word is a pointer to the [B] matrix record which contains row number NCØL.
9	Let NRØW designate the number of the first matrix row contained in the next record. This word is a pointer to the [C] matrix record which contains row number NRØW.
10	Let LRØW designate the last matrix row in the record. This word is a pointer to the [C] matrix record which contains row number LRØW.
11	Not used
12	Same as word 9
13	Same as word 10
14	Same as word 8
15-16	Not used
17-n	Matrix elements stored sequentially by rows. Each row is stored left to right, starting with the first nonzero element and ending with the diagonal element.
n+1	Integer checksum of elements 1 through n.

712.3.3.4 Storage format 4

This format consists of three records with the following contents:

Record	Contents
1	Array containing the lengths of each [A] matrix record.
2	Array containing the lengths of each [B] matrix record.
3	Array containing the lengths of each [C] matrix record.

712.3.3.5 Storage format 5

This format is used while processing multiplication scheme 4. The file contains NØAREC+1 records, where NØAREC is the number of partitions of the [A] matrix.

Record	Contents
1	Array NØAREC+1 long. Word i contains word 8 of record i+1 of storage format 3.

	Word NØAREC is zero, word NØAREC+1 contains the integer checksum of words 1 through NØAREC.
j (2≤j≤NØAREC+1)	Each record j corresponds to record j-1 of storage format 3. The record is NRØW+1 words long, where NRØW is the number of matrix rows stored in record j-1 of format 3. Each element contains the position (column number) of the first nonzero element in the corresponding matrix row. Word NRØW+1 contains the integer checksum of the record elements.

712.3.4 Overlay Description

712.3.4.1 ØVERLAY (MULTØLF,1,0) - MULTPAC

The only task performed by this code is to call the two secondary overlays.

712.3.4.2 ØVERLAY (MULTØLF,1,1) - SETPARS

This overlay performs the following functions:

a. Decode and interpret the execution control parameters associated with the program call. Internal control information parameters are set. A thorough checkout is performed on the matrix expression to be evaluated.

b. Internal matrix buffer sizes are computed. The program attempts to allocate the optimal buffer sizes developed in section 712.3.2. If the actual core work area is too small to satisfy this requirement, the best possible buffer arrangement is adopted.

712.3.4.3 ØVERLAY (MULTØLF,1,2) - MATMULT

This overlay is entered twice during the execution of the MULTIPLY processor. The task in (a) below is performed during the first entry; the tasks described in (b) and (c) during the second entry.

a. The length of the core work area is computed and saved for use in overlay program SETPARS.

b. A check is made to see if the evaluated matrix expression will result in a zero matrix. In this case a null result matrix is generated and further processing in the module aborted.

c. Matrix products are performed in the sequence specified in the matrix expression. The products will be added successively to the add matrix given in the matrix expression. If no add matrix is specified, a zero add matrix will be generated before multiplication takes place. Each matrix product is preceded by the loading of the add matrix and the two product matrices, and followed by the unloading of the result.

712.4 COMMON BLOCK USAGE

712.4.1 /CØNPARS/
/KERRØR/ - ATLAS system common blocks,
/KQBUFP/ described in section 100.2.
/KQRNDM/

712.4.2 /EPAR/ - Used for the matrix loading operation. The parameters apply to the input matrix of this operation.

/EPARB/ - Used for the matrix unloading operation. The parameters apply to the output matrix of this operation.

/FILNAM/ - Contains file names of scratch files.

/IPAR/ - Used for the matrix loading operation. The parameters apply to the output matrix of this operation.

/IPARB/ - Used for the matrix unloading operation. The parameters apply to the input matrix of this operation.

/MASHTP/ - Used in the matrix loading operation. It determines whether a loaded matrix should be stored in a compressed format (zero elements deleted).

/MLTCØN/ - Contains control information mostly extracted from the ATLAS control program statement.

/MLTERR/ - Contains MULTIPLY processor error flags.

/RECPAR/ - Contains matrix parameters used to perform the actual matrix multiply operations.

/TEXT/ - Contains text for error heading.

	ØVERLAY: MULTØLF		
NAME	(1,0)	(1,1)	(1,2)
CØNPARS		x	
KERRØR	x		
KQBUFP			x
KQRNDM			x
EPAR			x
EPARB			x
FILNAM			x
IPAR			x
IPARB			x
MASHTP			x
MLTCØN	x	x	x
MLTERR	x	x	x
RECPAR			x
TEXT			x

713. RHØ3 PROCESSOR

713.1 GENERAL INFORMATION

713.1.1 Purpose

Calculate unsteady loadings caused by motions of lifting surfaces with trailing edge controls based upon the subsonic kernel function approach of reference 713-1.

The pressure singularities at hinge line and side edges are extracted analytically as a preliminary step to solving the integral equation by collocation.

The module calculates generalized aerodynamic forces for user supplied deflection modes. Optional intermediate output includes:

a. unsteady pressures at an array of points.

b. sectional generalized forces.

From one to four control surfaces on the half span can be accommodated.

713.1.2 Access

This module is called from the ATLAS control program by the EXECUTE RHØ3(plist) statement (sec. 250, ref. 1-1).

713.2 MATRIX ACTIVITY

		Name	Description	File
Input				
	1	R30i000	Input data case	DATARNF
	2	RCmi000	Input hinge rotation coefficients	DATARNF
	3	xxxxxxx	Interpolation coefficients	INTERNF
In/Out				
	1	CM00000	C-matrix index	RHØ3RNF
	2	CMi0000	C-matrix	RHØ3RNF
Output				
	1	ACMij00	Control matrix	RHØ3RNF
	2	DW0ijkl	Full downwash	RHØ3RNF
	3	DWMijkl	Moditied downwash	RHØ3RNF

	Name	Description	File
4	GF0ijkl	Generalized air forces	RHØ3RNF
5	HCmij00	Cubic hinge rotation coefficients	RHØ3RNF
6	MØ0ij00	Interpolated mode shapes	RHØ3RNF
7	PR0ijkl	Unsteady pressure report	RHØ3RNF
8	PS0ijkl	Pressure series coefficients	RHØ3RNF
9	R30ij00	Case and condition data	RHØ3RNF
10	SFmijkl	Sectional generalized forces	RHØ3RNF

A detailed description of the matrix formats is contained in reference 1-2.

713.3 PROGRAM METHOD

The RHØ3 module consists of a primary overlay, RHØIII, and thirteen secondary overlays:

INPREP,MIPREP,DWPREP,PRSPREP,SGFPREP,GFPREP,RDWRTC, CMCALC,PCØEFF,PRESURE,SGFØRCE,GFØRCE,ERRØR

713.3.1 ØVERLAY (RHØ3ØLF,1,0) - RHØIII

RHØIII defines the local labeled common blocks and calls the secondary overlays in the sequence required to complete the task(s) defined by the card input data case and the execution parameters.

Note that secondary overlays 1 through 6 are Mach number and K-value independent and are called only once.

713.3.2 ØVERLAY (RHØ3ØLF,1,1) - INPREP

INPREP assembles all case and condition data to direct the RHØ3 analysis:

a. Checks execution parameters for a case number.

b. Reads the designated data case, R30i000, from DATARNF.

c. Interprets and stores the execution parameters.

d. If vibration mode shapes were specified, reads them from INTERNF (sec. 150, ref. 1-1), checks the number of modes, and saves the array(s) on the internal scratch file IFSFILE (SC01RNF).

e. Alternatively, if rigid body mode shapes were specified, INPREP generates an array similar to the INTERPØLATIØN motion point coefficients and saves it on IFSFILE.

713.3.3 ØVERLAY (RHØ3ØLF,1,2) - MIPREP

Calculates control surface rotations for each control surface.

a. Call AINTL, using the interpolation coefficient array(s) read from IFSFILE, to calculate the streamwise slopes of the control surface modes at four equally spaced points on the hinge line.

b. Repeat a. for the same four points to find the slopes of the modes on the surface to which the control surface is attached (control surfaces may be stacked).

c. Calculate the angles, θ, between the four pairs of slopes (optionally modified by any user supplied velocity profile) and then find the cubic coefficients of the V curve passing through the four points. The curve defines the rotations along the hinge line.

d. Insert any hinge line rotations specified by card input data (optionally modified by any user supplied velocity profile).

e. Write the hinge rotation coefficient matrix, HCmij00, on RHØ3RNF and IFSFILE.

f. Write the case and condition data array, R30ij00, on RHØ3RNF.

g. Write the control matrix, ACMij00, on RHØ3RNF.

713.3.4 ØVERLAY (RHØ3ØLF,1,3) - DWPREP

a. Read the main surface interpolation coefficient array from IFSFILE and call AINTL to calculate the modal deflections and slopes at all downwash points on the main surface.

b. Repeat a. for each control surface. Note: the downwash points have been ordered such that they are grouped:
   - main surface points

- control surface 1 points
- control surface 2 points
- etc.

c. Optionally, modify slopes with user supplied velocity profile.

d. Write the interpolated mode shapes, slopes and deflection arrays on the file RHØ3RNF with the name MØ0ij00.

e. Write the mode shapes on the scratch file DWSFILE (SC01SSF).

713.3.5 ØVERLAY (RHØ3ØLF,1,4) - PRSPREP

If an unsteady pressure report was requested PRSPREP generates the preliminary information (independent of Mach number and K-values).

a. Write the following arrays on the scratch file DWSFILE (SC01SSF):

1. Y-BAR - values of pressure report chords

2. X-BAR - values of pressure report points

Perform steps b. and c. for each pressure report chord.

b. Calculate assumed main surface pressure modes and write on DWSFILE.

c. Calculate miscellaneous control surface information and the modified hinge line rotations for each control surface, and write on DWSFILE.

713.3.6 ØVERLAY (RHØ3ØLF,1,5) - SGFPREP

If sectional generalized forces (SGF) were requested, SGFPREP prepares preliminary information independent of Mach number and K-value.

a. Write the Y-HAT values of SGF report chords on the scratch file DWSFILE (SC01SSF). Perform steps b. and c. for each chord.

b. For the assumed main surface pressure contribution, determine the chordwise integration grid, calculate the assumed main surface modes at the integration points, and combine with the specified quadrature

weights; calculate the modal deflections at integration points and combine with the weighted assumed pressure modes to form the chordwise integrals on a termwise basis, and save the termwise integrals on DWSFILE.

For each control surface;

c. Determine condition-independent control surface pressure terms and chordwise integration grid, calculate the modal deflections at integration points and combine with the specified quadrature weights, calculate the control surface rotation at the particular sectional chord in each mode, and save control surface information, integration points, modified hinge rotations, and weighted deflections at integration points on DWSFILE (SC01SSF) (see ref. 713-1).

713.3.7 ØVERLAY (RHØ3ØLF,1,6) - GFPREP

GFPREP prepares a file of Mach number- and K-value-independent data for later calculation of generalized airforce matrices.

For the main surface assumed pressure contribution,

a. Determine the spanwise integration grid. Perform steps b. through d. for each chord.

b. Calculate the spanwise pressure terms and chordwise integration grid.

c. For each integration point, determine the chordwise pressure terms, form the assumed main surface modes, and combine with the specified quadrature weights.

d. For each integration point, calculate the modal deflection at integration points along the chord and accumulate the weighted pressure - modal deflection products.

e. Write the matrix of termwise surface integrals on the scratch file DWSFILE (SC01SSF).

For each control surface:

f. Determine the spanwise integration grid. Perform steps g. through j. for each chord.

g. Determine the control surface pressure terms and chordwise integration grid. Steps h through j are repeated for each integration point.

h. Calculate the modal deflections at integration points and combine with specified quadrature weights.

i. Calculate the control surface rotation at the integration chord in each mode.

j. Write control surface information, integration points, modified hinge rotation, and weighted modal deflections on DWSFILE (SC01SSF).

713.3.8 ØVERLAY (RHØ3ØLF,1,7) - RDWRTC

Depending upon the value of the two variables ITHMAT and INDCM, set in the (1,0) overlay, RDWRTC does the following:

INDCM Control surface indicator
= 0 - Main surface
= j - Control surface j

ITHMAT = 0 - Read the C-matrix
= i - Read the C-matrix with the name CMi0000 from RHØ3RNF and save on the scratch file CMSFILE (SC02SSF) by rows.

= -i - Read a C-matrix from the scratch file CMSFILE by rows and then write it on RHØ3RNF with the record name CMi0000.

Also write the C-matrix index array, CM00000, on RHØ3RNF.

713.3.9 ØVERLAY (RHØ3ØLF,1,10) - CMCALC

CMCALC calculates a C-matrix associated with an assumed main surface or control surface pressure modes for a given planform, downwash point distribution, K-value, and Mach number. For each downwash chord point:

a. Calculate terms in the downwash integral expression which are dependent only on the downwash chord and point.

b. Determine the spanwise integration grid-integration region endpoints, and quadrature. Perform steps c. through e. for each region.

c. Determine integration chord locations and associated quadrature weight.

d. Increment the row of the C-matrix by the weighted value of the spanwise integrand.

e. Write the row of the C-matrix associated with the particular downwash point on the scratch file CMSFILE (SC02SSF).

713.3.10 ØVERLAY (RHØ3ØLF,1,11) - PCØEFF

PCØEFF removes the step discontinuities from the downwash matrix due to any control surface rotation and solves the set of simultaneous equations for coefficients of the assumed main surface pressure modes.

PCØEFF writes the following arrays on the file RHØ3RNF.

DW0ijkl - full downwash matrix
DWMijkl - modified downwash matrix
PS0ijkl - pressure series coefficients

PCØEFF also writes on the scratch file CØFFILE (SC02SSF) the coefficients of assumed major surface pressure modes to be read by PRESURE.

713.3.11 ØVERLAY (RHØ3ØLF,1,12) - PRESURE

If an unsteady pressure report was requested PRESURE calculates the pressures based upon the data generated by PRSPREP and PCØEFF.

For each main surface point, the main surface contribution to the pressure is computed using a matrix of assumed pressure modes read from the scratch file DWSFILE (SC01SSF) and the pressure coefficients read from CØFFILE (SC02SSF).

The previously calculated control surface information is read from DWSFILE (SC01SSF), the control surface contribution is calculated and added to that of the main surface.

The pressure report matrix, PR0ijkl, is written on RHØ3RNF.

713.3.12 ØVERLAY (RHØ3ØLF,1,13) - SGFØRCE

Calculate sectional generalized forces (SGF) for a K-value and Mach number.

For each chord:

a. Calculate the main surface contribution to the total integral by reading from DWSFILE (SC01SSF) the matrix of chordwise integrals generated in SGFPREP, and combining with the pressure coefficients read from CØFFILE.

b. Read the precalculated information from DWSFILE.

c. Calculate the control surface pressure at the quadrature point distribution and multiply by the weighted deflections to generate the control surface contributions to SGF on the chord. Repeat steps b. and c. for each control surface.

d. Write the SGF matrix, SFmijkl, on the file RHØ3RNF.

713.3.13 ØVERLAY (RHØ3ØLF,1,14) - GFØRCE

GFØRCE calculates the generalized force matrix for a K-value and Mach number.

For each main surface integration chord:

a. Multiply the chordwise integrals calculated in GFPREP by the various weighted modal deflections.

b. Combine the results from a. with the pressure coefficients read from CØFFILE (SC02SSF), and add to the general airforce matrix.

For each control surface integration chord

c. Read the previously calculated information from DWSFILE (SC01SSF).

d. Calculate the control surface pressure at each quadrature point, multiply by the weighted deflections and add to the generalized airforce matrix.

e. Write the generalized airforce matrix, GF0ijkl, on the file RHØ3RNF.

713.3.14 ØVERLAY (RHØ3ØLF,1,15) - ERRØR

If an error is detected by any RHØ3 module (except INPREP) the overlay ERRØR is called to print a diagnostic message before control is returned to the ATLAS control program. The error types include I/Ø errors, SNARK errors, AINTL errors, (see description of AINTL, sec. 900), and errors resulting from singular or near singular matrices.

713.4 COMMON BLOCK USAGE

713.4.1 /CØNPARS/,/KQBUFP/,/KERRØR/, and /KQRNDM/ are ATLAS system common blocks, described in section 100.2.

713.4.2 RHØ3 Common blocks:

/BASIC/ contains the symmetry option and numeric constants such as (0.,0.), pi, pi/2, semi-span, Mach number, K-value, etc.

/CØND/ contains the arrays of K-values and Mach numbers.

/CØUNT/ contains integer variables giving the sizes of RHØ3 arrays or pointers to the current Mach number and K-values.

/CQUAD/ contains information pertinent to chordwise integration of the pressure kernel function.

/CSGEØM/ contains geometry data for the control surfaces.

/CSINFØ/ contains miscellaneous calculated control surface data.

/CSTERM/ contains constants required for control surface pressure calculations.

/CSVALU/ contains variables associated with the chords used during calculation of unsteady pressures, sectional generalized forces, and generalized forces.

/DWCHD/ contains variables associated with a downwash point: geometry, chordwise pressure terms and their derivatives, and the row of the C-matrix for the point.

/ENDIT/ contains variables used to print diagnostics in the overlay ERRØR (error number, error location, etc.)

/FILES/ contains the names of all files used inside RHØ3.

/HEDMX1/ contains the list of variables used as header information for a MATRIX1 formatted array.

/INTCHD/ contains chord integration data.

/KRNCØM/ contains variables used for kernel function calculations.

/MSGEØM/ contains geometry data for the main surface.

/ØPTIØNS/ contains the options specified by card input data or execution parameters.

/QUADWTS/ contains quadrature weighting factors used by RHØ3 in the integration scheme.

/TABLE/ contains the C-matrix index used to identify C-matrices and the data case/condition they represent.

NAME	OVERLAY: RHØ3ØLF						
	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)
/CØNPARS/	x	x					
/KERRØR/	x	x					
/KQBUFP/		x					
/KQRNDM/	x	x	x				
/BASIC/	x	x	x	x	x	x	x
/CØND/	x	x	x				
/CØUNT/	x	x	x	x	x	x	x
/CQUAD/							
/CSGEØM/	x	x	x	x	x	x	x
/CSINFØ/							
/CSTERM/	x						
/CSVALU/	x						
/DWCHD/							
/ENDIT/	x	x	x	x	x	x	x
/FILES/	x	x	x	x	x	x	x
/HEDMX1/	x	x	x				
/INTCHD/							
/KRNCØM/							
/MSGEØM/	x	x	x	x			
/ØPTIØNS/	x	x	x	x	x	x	
/QUADWTS/	x					x	
/RØ3MØD/	x	x	x	x	x		
/TABLE/	x	x	x	x	x		
	(1,7)	(1,10)	(1,11)	(1,12)	(1,13)	(1,14)	(1,15)
/BASIC/	x	x	x	x	x	x	
/CØND/							
/CØUNT/	x	x	x	x	x	x	x
/CQUAD/		x					
/CSGEØM/	x	x	x	x	x	x	
/CSINFØ/		x					
/CSTERM/		x			x	x	
/CSVALU/				x	x	x	
/DWCHD/		x					
/ENDIT/	x	x	x	x	x	x	x
/FILES/	x	x	x	x	x	x	x
/HEDMX1/							
/INTCHD/		x					
/KRNCØM/		x					
/MSGEØM/		x	x				
/ØPTIØNS/		x	x	x	x	x	
/QUADWTS/		x					
/RØ3MØD/							
/TABLE/			x	x	x	x	

714. STIFFNESS PROCESSOR

714.1 GENERAL INFORMATION

714.1.1 Purpose

1. Generate the element stiffness matrices
2. Generate the element stress matrices
3. Generate the element geometric stiffness matrices
4. Generate the freedom activity matrix, which indicates the freedoms that have structural or geometric stiffness.

714.1.2 Access

This module is called from the ATLAS Control program by the EXECUTE STIFFNESS(plist) statement (sec. 252, ref. 1-1)

714.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	KPARMS1	Data set parameters	DATARNF
2	KNØALTa	Nodal data	DATARNF
3	KNC100a	Nodal correspondence table	DATARNF
4	KLØCØØa	Local coordinate systems	DATARNF
5	KSF001a	Element data	DATARNF
6	KM00001	Material data	DATARNF
7	ST001ba	Element stresses	STRERNF**
8	KELEKEY	Element keys	DATARNF**
9	KCMSUMM	Composite material data	DATARNF
10	DCØØRba	Load case correspondence label	STRERNF**
Output			
1	KA0001a	Element stiffness	STIFRNF
2	GP0001a	Element stress	STIFRNF
3	KG0001s	Element geometric stiffness	STIFRNF
4	KFAV01a	Freedom activity	STIFRNF
5	AA0001a	Element stiffness transformation	SC01RNF*
6	AG0001a	Element geometric stiffness transformation	SC01RNF*
7	LK0001a	Element local stiffness	SC01RNF*
8	LS001a	Element local stress	SC01RNF*

9	IFAVsss	Interact freedom activity (in certain cases)	STIFRNF
10	GFAV0ls	Geometric freedom activity	STIFRNF
11	KCMSUMM	Composite material data	DATARNF
12	DCØØRba	Load case correspondence label	STRERNF**

A detailed description of the matrix format is contained in reference 1-2.

* These matrices are written on STIFRNF only if the checkout flag is ≥3.

**These matrices are used when element geometric stiffnesses are generated.

714.3 PROGRAM METHOD

The STIFFNESS module consists of a primary overlay, STIFLEX, and seven secondary overlays, SETPARS, KGEN, PGEN, AGEN, GTRASNK, KGGEN, AKGGEN.

714.3.1 ØVERLAY(STIFØLF,1,0)-STIFLEX

a. Establishes module common blocks (includes putting values into /MSKCØN/)

b. Opens files

c. Loops through secondary overlays

d. Writes summary of core usage and CP time (if checkout flag is on, also writes this summary after each secondary overlay).

714.3.2 ØVERLAY(STIFØLF,1,1)-SETPARS

a. Establishes the values of /LØCPARS/. The variables are first set to the defaults, then modified to comply with the parameters on the EXECUTE STIFFNESS statement, transmitted via /CØNPARS/.

b. If the data set specified does not exist, SETPARS prints an error message, increments /KERRØR/, and returns to STIFLEX.

c. If a geometric stiffness matrix generation is requested, the specification of a loadcase and the existence of corresponding stresses is checked. If missing, error

message is printed, KERRØR incremented and return to STIFLEX called.

d. It a higher order BRICK is specified, (more than 8 nodes), the buffer sizes for the output matrices are automatically set large enough to accommodate at least one such element.

714.3.3 ØVERLAY(STIFØLF,1,2)-KGEN

Element local stiffness, [k], matrix generator

a. Sets variables equal to pertinent /LØCPARS/ values, e.g., buffer sizes, names, set number.

b. Reads KPARMS1 to get problem data, e.g., number of nodes and elements.

c. Reads nodal data and KNC100x, the nodal correspondence table, and calls NCTFIX, which retrieves the proper bits from block 3 of KNC100X and stores them as a set of direct pointers to the nodal data.

d. Reads first blocks of element, and material data, establishes [k] matrix block, sets the flags, line counters, and indices, and calls GENK to generate the element local stiffness matrices.

e. Checks flags upon return from GENK:

flag:	purpose:
MBACK	get new material matrix
ISF1	get more element data
IKX1	filled [k] block;write it out and start another.

714.3.3.1 GENK: (FØRTRAN subroutine)

a. Processes the elements in internal order, first checking that material and element data are available for the current element. If not, sets flags as indicated in KGEN, above, and returns to KGEN.

b. With all the necessary data available, GENK transfers to the element generating SXXX routines, where XXX represents the element type (RØD,BRICK,SPAR,etc.). These, respectively, generate the [k] matrices directly into the [k] matrix block. The element routines check for space in the block.

c. Inserts the ID word (as described in ref. 1-2) at the beginning of each matrix, updates the counters and indices to reflect the additional stored information and proceeds to the next element. Note: KGEN,GENK, and the element routines use common block /STIFCØM/ to transmit flags, counters, and other needed data. For a description of /STIFCØM/ see section 714.4.

714.3.4 ØVERLAY(STIFØLF,1,3) -PGEN

Element local stress, [p], matrix generator

a. Begins with the same steps as KGEN (secs. 714.3.3 a, b, c).

b. Reads first blocks of element, and material data, establishes [k] matrix block, sets the flags, line counters, and indices, and calls GENP to generate the element local stress matrices.

c. Checks flags upon return from GENP:

flag:	purpose:
MBACK	get new material matrix
ISF1	get more element data
IPP1	filled [p] block;write it out and start another.

714.3.4.1 GENP: (FØRTRAN subroutine)

a. Processes the elements in internal order, first checking that material and element data are available for the current element. If not, sets flags as indicated in PGEN, above, and returns to PGEN.

b. With all the necessary data available, GENP transfers to the element generating PXXX routines, where XXX represents the element type (RØD,BRICK,SPAR,etc.). These, respectively, generate the [p] matrices directly into the [p] matrix block. The element routines check for space in the block.

c. Inserts the ID word (as described in ref. 1-2) at the beginning of each matrix, updates the counters and indices to reflect the additional stored information and proceeds to the next element. Note: PGEN,GENP, and the element routines use common block /STRECØM/ to transmit flags, counters, and other needed data. For a description of /STRECØM/ see section 714.4.

714.3.5 ØVERLAY(STIFØLF,1,4)-AGEN

Element local to analysis frame transformation matrix generator

a. Begins with the same steps as KGEN (sec. 714.3.3 a,b,c).

b. Checks the remaining space in core, and if there is room, reads the first stiffness element matrix, KSFXXXX, and establishes an element transformation ([A]) matrix block.

c. Processes the elements in internal order: gets element type, number of nodes, and position in KSF matrix, extracts nodes from KSF, extracts nodal geometry from KNØALTX, and calls the element matrix generating routine, AXXX, where XXX represents element type.

d. Checks flags on return from element routine:

flag:	purpose:
ID1	shows error or not enough room for [A]
KØØRD	analysis frame coordinates used

It there is not enough room, the current block is written to the appropriate file, and a new block is started.

If analysis frame kinematics are indicated, ALØCAL is called to perform the transformation. This results in the replacing of [A] with [A]', the transformation between local and analysis frame coordinate systems. ALØCAL calls GRAB, an ATLAS system routine, to produce the required transformation from the user-specified local coordinate system, stored in KLØCØØa.

e. Calls MØVAIN to generate runcodes for each column of [A]. A runcode contains the node and freedom number, each packed into thirty bits. MØVAIN also compresses out zero columns of [A].

f. Stores ID word at beginning of each matrix in [A].

714.3.6 ØVERLAY(STIFØLF,1,5) -GTRASNK

Element stress [P], stiffness [K], and geometric stiffness matrix [KG], matrix generator

a. Establishes matrix names, dimensions and I/Ø files from /LØCPARS/

b. Establishes positions in core and flags indices and counters for element [k] or [kg] and [A] matrices, and for intermediate storage of the [k][A] or [kg][A] product.

c. If [K] or [KG] required, then the [K] or [KG] block and the active freedom table are initialized together with associated indices and counters.

d. If [P] required, then the [p] block is initialized and a position in core established for [P]. Indices and counters are initialized.

e. Calls GTRAFTN to generate the required matrices. Common block /GTRACØM/ is used to communicate flags, counters and indices with GTRAFTN.

f. Checks flag KRET upon return from GTRAFTN

KRET value:	purpose:
≤ 0	Computations completed
1	[k] or [kg] matrix block exhausted
2	[p] matrix block exhausted
3	[A] matrix block exhausted
4	[P] matrix block filled
5	[K] or [KG] matrix block filled
6-10	Refers to matrix block indicated by (value -5) but to clear from core for efficient core management
11	Set up bigger [kA] or [kGA] block
12	If applicable, wite active freedom vectors

714.3.6.1 GTRAFTN: (FØRTRAN subroutine)

a. Processes the matrices in internal order, using GTRASNK to perform all I/Ø and core management.

b. Locates the [A] matrix in block and extracts matrix sizes. The word "locates" designates the reading of a new block if required.

c. If stress run only, skips to step j.

d. Locates the [k] or [kg] matrix in the block and extracts matrix parameters.

e. Checks on space in the scratch area and forms the [k][A] or [kg][A] product by call to MULTKA.

f. Checks for space in [K] or [KG] matrix block and forms the matrix product by repeated calls to VTMEI, storing directly into the block.

g. If [K] matrix generation calls ZRØWDEL to locate diagonal elements of [K] whose values are less than a given epsilon (presently 10^{-12}) times the largest element on the diagonal, the rows and columns that correspond to such elements are deleted from the [K] matrix.

h. The run-codes are taken from the [A] matrix and written into the beginning of the [K] or [KG] matrix. If the rows have been removed by step g, the runcodes are adjusted accordingly and in KFAVXXX, the active freedom vector.

i. If no stress matrices are to be generated, skip to step n.

j. Locates the [p] matrix in block and extracts the matrix sizes. Also, checks for space in [P] matrix block.

k. If [kA] matrix needed and not computed, performs steps d and e.

l. Multiplies [p][A] or [p][kA] using MAMULT.

m. Packs ID word and copies runcodes from [A] into [P].

n. Checks whether all elements are processed and if so, causes the last blocks to be written. Also causes KFAVXXX to be written, if appropriate, and also IFAVXXX, if appropriate.

714.3.7 ØVERLAY(STIFØLF,1,6) -KGGEN

Element local geometric stiffness, [kg], matrix generators.

a. Begins with the same steps as KGEN (sec. 714.3.3 a,b,c)

b. Read first partition of the matrix of stresses, establishes the [kg] matrix block and reads the first block of element data.

c. Extracts the number of stresses per element from the element key matrix.

d. Initializes counters, flags and indices and calls GENKG to generate element local geometric stiffness matrices.

e. Checks flags upon return from GENKG

flag:	purpose:
ISF1	Get more element data
IST1	Get more stresses
IKK1	Filled [kg] block; write it out and start another

714.3.7.1 GENKG: (FØRTRAN subroutine)

a. Processes the elements in internal order, first extracting nodal data and property data for current element.

b. Transfers to the element generating GSXXX routines, where XXX represents the element type (RØD, BEAM, etc.). These, respectively, generate the [kg] matrices directly into the [kg] block. The element routines check for space in block.

c. Inserts the ID word (as described in ref. 1-2) at the beginning of each matrix, updates counters and indices, checks if new element matrix partition needed and if so, causes it to be read, then checks if a new matrix of stresses is needed and if so causes it to be read. KGGEN and GENKG and the element routines use the common block /KGCØM/ to transmit flags, counters and other needed data. For a description of /KGCØM/ see section 714.4.

714.3.8 ØVERLAY(STIFØLF,1,7) -AKGGEN

Element local to analysis frame transformation matrix generator for geometric stiffnesses.

a. Begins with the same steps as KGEN (sec. 714.3.3 a,b,c,d)

b. Initiates flags, counters, etc., and calls GENAKG to generate the element transformation matrices.

c. Checks flags upon return from GENAKG

flag:	purpose:
ISF1	Get more element data
IAA1	Filled [AG] block; write it out and start another

714.3.8.1 GENAKG (FØRTRAN subroutine)

a. Processes the elements in internal order, first extracting nodal data for the current element.

b. Transfers to the element generating AKGXXX routines, where XXX represents the element type (RØD, BEAM, etc.). These, respectively, generate the [AG] matrices directly into the [AG] block. They also check for space in block and if not room, cause GENAKG to return to AKGGEN for writing of block and start a new block.

c. If analysis frame kinematics are indicated, ALØCAL is called to perform the transformation from element local coordinates to node analysis frame coordinates. This results in replacing the element routine generated matrix by one postmultiplied by the appropriate transformation matrix. ALØCATE calls GRAB, an TLAS system routine, to generate the required transformation matrices. This step is identical to the one performed by AGEN.

d. Corrects the transformation matrix for offset effects by use of subroutine ØFFTRAN.

714.4 COMMON BLOCK USAGE

Name	ØVERLAY: (STIFØLF)							
	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
CØNPARS		x						
KERRØR	x	x	x	x	x	x	x	x
KØBUFP		x	x	x	x	x	x	x
KØRNDM	x	x	x	x	x	x	x	x
CØREUSE	x	x	x	x	x	x	x	x
FLXINT	x		x	x				
GENSCRA	x		x	x	x		x	x
LØCPARS	x	x	x	x	x	x	x	x
MSKCØN	x		x	x	x	x	x	x
ØTSAKG	x		x	x				
BRICKCØ			x					
GTRACØM						x		
KADØPE					x			x
KGCØM							x	
MACØM			x	x	x	x	x	x
STIFCØM			x					
STRECØM				x				
TRANSF			x	x				
TRIARG			x	x				

714.4.1 /KERRØR/,/KQRNDM/,/KQBUFP/,/CØNPARS/, are ATLAS system common blocks, described in section 100.2.

/CØNPARS/ is filled by the ATLAS (0,0) overlay with the contents of the EXECUTE STIFFNESS statement.

/KERRØR/ contains the variables KERRØR AND KWARN, which are incremented whenever the processor encounters problems and an error message is written.

714.4.2 /LØCPARS/,/CØREUSE/, and /MSKCØN/ are used throughout the processor.

/LØCPARS/ contains parameters which control the execution of the processor. These include buffer sizes, input/output file names, and other control information. /LØCPARS/ is established by SETPARS, from defaults and the /CØNPARS/ specifications.

/CØREUSE/ transmits core usage information from the secondary overlays to STIFLEX, where it is included in a message printed when the checkout switch is on.

/MSKCØN/ establishes standard masking variables for masking 1 to 60 bits. These are initialized by STIFLEX.

714.4.3 /QTSARG/,/GENSCRA/ and /FLXINT/ are common blocks available throughout the processor but used for special purpose.

/GENSCRA/ is used as a scratch area by KGEN,PGEN, AGEN,KGGEN and AKGGEN, as well as their generating routines. By the appropriate use of EQUIVALENCE statements between locations in GENSCRA and subroutine variables, core usage can be reduced.

/QTSARG/ is used by the GPLATE routines. By positioning it next to /GENSCRA/ in core, and dimensioning it (1), the two blocks are forced to share virtually the same core.

/FLXINT/ is the communication between some element stiffness generating routines (SBEAM,SRØD,SSPAR, SSRØD) and FLFØ, a routine which evaluates certain integrals in the flexibility formulation. It is initialized by STIFLEX.

714.4.4 /STIFCØM/,/STRECØM/,/KADØPE/,/GTRACØM/,/KGCØM/, are used to transmit information between the secondary overlays and the generating or computational routine in both ways. They are local to the respective overlay.

/STIFCØM/ is used in KGEN. It contains information such as element nodal data, properties, material data, pointers and flags.

/STRECØM/ is used in PGEN. It is similar to /STIFCØM/.

/KADØPE/ is used both in AGEN and in AKGGEN. Both occurances are identical. It contains information similar to that of /STIFCØM/, with exceptions for material data, active nodes, etc.

/GTRACØM/ is used in GTRASNK. It contains information such as names, position references, locations, dimensions etc. of matrices being processed.

/KGCØM/ is used in KGGEN. It is similar to /STIFCØM/.

714.4.5 /BRICKCØ/,/MACØM/,/TRANSF/ and /TRIARG/ are common blocks associated with particular subroutines.

/MACØM/ is used to transmit matrix dimensions and skipping factors to the ATLAS general purpose matrix multiply routine, MAMULT.

/TRANSF/ and /TRIARG/ are used by the subroutine QTSHEL which is the generating routine for the GPLATE stiffness and stress matrices.

715. STRESS PROCESSOR

715.1 GENERAL INFORMATION

715.1.1 Purpose

715.1.1.1 For a Boundary Condition Stage:

1. Assemble free, retain, and support displacement matrices into one total displacement matrix.
2. Compute element stresses by multiplying element local stress matrices by displacements and then adding initial stresses computed in the LOADS module.
3. Compute element nodal forces by multiplying element stiffness matrices by displacements.

715.1.1.2 For a Superposition Stage (SUPSTAGE):

1. Generate a single matrix of displacements for all active, structural freedoms for selected SUPSTAGE components.
2. Generate element stresses either by computation from the superimposed displacements or by superimposing previously-calculated stresses for selected SUPSTAGE components.

715.1.2 Access

This module is called from the ATLAS Control program by the EXECUTE STRESS(plist) command (sec. 254, ref. 1-1).

715.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	KPARMS1	Parameter matrix	DATARNF
2	KRFV0ba	Retained freedom vector	DATARNF
3	KFAT0ba	Freedom assignment table	MERGRNF
4	KLCT00a	Element correspondence table	DATARNF
5	ISSCsss	Substructure correspondence table	DATARNF
6	ISSSCØR	Substructure control table	DATARNF

7	KSF001a	KSF matrices in internal order	DATARNF
8	GP0001a	Element local stress matrix	STIFRNF
9	ISC01ba	Initial stress control matrices	LØADRNF
10	IS001ba	Initial stress matrices	LØADRNF
11	DCØØRba	Loadcase correspondence table	LØADRNF
12	ILCLsss	Loadcase correspondence table Displacement User Matrices free displacements retain displacements support displacements	DATARNF
13	SULCTba	Superposition loadcase correspondence table	DATARNF
14	IFATsss	Freedom assignment table	MERGRNF
15	IRFVsss	Retained freedom vector	DATARNF
16	KA0001a	Element stiffness matrix	STIFRNF
17	SUPEkba	Superposition data with known or unknown factors	DATARNF
18	SUPSTGa	Stage type and unknown factors key	DATARNF
19	SUDISba	Superposition displacement constraints	DATARNF
20	SUSTRba	Superposition stress	DATARNF

	Name	Description	File
Output			
1	SLCSTba	Stress loadcase specification table	STRERNF
2	DCNTRba	Displacement control matrix	STRERNF
3	DI001ba	Displacement matrices	STRERNF
4	SCN01ba	Stress control matrices (internal order)	STRERNF
5	ST001ba	Stress matrices (internal order)	STRERNF
6	SUELCTa	Stress user element correspondence table	STRERNF
7	SELSITa	Stress element sorting index table	STRERNF
8	KECØMAa	Flexible element control matrix	STRERNF
9	KSF001a	KSF matrices in user order	STRERNF
10	US001ba	Stress matrices (user order)	STRERNF
11	USC01ba	Stress control matrices (user order)	STRERNF
12	DCØØRba	Loadcase correspondence table	STRERNF

13	FCNTRba	Force control matrix (internal order)	STRERNF
14	FØØ01ba	Force matrices (internal order)	STRERNF
15	UFC01ba	Force control matrix (user order)	STRERNF
16	UFXXXba	Force matrices (user order)	STRERNF
17	UDC01ba	Displacement control matrix (user order)	STRERNF
18	UDXXXba	Displacement matrices (user order)	STRERNF
19	SUPERba	Superposition data with input or computed factors	STRERNF

A detailed description of the matrix formats is contained in reference 1-2.

715.3 PROGRAM METHOD

The STRESS module consists of a primary overlay, STREØLF, and ten secondary overlays, SETPARS, RETSØRT, DEFLEC, LCREMØV, STRESS, UØRDER, MULDIS, SUPRFAC, SUPRDIS, and SUPRSTR.

715.3.1 ØVERLAY(STREØLF,1,0) - STREØLF

Calls secondary overlays as needed.

715.3.2 ØVERLAY(STREØLF,1,1) - SETPARS

a. Reads the parameters in the EXECUTE STRESS statement and sets up local parameters in the /STRPARS/ common block.

b. Checks for the presence of all matrices needed for input in all other overlays of the STRESS module. Stores index and file names in /STRPARS/.

c. Sets up the stress loadcase specification table when specified loadcases are requested.

715.3.3 ØVERLAY(STREØLF,1,2) - RETSØRT

a. When retained freedom displacements are present and not in internal nodal order, this overlay sorts the displacements into internal nodal order. The ATLAS

routine SHELL is used to determine the sorting order of the displacements.

b. When all sorted matrices fit in core, sorting is a simple operation of moving displacements from their old position to their new position using two routines INDSET and INSERT.

c. When core limitations prevent sorting as in b, two additional routines aid in the sorting, INFIND and ØUTFIND. Available core is fully used by INFIND controlled sorting. Matrices which do not fit in core are temporarily stored on scratch while ØUTFIND controls input/output and number moving. Local common block /FINDPAR/ passes parameters between routines.

715.3.4 ØVERLAY(STREØLF,1,3) - DEFLEC

a. Displacement partitions are assembled on a freedom by freedom basis. The freedom assignment table has each node's freedoms broken down into 4 categories - not active, free, retained, or supported. The main program does matrix input/output only while the routine DISMERG does the actual matrix assembling. The local common block /MERPARS/ transmits parameters between DISMERG and DEFLEC.

715.3.5 ØVERLAY(STREØLF,1,4) - LCREMØV

a. When stresses are not wanted for all loadcases, this overlay is called. Its function is to modify the displacement matrices by removing all loadcases not requested. The main program does matrix input/output while the routine SLCDISP does the actual matrix modifications.

b. The modified total displacement matrices and their control matrix are written on scratch.

715.3.6 ØVERLAY(STREØLF,1,5) - STRESS

a. This overlay calculates the element stresses by multiplying the local element stress matrices GP0001a from STIFRNF by the total displacement matrices. Once the multiplication is completed, initial stresses, if any, are added. The main program does all matrix I/Ø while five subroutines do all other functions.

b. The code is designed so that if all needed displacement matrices will not fit in core at one time, matrix I/Ø is minimized. This is accomplished by taking advantage of the banded properties of the element stress matrices. As soon as STRESS is done with a displacement partition, its position in core is filled by the next partition not already in core. Thus, a range of nodal displacements "moves" with the program as it calculates the stresses element by element. When a displacement matrix not in the "nodal range" is required, it is fully utilized by all elements in the current stress matrix.

c. The routine SCSETUP sets up the stress control matrix and determines an "element range" for which stresses can be calculated at the current time.

d. The routine MULTSET sets up a matrix MULTAB which contains information on how the stress and displacement matrices should be multiplied.

e. The routine MULCNTR, by using MULTAB, calls the ATLAS routine MAMULT to do the actual multiplication. It also transmits I/Ø requests for displacement matrices to the main program thru the local common block /MULPARS/.

f. The flow in the program goes through SCSETUP, MULTSET, and MULCNTR in a cyclic fashion as the "element range" moves from the first element to the last.

g. After all multiplication is completed, the routine ADDSIG0 adds in initial stresses to the results.

715.3.7 ØVERLAY(STREØLF,1,6) - UØRDER

a. Forms an element correspondence table using the local routine SMASK and the library routine ESØRT1.

b. Forms an element sorting index table using SHELL.

c. Forms KS. matrices in user element order using the library routine KSFREØR.

d. Forms user ordered stress and stress control matrices using the local routines SPLIT and GLUE and the library routine SØERD.

e. Prepares the force control matrix for sorting in the routine SPLITF.

f. Uses SØERD to sort forces and their force control matrix.

g. Reformats the control matrix in the routine GLUEF.

h. Prepares for displacement sorting in the routine DCSPLIT.

i. Uses SØERD to sort displacements and the control matrix.

j. Reformats the displacement control matrix in the routine DCGLUE.

715.3.8 ØVERLAY(STREØLF,1,7) - MULDIS

a. This overlay calculates element nodal forces by multiplying the element stiffness matrices by the displacements. The core management logic is similar to that used in the STRESS program.

b. The routine MULSET sets up the common block MULPARS and all pointers needed for the multiplication. It also sets up the force control matrix.

c. The routine MULDØ does the controlling of the multiplication by making calls to MAEXPND for expansion of a column in a lower triangular matrix and calls to MAMULT for the multiplication.

715.3.9 ØVERLAY(STREØLF,1,10) - SUPRFAC

a. This overlay updates the superposition data matrix SUPERba from file DATARNF and writes it out on file STRERNF. The main program does all matrix input/output except for constraint data which is managed by CØNSTR when unknown loadcase factors are present. Unknown factors (if any) are calculated and replaced in the SUPERba output matrix.

b. Routine SUPRREP replaces user ID with internal ID for all component loadcases in SUPERba matrix.

c. Routine CØNSTR is called when unknown factors are present in the SUPERba matrix. It generates the coefficient scratch matrix and the constraint scratch matrix which are solved in SUPRFAC to produce the calculated factors. Routine EXPAND is called by CØNSTR to return the SCLSTba matrix index for a candidate loadcase.

d. Routine FACTØRS replaces unknown factors (if any) with computed factor values in SUPERba matrix.

e. Routine REFØRMA reformats the SUPERba matrix. The component loadcases which were collected into one data block per superposition loadcase are now collected into one data block per active stage.

715.3.10 ØVERLAY(STREØLF,1,11) - SUPRDIS

a. This overlay performs superposition to form one stage of displacements. The main program does matrix input/output and two subroutines do all other functions.

b. SUPGDCM generates a displacement control matrix and a vector of number of freedoms per partition of superimposed displacements for one superposition stage. Routine SUPDISP does the actual superposition of displacements.

715.3.11 ØVERLAY(STREØLF,1,12) - SUPRSTR

a. This overlay performs superposition to form one stage of stresses. The main program reads in the superposition matrices SUPSTGa and SUPERba.

b. ADDITUP reads in the stress and loadcase data to be extracted from each component stage and generates an output stress matrix for the one superposition stage. Routine ADDØNE will add all loadcases belonging to one stage to the stresses in the output matrix.

715.4 COMMON BLOCK USAGE

	ØVERLAY: STREØLF										
NAME	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,10	1,11	1,12
/KERRØR/	x	x	x	x	x	x	x	x	x	x	x
/CØNPARS/	x	x									
/KQBUFP/	x	x	x	x	x	x	x				
/KQRNDM/	x	x	x	x	x	x	x	x	x	x	x
/STRPARS/	x	x	x	x	x	x	x		x	x	x
/FINDPAR/			x								
/MERPARS/				x							
/SLCPARS/					x						
/MULPARS/						x					
/MACØM/						x		x			
/SØRTPAR/							x				
/FILES/	x	x									
/FØRPARS/								x			
/SUPERPØ/									x		
/SUPERDI/										x	
/SUPERIM/											x

715.4.1 /KERRØR/,/CØNPARS/,/KQBUFP/,/KQRNDM/ are ATLAS system common blocks described in section 100.2.

715.4.2 /STRPARS/ is used by all overlays of the STRESS module. It contains control parameters, file names, index names, buffer sizes, and partition numbers, all set by the SETPARS overlay.

/FINDPAR/ is used to pass parameters and control information between the SNARK program RETSØRT and its two subroutines INFIND and ØUTFIND.

/MERPARS/ is used to pass parameters and control information between the SNARK program DEFLEC and its subroutine DISMERG.

/SLCPARS/ is used to pass parameters and control information between the SNARK program LCREMØV and its subroutine SLCDISP.

/MULPARS/ is used to pass parameters and control information between the SNARK program STRESS and its subroutines MULCNTR and MULTSET.

/MACØM/ passes parameters between the ATLAS routine MAMULT and the routine MULCNTR.

/SØRTPAR/ passes parameters between the ATLAS library routine SØERD and the program UØRDER which calls SØERD.

/FILES/ holds array space in common for use in calls to FILEADD.

/FØRPARS/ passes parameters between the SNARK program MULDIS and its subroutines MULSET and MILLDØ.

/SUPERPØ/ is used to pass parameters and control information between the program SUPRFAC and its subroutines.

/SUPERDI/ is used to pass parameters and control information between the program SUPRDIS and its subroutines.

/SUPERIM/ is used to pass parameters and control information between the program SUPRSTR and its subroutines.

716. VIBRATION/BUCKLING PROCESSOR

716.1 GENERAL INFORMATION

716.1.1 Purpose

1. Generate the frequency (eigenvalue) matrix.
2. Generate the mode (eigenvector) matrix.

716.1.2 Access

This module is called from the ATLAS control program by the EXECUTE VIBRATIØN(plist) or EXECUTE BUCKLING(plist) statement (sec 258 and 208, ref. 1-1).

716.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1.	xxxxxxx	Mass matrix	
2.	xxxxxxx	Stiffness or flexibility matrix	
3.	xxxxxxx	Geometric stiffness matrix	
4.	SNKddda	Nodal data subset	DATARNF
5.	KRFV0ba	Retained freedom vector	DATARNF
6.	KNØALTa	Nodal data	DATARNF
7.	KNC100a	Nodal correspondence table	DATARNF
8.	TAPLWTa	Condition summary	MASSRNF
9.	KLØCØØa	Local coordinate system	DATARNF
10.	MPARMS1	Mass module control data	DATARNF
Output			
1.	FREQSxx	Eigenvalues	VIBRRNF
2.	MØDESxx	Eigenvectors	VIBRRNF/BUCKRNF
3.	SMdddxx	Subset mode shape	VIBRRNF
4.	GMASSxx	Generalized mass	VIBRRNF
5.	GSTIFxx	Generalized stiffness	VIBRRNF
6.	TØTWTxx	Total mass	VIBRRNF
7.	SFdddxx	Subset freedom and node numbers	VIBRRNF
8.	VSETCØN	Vibration set conditions	VIBRRNF
9.	BSETCØN	Buckling set conditions	BUCKRNF
10.	EIGENxx	Eigenvalues	BUCKRNF

Where xx is the vibration or buckling set number.

A detailed description of the matrix formats is contained in reference 1-2.

716.3 PROGRAM METHOD

This module consists of a primary overlay, VIBRPRG, and six secondary overlays, PICKUP, EIGENC, EXPAND, RIGIDBM, EQCHECK, and PSUBSET.

716.3.1 ØVERLAY VIBRØLF(1,0)/BUCKØLF(1,0) - VIBRPRG

a. Establishes the FET (file environment table) and associated buffers for the seven sequential blocked scratch files which are used by the eigenvalue and eigenvector generating routines.

b. Controls the logical flow of the calls to the secondary level overlay programs for the desired eigenvalue and eigenvector results.

c. Sets values in module common blocks, including /TAPES/.

716.3.2 ØVERLAY(VIBRØLF,1,1) - PICKUP

a. Establishes the values of /PRGPAR/ common block. The variables are first set to defaults, then modified to comply with the parameters on the EXECUTE VIBRATIØN or EXECUTE BUCKLING statement. (These parameters are transmitted via the /CØNPARS/ common block.)

b. If errors are found in the EXECUTE statement, PICKUP prints error messages as they occur and keeps a tally of warnings and fatal errors in the /KERRØR/ common block.

c. Routine RMATRIX reads the mass and stiffness or flexibility matrix (for vibration analysis), or the stiffness and geometric stiffness matrices (for buckling analysis). The matrices are expanded from the ATLAS user matrix form to a full upper triangular form in a single dimensional array by routine EXPAN and then written on scratch file (VIBRSC1) by routine WRTSCR for the eigensolution routines called later in EIGENC.

d. Insert set and stage numbers, mass and stiffness or flexibility names in vibration set condition matrix.

716.3.3 ØVERLAY(VIBRØLF,1,3) - EIGENC

Solves the eigenvalue/eigenvector problem.

a. Establishes the size of the dynamic storage area, pointers for the required arrays, and calls routine EIGENS which calculates the frequencies and modes.

716.3.3.1 EIGENS (FØRTRAN subroutine)

a. Reads the matrices for the eigenvalue problem into core from scratch storage.

b. Calls KEDSY3 or REDSY4 to reduce the eigenproblem $[K]\{x\} = \lambda[M]\{x\}$ to the form $[P]\{z\} = \lambda\{z\}$, where $\{z\} = [L(t)]\{x\}$ and $[M] = [L][L(t)]$, $[P] = [L]^{-1}[K][L(t)]^{-1}$. [M] must be positive definite and [K] and [M] are stored in compact upper triangular form on scratch disk files.

c. Calls TRIDIL to tridiagonalize [P] by Householder's reduction, forming matrix [D], and finds the norm of matrix [D]. The norm is defined as the maximum value of the sum of the absolute values of the elements in a row of [D].

d. If QR or Sturm extraction method is requested EIGENS calls SYMQR or SEPAR2 respectively to find eigenvalues from the real symmetric tridiagonal matrix, [D]. QR finds all eigenvalues, Sturm only those requested. Calculates the cutoff value $(\text{norm} * 10^{-12})$. If absolute value of an eigenvalue is less than the cutoff value, the eigenvalue is set to zero.

e. If eigenvectors are not requested, EIGENS skips to step j.

f. Loops on the number of eigenvectors requested calling VECTØL to find eigenvectors, y, corresponding to the given eigenvalues, λ, by the Wielandt inverse method.

$$([D]-\lambda[I])x = y(i)$$

$$y(i+1) = x/||x||$$

where $||x||$ is the Euclidean norm of x.

g. If nondiagonal mass indicator is on, EIGENS calls RECØVL to transform eigenvectors back to the original reference by

$$x = [L(t)]^{-1}y$$

h. Calls NØKMA to normalize the eigenvectors. The normalizing values, r, are the inverse of the largest element of the vector, and are stored as the unscaled generalized masses.

i. Writes the eigenvectors on scratch VIBRSC3.

j. Eigenvalues are transformed as follows:

$f=l^2$ - when eigenvalue problem is formulated in terms of stiffness and mass matrices

$f=1/l^2$ - when eigenvalue problem is formulated in terms of flexibility and mass matrices

$f=-1/l$ - when eigenvalue problem is formulated in terms of stiffness and geometric stiffness matrices

k. Writes the eigenvalues, f, on scratch file VIBRSC2.

l. Write the unscaled generalized masses, r, on scratch file VIBRSC2.

716.3.4 ØVERLAY(VIBRØLF,1,4)-EXPAND

Generate generalized mass and generalized stiffness matrices.

a. Reads the eigenvalues, f, from scratch file and writes them on VIBRRNF with the index name FREQSxx (vibration analysis) or EIGENxx on BUCKRNF (buckling analysis).

b. Reads the unscaled generalized masses, r, from scratch file and generates the vectors of scaled generalized masses, mbar, and generalized stiffnesses, kbar, by:

$mbar = r^2$

kbar = mbar*l for stiffness type or

kbar = mbar/l for flexibility type

c. Reads eigenvectors, y, from scratch file and writes the vectors on random disk SC00RNF with index names VECxxx where xxx is the column number of the mode matrix.

d. Expands the generalized mass and generalized stiffness vectors to rectangular matrices with zero off-diagonal elements.

e. For the zero frequency partition, calculate generalized mass matrix by $[\phi(t)][M][\Phi]$.

f. Writes the generalized mass matrix, [mbar], and generalized stiffness matrix, [kbar], on VIBRRNF with the index names GMASSxx and GSTIFxx respectively.

716.3.5 ØVERLAY(VIBRØLF,1,5) - RIGIDBM

Specified rigid body modes, RBM, and total mass matrix for the rigid model.

a. Reads retained freedom vector, nodal data matrix and nodal correspondence table.

b. Reads th user specified uncoupled rigid body freedom indicators URBM, TRBM, PRBM, and codes them as follows:

Code	Freedom
1	x-translation
2	y-translation
3	z-translation
4	x-rotation
5	y-rotation
6	z-rotation

c. Reads condition summary matrix. From the given mass matrix index name or from the concentrated mass subset number, finds the row of the condition summary matrix where the total mass elements are stored.

d. Establishes the total mass matrix as follows:

$$[J] = \begin{bmatrix} Wt & & & 0 & & \\ & Wt & & & & \\ & & Wt & & & \\ \hline & & & Ixx & Ixy & Ixz \\ & 0 & & Ixy & Iyy & Iyz \\ & & & Ixz & Iyz & Izz \end{bmatrix} \cdot \frac{1}{g}$$

where g = mass matrix division factor from matrix MPARMS1 on DATARNF.

e. Picks up the coordinates of the center of gravity (Xcg, Ycg, Zcg) from the condition summary matrix.

f. If full model skip to step j.

g. For a half model, RIGIDBM finds the plane of symmetry of the whole system and offset (distance from the plane of symmetry to the c.g.) from the HALF parameter in the execute statement and the center of gravity coordinates of the half system.

h. Establish the transformation matrices: [ar] and [al] to move the half model center of gravity to the plane of symmetry from the right or left, respectively.

i. Transforms the total mass matrix from the half system c.g. to the whole system c.g. with the following equation:

$$[JH] = \frac{[ar(t)][J][ar]+[al(t)][J][al]}{2}$$

(These matrix operations are performed using SNARK commands.) Replaces [J] with [JH].

j. If URBM indicated, i.e., RBM relative to the global axis, skip to step u.

k. If TRBM indicated, i.e., RBM relative to the user selected triad, skip to step p.

l. For PRBM, i.e., RBM relative to the principal axis, RIGIDBM establishes SNARK position numbers for scratch arrays, extracts 3X3 inertia partition, [It], from total mass matrix, and calls routine VALVCT to extract eigenvalues and eigenvectors. RIGIDBM then reorders eigenvalues and eigenvectors such that the largest element of an eigenvector lies on the diagonal of the eigenvector matrix, normalizes the eigenvector by dividing by its length and transposes the eigenvectors, forming the matrix which rotates from global to principal axes. Expands the eigenvalues to a full 3X3 diagonal matrix, which replaces [It], forming [JP]. Replaces [J] with [JP].

n. Converts direction cosines of the 3X3 eigenvector matrix to Euler angles, in the sequence: yaw, pitch, roll; and prints them.

o. Skips to step s.

p. For the TRBM (user selected triad), RIGIDBM establishes position numbers for scratch arrays, and calls routine RØTRAN to generate 3X3 rotation matrix [r], from user specified angles. The rotation matrix rotates the reference frame from global to user axes.

q. Establishes translation matrix, [G], and multiplies [r][G] to get [rG].

r. Establishes the transformation matrix, [a], which transforms the total mass matrix from the global reference frame to the user specified reference frame, transforms the total mass matrix by:

$$[JU] = [a(t)][J][a] \text{ and replaces } [J] \text{ with } [JU]$$

s. Generates RBM (rigid body modes) in the following steps:

1. Establishes SNARK matrix position numbers for coordinates of c.g., node location, and difference of c.g. and node location.

2. Reads local coordinate system matrix.

3. Loops on the number of retained degrees of freedom, steps 3 through 10, to generate the component RBM (the 6X6 matrix for each node location).

4. Picks up internal node number from retained freedom vector, corresponding user node number and pointer to nodal data matrix for node coordinates from nodal correspondence table, and node coordinates from nodal data matrix.

5. Computes the difference array relative to the specified frame.

6. Establishes the mode component matrix by appropriate insertion of the elements of the difference array.

7. If analysis system indicator is off, skip steps 8 and 9.

8. RIGIDBM gets the global to analysis frame transformation matrix from the local

coordinate system matrix and establishes the rotation matrix between user specified frame and analysis frame.

9. Forms the transformation matrix, [a], between the user specified frame and the analysis frame and transforms the component mode matrix, [C].

10. Selects the RBM freedoms from the component matrix as requested by the user and inserts these freedoms in the appropriate column of the modes matrix.

t. If a total mass index name is specified by the user, RIGIDBM writes the total mass matrix onto VIBRRNF.

u. If the normalizing indicator is off, RIGIDBM skips step v.

v. Calls routine NØRMA to normalize each RBM column of the modes matrix by dividing by the largest value, r, and saves it. Establishes the normalizing transformation matrix, [Norm], where the diagonal elements are the inverses of the absolute elements, r, found above, and normalizes the RBM partition of the generalized mass matrix by:

$$[JN] = [Norm][J][Norm]$$

Replaces [J] by [JN]

w. Replaces the flexible modes by the RBM modes by overwriting the respective column arrays, on the scratch random file SC00RNF.

x. Reads the generalized mass matrix. Replaces the partition of the generalized matrix effected by the RBM with the respective freedoms of the (normalized) total mass matrix, and writes the generalized mass matrix on VIBRRNF.

y. If the RBM generalized mass matrix name is specified by the user, RIGIDBM writes the generalized mass matrix on VIBRRNF with specified matrix name.

z. If the RBM generalized stiffness matrix name is specified by the user, RIGIDBM reads the

generalized stiffness matrix and then writes it on VIBRRNF with the specified matrix name.

716.3.6 ØVERLAY(VIBRØLF,1,6)-EQCHECK

Perform equilibrium and orthogonality checks.

a. Establish SNARK position numbers and matrix sizes for the various matrices used in this program.

b. Read the eigenvalues [f], mass or geometric stiffness [M], and stiffness [K] matrices.

c. Performs equilibrum (compatibility) checks, steps d through g, for each vector.

d. Reads an eigenvector {y}.

e. If stiffness type requested, EQCHECK calls a series of routines (MULTI or READM, SMULTUR, and SUBTR) to solve the following equation:

$$[delta] = [K-f*M]\{y\}$$

f. If flexibility type requested, EQCHECK calls the same series of routines as in step e to solve the following equation:

$$[delta] = [FM-f*I]\{y\}$$

where [F] is the flexibility matrix

g. Computes and prints the following RMS equilibrium check result by transpose-multiply and squareroot routines:

$$RMS = SQRT(1/(NM-1)*[delta(t)]*[delta])$$

where NM is the number of modes.

h. If rigid body modes requested, EQCHECK finds the number of uncoupled rigid body translations and rotations specified by the user, reads the generalized mass, [m], and mass, [M], and loops on the number of vectors, steps i through m to perform the mode shape orthogonality check.

i. Reads the pivot eigenvector [y(i)]

j. Calls routine MULTI or SMULTUR to compute the temporary result, [T1], by the following equation

$$[T1] = [M][y]$$

k. Performs steps l through m for each vector, to compute a row of the orthogonality check results.

l. Reads an eigenvector [y(j)], and calculates

$$[T2] = [T1(t)][y(j)].$$

m. Computes the value, TMP, by

$$TMP = ABS(T2)/MIN(M(i,i),M(j,j))$$

The value, TMP, is represented by a symbolic code given in the table below; and which is printed

Symbol	Indicates
0	$TMP < 10^{-12}$
1	$10^{-12} < TMP < 10^{-8}$
2	$10^{-8} < TMP < 10^{-4}$
X	$10^{-4} < TMP$

n. Forms the full modes matrix, [Φ], by reading eigenvectors into the appropriate columns of [Φ].

o. Write mode shapes on VIBRRNF with index name MØDESvs (vibration analysis) or on BUCKRNF with index name MØDESvs (buckling analysis).

716.3.7 ØVERLAY(VIBRØLF,1,7) - PSUBSET

Extract mode shapes associated with given nodal data subsets.

a. Establish SNARK position numbers for the various matrices used in the program.

b. Read mode shapes, nodal correspondence table, and retained freedom vector.

c. Performs steps d through h for each specified subset, forming the subset mode shape matrix and subset freedom and node numbers.

d. Reads the nodal data subset matrix, to get the user node numbers for the subset.

e. Finds the total number of freedoms of the subset (row dimension of subset mode shapes, NDØF).

f. Establishes the size of the subset mode shape matrix (NDØF,NM), where NM is the number of modes (vectors).

g. Calls routine PICLØC to extract the rows of the mode shape matrix and form the subset mode shape matrix which is associated with the subset.

h. Pack the retained freedom indicator in the first thirty bits and user node number in the second thirty bits of a word and store the word in the subset freedom and node number matrix.

i. Writes the subset mode shape matrix on VIBRRNF or BUCKRNF with index name SMdddxx and the subset freedom and node number matrix on VIBRRNF with index name SFdddxx.

716.4 COMMON BLOCK USAGE

	ØVERLAY: VIBRØLF						
NAME	(1,0)	(1,1)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
KERRØR	x	x	x	x	x	x	x
CØNPARS	x	x	x	x	x	x	x
KQRNDM	x	x	x	x	x	x	x
KQBUFP	x	x	x	x	x	x	x
PRGPAR	x	x	x	x	x	x	x
TAPES	x	x	x	x	x	x	x
DIMS	x	x	x	x	x	x	x
CHKPKNT	x	x	x	x	x	x	x

716.4.1 /KERRØR/,/CØNPARS/,/KQRNDM/,/KQBUFP/ are ATLAS system common blocks, described in section 100.2.

716.4.2 /PRGPAR/ -- program variables

DYNTYPE - type of dynamic matrix
= 1, stiffness
= 2, flexibility
= 3, buckling

MASTYPE - mass or geometric stiffness matrix format
= 0, diagonal
= 1, non-diagonal

VALTYPE - eigenvalue extraction method
= 0, Q-R
= 1, Sturm

N - number of equations in eigenvalue problem

NVAL - number of eigenvalues (frequencies) requested
NVEC - number of eigenvectors (mode shapes) requested
SET - data set number
STAGE - stiffness execution stage number
CØNMASS - concentrated mass subset number (not used)
HALF - model plane of symmetry indicator
= 0, full model
= 1, offset x, yz plane of symmetry
= 2, offset y, zx plane of symmetry
= 3, offset z, xy plane of symmetry
NØRM - rigid body mode normalizing indicator
= 0, do not normalize
= 1, normalize
IVSET - vibration set number or buckling set number
NAMERNF - VIBRRNF or BUCKRNF
KEYRBM - rigid body mode reference frame indicator
= 0, no rigid body modes
= 1, global axis, URBM
= 2, principal axes, PRBM
= 3, user defined axis, TRBM
XREF - origin of user defined triad relative
YREF to the global frame
ZREF
IRBMRI - number of R.B.M. rotations for user defined axis
IRBMR(6) - axis about which rotation is made for user defined axis
= 1, x-axis
= 2, y-axis
= 3, z-axis
RBMR(6) - angle of rotation about selected axis for user defined axis
NAMVSET - vibration/buckling set condition matrix index name
NAMMSS - mass matrix index name
NAMSTIF - stiffness/flexibility matrix index name
NAMCØØR - nodal coordinates index name
NAMRFV - retained freedom vector index name
NAMNCT - nodal correspondence table index name
NAMVAL - eigenvalues (frequencies) index name
NAMVEC - eigenvectors (mode shapes) index name
NAMRIN - mode matrix to be altered index name (RBM)
NAMRØØ - altered mode matrix index name (RBM)
NAMTMA - transformed total mass index name (not used)
NAMGMA - generalized mass index name
NAMRGM - altered generalized mass index name (RBM)
NAMGST - generalized stiffness index name

NAMRGS - altered generalized stiffness index name (RBM)
NRBMS - number of rigid body modes selected (≤3)
IRBMS - current rigid body mode (RBM)
NØEIG - eigensolution equation indicator
= 0, no equation specified in execute list
= 1, equation specified
ISUBSET - number of subsets requested for eigensolution results
ISUBRBM - number of subset requested per RBM (not used)
SUBSETS(100) - index names of stiffness nodal subset matrix
SUBRBMS(100) - index names of stiffness nodal subset matrices for RBM
URBM - indicator for global reference axis
PRBM - indicator for principal reference axis
TRBM - indicator for user defined reference axis

716.4.3 /TAPES/ - scratch tape variable names

NTP - (VIBRSC1) - mass and stiffness/flexibility matrices in singly dimensioned array, stored upper triangular by rows
NTPX - (VIBRSC2) - used for intermediate
NTPY - (VIBRSC3) results while executing
KD3 - (VIBRSC4) partial out-of-core routines
KD4 - (VIBRSC5) for eigenvalue/eigenvector
KD5 - (VIBRSC6) solutions
KD6 - (VIBRSC7)
MTP - (VIBRSC2) - eigenvectors per mode
MTPX - (VIBRSC3) - eigenvalues and generalized mass

716.4.4 /DIMS/ - F.E.T.s for scratch tapes

TITLE(17) - title for vibration
V142SC1(548) - F.E.T. and buffer area for VIBRSC1
V142SC2(548) - F.E.T. and buffer area for VIBRSC2
V142SC3(548) - F.E.T. and buffer area for VIBRSC3
V142SC4(548) - F.E.T. and buffer area for VIBRSC4
V142SC5(548) - F.E.T. and buffer area for VIBRSC5
V142SC6(548) - F.E.T. and buffer area for VIBRSC6
V142SC7(548) - F.E.T. and buffer area for VIBRSC7

716.4.5 /CHKPRT/ - checkout parameter

IPRINT - checkout print indicator
=0, no printout
=1, print timing information only
=9, print everything for checkout

800. ATLAS POSTPROCESSORS

All ATLAS hardcopy output material is produced by one of the postprocessors. The PRINT postprocessor prints matrix values, processor input and output, and checkout quantities, such as checksums. The EXTRACT and GRAPHICS postprocessors prepare information which will drive one of the available offline plotters, and write it to the GRAPHICS output file of ATLAS. They also support interactive graphics processing.

801. PRINT POSTPROCESSOR

801.1 PURPOSE

The PRINT module handles all of the ATLAS printed output.

801.2 ACCESS

This module is called from the ATLAS control program by one of the following commands:

```
PRINT INPUT(PRØC,plist)
PRINT ØUTPUT(PRØC,plist)
PRINT MATRIX(Fname,plist)
PRINT MATRIXID(Fname,mlist)
PRINT MATRIX(Fname,CHECKSUM,plist)
PRINT MATRIX(Fname,CHECKPRINT,plist)
```

where:

PRØC - the type of data print desired
Fname - ATLAS random access file
plist - user selected execution parameters
mlist - list of matrix names

801.3 PRINT INPUT/PRINT ØUTPUT COMMANDS

Table 801-1 shows the available postprocessor print options, and the corresponding secondary overlay numbers. References are to sections in reference 1-1.

Table 801-1. Postprocessor Secondary Overlays

	PRINT INPUT		PRINT OUTPUT	
Print Type	Overlay	Reference	Overlay	Reference
STIFFNESS	(1,1)	252	n.a.	n.a.
BC	(1,3)	206	n.a.	n.a.
MASS	(1,4)	238	(1,5)	238
STRESS	n.a.	n.a.	(1,6)	254
DISPLACEMENTS	n.a.	n.a.	(1,7)	254
DESIGN	(1,12)	212	(1,13)	212
INTERACT	(1,9)	230	(1,9)	230
LØADS	(1,10)	234	(1,11)	234
AF1	(1,14)	204	(1,15)	204
MACHBØX	(1,16)	236	(1,17)	236
RHØ3	(1,18)	250	(1,19)	250
DUBLAT	(1,20)	216	(1,21)	216
VIBRATIØN	n.a.	n.a.	(1,22)	258
ADDINT	n.a.	n.a.	(1,23)	202
FLUTTER	(1,30)	222	(1,31)	222
REACTIØNS	n.a.	n.a.	(1,11)	248
FLEXAIR	n.a.	n.a.	(1,24)	220
MATERIAL	(1,25)	240	n.a.	n.a.
FØRCES	n.a.	n.a.	(1,32)	254
EXSTRESS	n.a.	n.a.	(1,33)	254
EXDISP	n.a.	n.a.	(1,34)	254
BUCKLING	n.a.	n.a.	(1,22)	208
NØDAL	(1,1)	246	n.a.	n.a.
LAMINA	n.a.	n.a.	(1,38)	254
FREEBØDY	n.a.	n.a.	(1,39)	224

801.4 PRINT MATRIX/PRINT MATRIXID COMMANDS

These commands are described in section 200, reference 1-1. OVERLAY(PRINØLF,1,77B)-GPRINT does the actual printing using the SNARK PRNTMAT command.

801.4.1 Checksum/Checkprint matrix printing

The form of the command is as follows:

```
PRINT MATRIX(Fname,Chkword,INDEX=XXXXXXX,
R1=N1,R2=N2,C1=N3,C2=N4,FØRMAT=format,SKIPL=N5)
```

where

Fname - ATLAS random access file name
Chkword - Either CHECKSUM or CHECKPRINT

XXXXXXX - Matrix name to be printed - Asterisk option as described in section 200, reference 1-1, is available. The parameters INDEX=name may be repeated.

The following options apply only to the CHECKPRINT option.

N1	- First row to be printed (default-1)
N2	- Last row to be printed (default-last row of matrix)
N3	- First column to be printed (default-1)
N4	- Last column to be printed (default-last column of matrix)
format	- Print format: (4HREAL(default), 7HINTEGER,5HØCTAL,6HMIXEDI,6HMIXEDØ)
N5	- >0, Skips N5 lines before printing ≤0, Skips to new page, skips |N5| lines before printing (default - N5=2)

These commands result in the calling of ØVERLAY(PRINØLF, 1,76B)-PRNTSUM.

801.5 ADDITION OF PRINT POSTPROCESSORS FOR ATLAS MODULES

a. Changes to ØVERLAY(PRINØLF,1,0)-PRINPK

1. Change dimensions of ISØNAM and ISØNUM arrays, which contain the module names and secondary overlay numbers, respectively. Note that ISØNUM is doubly dimensioned.

2. Change DATA statements to reflect the new names in ISØNAM and new overlay numbers in ISØNUM.

3. Change value of LØØP, set in a DATA statement, to the new number of modules represented.

4. Add appropriate statement numbers to computed GØ TØ statement, which jumps to appropriate secondary overlay call.

5. Call FILEADD (sec. 900) if ATLAS random files are required by secondary overlay. Use array FETB for call.

b. Changes to PRINT dummy routines

1. Add ØVERLAY card in appropriate numerical sequence.

2. Add PRØGRAM, SUBRØUTINE and END cards as required.

802. EXTRACT POSTPROCESSOR

802.1 GENERAL INFORMATION

802.1.1 Purpose

To extract from ATLAS generated data the items that are recognized in the ATLAS DATA DIRECTORY, and to write them in indexed sequential form.

802.1.2 Access

This module is called from the ATLAS control program via the EXECUTE EXTRACT (plist) statement. (sec. 218, ref. 1-1)

802.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1	ADATDIR	ATLAS Data Directory	DATARNF
2	BSETCØN	BSET Control matrix	STIFRNF
3	CVEC01a	Cargo condition vector	MASSRNF
4	DCNTRba	Displacement control matrix	STRERNF
5	DCØØRba	Loadcase correspondence table	STRERNF
6	DIØØ1ba	Displacement matrix	STRERNF
7	EIGEN01	Buckling Eigenvalues	BUCKRNF
8	FPijklm	Flutter plot data control matrix	FLUTRNF
9	FPijklmn	Flutter plot data matrix	FLUTRNF
10	FREQS01	Vibration eigenvalues	VIBRRNF
11	FVEC01a	Fuel vector matrix	MASSRNF
12	KLØCØØa	Local coord. systems matrix	DATARNF
13	DMELNØa	Flexible element nodal matrix	DATARNF
14	KNC100a	Nodal correspondence table	DATARNF
15	KNØALTa	Nodal data matrix	DATARNF
16	KPRAMS1	Stiffness parameter matrix	DATARNF
17	KRFV0ba	Retained freedom vector	DATARNF
18	KSF001a	Flexible element data matrix	DATARNF
19	KUFRT0a	User freedom reference table	DATARNF
20	MØDES01	Buckling eigenvectors	BUCKRNF
21	MØDES01	Vibration eigenvectors	VIBRRNF

22	MPARMS1	Mass parameter matrix	DATARNF
23	MØØ1cba	Strength min. margins of safety matrix	DESIRNF
24	NØØ1cba	Thermal min. margin of safety matrix	DESIRNF
25	PVEC01a	Passenger condition vector	MASSRNF
26	SCN01ba	Stress control matrix	STRERNF
27	SDTNLST	Label subset list matrix	DATARNF
28	SEKddda	Stiffness element subset matrix	DATARNF
29	SEMddda	Mass element subset matrix	DATARNF
30	SFddd01	Subset freedom and node nos.	VIBRRNF
31	SITM001	Label subset matrix	DATARNF
32	SLCSTba	Stress loadcase specification table	STRERNF
33	SMddd01	Subset mode shapes matrix	VIBRRNF
34	SNKddda	Nodal subset matrix	DATARNF
35	SPKddda	Ordered nodal subset matrix	DATARNF
36	ST001ba	Stress matrices	STRERNF
37	TAPLWTa	Condition summary matrix	MASSRNF
38	TØTLWTa	Total mass properties matrix	MASSRNF
39	VSETCØN	VSET control matrix	VIBRRNF

	Name	Description	File
Output			
1	ADATDIR	Copy of the ATLAS Data Directory	EXTRRNF
2	DBEXCØN	Extract Control matrix	EXTRRNF
3	DBEXTNM	Extract name list	EXTRRNF
4	DBINDEX	Attribute name list matrix	EXTRRNF
5	DB001XX	Extracted data matrix	EXTRRNF
6	DBINDXX	Low index list matrix	EXTRRNF
7	ALNMLST	Altitude subset list matrix	EXTRRNF
8	ALNM001	Altitude subset matrix	EXTRRNF
9	CANMLST	Flutter case subset list matrix	EXTRRNF
10	CANM001	Flutter case subset matrix	EXTRRNF
11	CSNMLST	Changeset subset list matrix	EXTRRNF
12	CSNM001	Changeset subset matrix	EXTRRNF
13	CØNMLST	Condition subset list matrix	EXTRRNF

14	CØNM001	Condition subset matrix	EXTRRNF
15	CYNMLST	Cycle subset list matrix	EXTRRNF
16	CYNM001	Cycle subset matrix	EXTRRNF
17	LCNMLST	Loadcase subset list matrix	EXTRRNF
18	LCNM001	Loadcase subset matrix	EXTRRNF
19	MDNMLST	Mode subset list matrix	EXTRRNF
20	MDNM001	Mode subset matrix	EXTRRNF
21	RSNMLST	Retainset subset list matrix	EXTRRNF
22	RSNM001	Retainset subset matrix	EXTRRNF
23	SUBSLST	Node/element subset list matrix	EXTRRNF
24	SPKddda	Copy of ordered modal subset matrix	EXTRRNF
25	SDTNLST	Label subset list matrix	EXTRRNF
26	SITM001	Label subset matrix	EXTRRNF

802.3 PROGRAM METHOD

The EXTRACT module consists of one primary overlay, EXTRACT and 18 secondary overlays.

802.3.1 OVERLAY (EXTRØLF,1,0) - EXTRACT

a) Sets up common block /C1/

b) Calls ØVERLAY (1,1) to interpret the control statement.

c) If it is a reformatter run it calls ØVERLAY (1,4) and (1,5) and returns.

d) If it is not a reformatter run, ØVERLAY (1,2) is called to set up the Extract Control information and sets up a loop governed by the variable IMG. Counting bits from left to right as 1 to 60, if the n'th bit of IMG is on, the secondary ØVERLAY (1,n+8) is called. Each of these secondary overlays extracts a certain type of data.

e) Calls ØVERLAY(1,3) to reorder the data extracted in step 'd) above.

802.3.2 ØVERLAY (EXTRØLF,1,1) - EXCON

a) Opens required files

b) Initializes common block /EXMØDCM/

c) Interprets control statement contained in /CØNPARS/ and stores the parameter values specified in appropriate locations in /EXMØDCM/

d) Checks for legality of parameters and conflicting specifications

e Sets up standard label subsets that are not predefined by the user, but are used in the control statement

f) Scans the label subsets and compiles the attribute dependency, element type dependency and matrix group list containing the data specified by the labels in the label subsets

g) Checks if all required attributes and/or subsets are specified

h) Sets up appropriate default values as required

i) Closes file

802.3.3 ØVERLAY (EXTRØLF,1,2) - EXDAT1

a) Opens required files

b) Generates the control information for the current extract

c) Updates the Extract Control matrix

d) Closes files

802.3.4 ØVERLAY (EXTRØLF,1,3)

a) Opens required files

b) Reorders the extracted data in terms of increasing index values. The following algorithm is used:

(1) A scratch matrix SC1 is defined to use up available cor

(2) A pass is made on all partitions of the extracted data and the index portion of each of the keys is extracted and put into the scratch matrix.

(3) The scratch matrix SC1 is compacted down to actual size and another scratch matrix SC2 of the same size is defined.

(4) Array SC1 is searched in sequence of ascending index values. The sequence number for each entry in SC1 is recorded in the corresponding entry in SC2.

(5) A scratch matrix SC3 is defined to use up the available core. An estimate is made as to how may keys and the associated data records (say N) may be contained in SC3.

(6) A sequential pass is made on the preliminary extracted data matrix partitions until all of the keys 1 thru N in the reordered sequence as laid out in SC2 are encountered. These keys and data records are passed into SC3 and SC3 is written out as a partition of the extracted data, containing the first N keys and associated data.

(7) The process (4) thru (6) is repeated until all data has been reordered.

c) Write out the DBINDXX matrix containing the lowest indexes in each of the extracted data partitions.

d) Closes files

802.3.5 <u>ØVERLAY (EXTRØLF,1,4) - RESØRT</u>

a) Opens required files

b) Sets up reordering specifications

c) Calls subroutine SERODAT to record the extracted data. Subroutine SERODAT contains code almost identical to that in the program REORDAT. The same basic algorithm is used for resorting the data

d) Closes files

802.3.6 ØVERLAY(EXTRØLF,1,5) - REFØRD1

REFØRD1 is a FØRTRAN program that writes random access records on a node/element basis from the data re-sorted by the program RESØRT.

a) Opens files

b) Reads SNARK generated matrices using RDRMX1

c) Writes out random access records using WRTER.

d) Closes files.

802.3.7 ØVERLAYS (EXTRØLF,1,11B) to (EXTRØLF,1,25B)

These overlays extract the data from ATLAS files and write them out on EXTRRNF. The actions taken by each overlay are identical in nature. They are:

a) Open required files

b) Read required ATLAS matrices

c) Call subroutine to extract data from ATLAS matrices

d) Write the extracted data matrices on EXTRRNF

e) Closes files.

The overlay numbers, the program names and the data extracted by each are summarized in the following table.

OVERLAY No.	PROG. NAME	DATA
(1,11)	EXTNØDE	NØDE-Related Data
(1,12)	EXTVMØD	Vibration mode-shape data
(1,13)	EXTVGVF	Flutter V/G-V/F curve related data
(1,14)	EXTSEDT	Stiffness element properties
(1,15)	EXTDESP	Nodal displacement data
(1,16)	EXTELST	Stiffness element stresses
(1,17)	EXTLØDB	Loadability curve related data
(1,20)	EXTKUDC	User reference table and loadcase corresp. table
(1,21)	EXTSMS	Stress margins of safety data
(1,22)	EXTMEDT	Mass element id and connectivity data
(1,23)	EXTEMØD	Buckling mode-shape data
(1,24)	EXTTMS	Thermal margins of saqety data
(1,25)	EXTMATR	ATLAS matrices

802.4 ORGANIZATION OF EXTRACTED DATA

802.4.1 General Comments

The following philosopy is used to organize the extracted data:

The idealized structure (nodes and elements) is the basic entity to be dealt with. It can be completely described in terms of nodal coordinates, element connectivity and stiffness properties. When the entity is subjected to an influence such as a loadcase or a vibration mode, its behavior can also be described in terms of the behavior of the nodes (i.e. the displacement components) and the behavior of the elements (stress components). The data associated with the description of the structure and its behavior is the basic data that is to be extracted by the EXTRACT processor. Each such data item is assigned a unique label. The ATLAS DATA DIRECTORY lists all these labels and provides a description for each (see sec. 802.6).

An execution of the ATLAS system generates a large number of data items. This total data is identifiable in terms of some major system parameters such as the set number, the stage number and the loacase labels. In addition, the concept of subsets is used to identify selected parts of the data. The node/element subsets identity parts of the idealized structure. The label subsets identify parts of the data items associated with the nodes and elements. In addition to these subsets, the user specifies subsets of loadcases, vibration mode-shapes, flutter conditions and other parameters of this nature, in THE EXECUTE EXTRACT command.

These major system parameters and the subsets defined are regarded as the attributes of the system and the extracted data is identified and organized in terms of the numeric values of these attributes.

802.4.2 System Attributes

The following 17 quantities are regarded as system attributes for the ATLAS system and all the data items extracted are recognized in terms of the value of one or more of these attributes

(1) Extract Name - This is the name specified by the user in a particular EXECUTE EXTRACT command.

(2) Stiffness set number
(3) Mass set number
(4) Stage number
(5) Loadcase subset number
(6) Mode subset number
(7) Flutter case subset number
(8) Altitude subset number
(9) Node/element subset number
(10) Label subset number
(11) Passenger condition number
(12) Cargo condition number
(13) Fuel condition number
(14) Changeset subset number
(15) Retainset subset number
(16) Flutter condition subset number
(17) Design cycle subset number

An attribute may be a single or multiple valued entity. Attributes 1,2,3,4,11,12 and 13 fall in the first category, e.g., stiffness and mass set numbers constitute one value only, whereas 5 thru 10 and 14 thru 17 fall in the second category, e.g., a loadcase subset may have many loadcases associated with it and a node/element subset consists of several nodes/elements.

Attributes may represent integer, floating point, or alphanumeric values. All attributes except 1,5,8 and 10 contain integer values. Attributes 1,5 and 10 have alphanumeric values, and attribute 8 contains a floating point value. All the alphanumeric and/or floating point values are put into a list and their sequence number in the list is used as the integer number associated with the particular non-integer value.

802.4.3 Attribute Usage

The attributes that are associated with data may be used to extract and organize data in one of the following manners:

1) To form the index name of the matrices containing the extracted data.

2) To form the index part of the keys that are associated with the extracted data.

3) The values of the components of an attribute are used in forming the index part of the keys that are associated with it.

In addition to the above usage the attribute values may be used in identifying the matrix index names and appropriate data items in the ATLAS matrices from which the data is extracted.

The current usage of the attribute values in identifying the extracted data is as follows:

Attribute Number 1 is used as specified in (1) above while attributes 5 thru 9 and 14 thru 17 are used as specified in (3) above.

802.4.4 Organization of Data

The extracted data is organized in the following manner:

The data extracted by one EXECUTE EXTRACT command is stored into a series of matrices of maximum size 3000 words. As many matrices as required are written. The matrix index names are formulated as DB001XX, where 001 is the numeric partition number and XX is the extract number in display code. (Each EXECUTE EXTRACT command in a job is assigned a sequential extract number by the processor.)

The data within each of the DB001XX matrix partitions is organized as follows:

WORD 1:	Bits 59-30	Unused
	Bits 29-15	No. of keys in the matrix
	Bits 14-0	Lowest key in the matrix

WORDS 2 thru n+1 contains n keys. Each key is composed as follows:

Bits 59-48	Pointer in the matrix where the data record associated with this key is stored.
Bits 47-36	Length (No. of words) of the associated data record.
Bits 35-0	Index. The index is formed by using the integer values of selected attributes or their components.

WORDS n+2 thru L (the last word):

Contain the data records for the n keys above.

A matrix DBINDXX is written out for each extract. This matrix contains as many rows as there are partitions of the extracted data matrices. Each row contains the lowest index (i.e., bits 35-0 of Word 2) in each of the corresponding extracted data matrices.

A matrix DBEXCØN is written out on EXTRRNF. This matrix contains a section for every extract for which the EXTRRNF contains data. Each section contains complete details of the nature of the data extracted.

802.4.5 The Data Index

The following table describes the manner in which the various attributes are used in forming the index.

ATTRIBUTE	USAGE TYPE (see sec. 802.4.3)	FIELD
LOADCASE SUBSET	3	Bits 35-24 or 11-0*
MODE SUBSET	3	Bits 35-24
FLUTTER CASE SUBSET	3	Bits 35-30
ALTITUDE SUBSET	3	Bits 17-12
NODE/ELEMENT SUBSET	3	Bits 23-0 or 35-12*
CHANGESET SUBSET	3	Bits 29-24
RETAINSET SUBSET	3	Bits 23-18
FLUTTER CONDITION SUBSET	3	Bits 17-12
DESIGN CYCLE SUBSET	3	Bits 35-24

NOTE: Attributes that use the same field are never associated with the same data items. Thus a unique INDEX is associated with each data item.

*Only for standard label subset names DISPRINT and STRPRINT

802.4.5.1

The node and/or element internal ID's are the components of the attribute "node/element subset." These together with the following additional information obtained from the ATLAS DATA DIRECTØRY (sec. 802.6) is used in forming the index.

For each node or element-related item the ATLAS DATA DIRECTØRY has assigned the element type (0 for nodes, 1 thru 13 for elements), a code (0 thru 6) and a sequence no. (1 thru n) so that each node/element-dependent label is uniquely identified by its element type, code and sequence number. Out of these, the element type and the code are used along with the internal id as follows:

bits 23-20 element type (0 thru 13)

bits 19-17 code (0 thru 6)

bits 16-0 Internal ID.

This coupled with the use of bits 35-24 for loadcase or mode or cycle numbers results in the data being sorted by loadcase/mode/cycle, element type, code and internal ID in that sequence.

The above usage of node element subset components is standard except when the standard label subset names DISPRINT and STRPRINT are used. In that case the arrangement is

bits 35-19 user ID

bits 18-15 element type

bits 14-12 code

bits 11-0 loadcase number

This results in sorting of the extracted data by user ID and loadcases in that order.

The data record corresponding to an index contains in general, all values, the labels for which have the same code, e.g., the nodal coordinates x, y and z have, code=1 and sequence numbers 1, 2 and 3. Whenever the label subset contains these 3 labels, the data record will contain the 3 values in the sequence x, y and z. If the label subset contains only the labels x and z, the corresponding data record will be 2 words long and the values will be x and z. All node and element related data is organized in this fashion. See ATLAS DATA DIRECTØRY for labels, their codes and sequence numbers (sec. 802.6).

The codes have been devised primarily from the viewpoint of having the data sorted for convenient and sequential access during generation of graphic displays.

802.4.6 Automated Generation of Indices

Currently the building of indices in the keys is hard-coded into various data extraction routines. However, some general provisions exist in the code to automate this process. They are as follows:

a) Data Statement for Array INDXDB in program EXCØN:

This array contains the program recognized names of all the attributes. The order of the attributes in this array may be considered as the hierarchy of the attribute usage in the index, i.e., if the first, seventh and eleventh attributes are to be used in forming the index, they would be used in that order from left to right in the index.

b) Data Statement for Array INDXFW in program EXCON:

The numbers here indicate the number of bits required to contain the values of the corresponding attributes (or their components) in the array INDXDB.

c) Data Statement for Array IDEF in program EXDATA

Each entry in IDEF corresponds to an attribute name in the array INDXDB and indicates the usage of the attribute in identifying the extracted data. The usage information for the n'th attribute is stored in the n'th 3-bit field from left to right in the n'th word. Number 2 in the field indicates that the attribute value is to be used in the matrix index name for the extracted data matrices, number 3 in the field indicates that the attribute value is to be used in the index in the key and number 4 in the field indicates that the values of the components of the attributes are to be used in forming the index in the key.

d) Subroutine KEYGEN

This subroutine is called from data extraction routines. It accepts the attribute names in the first row of the array IAREND dimensioned as (4,n) and pulls out the left most bit, the rightmore bit and the usage code (2,3 or 4) for the attributes in rows 2, 3 and 4 of IAREND.

802.5 COMMON BLOCK USAGE

NAME	ØVERLAY (EXTRØLF)																		
	1.0	1.1	1.2	1.3	1.4	1.5	1.11	1.12	1.13	1.14	1.15	1.16	1.17	1.20	1.21	1.22	1.23	1.24	1.25
CARDS		X		X															
CØNPARS	X	X		X															
C1	X	X	X		X		X	X	X	X	X	X	X	X	X	X	X	X	
C2	X	X	X				X	X	X	X	X	X	X	X	X	X	X	X	
EXMØDCM	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
KERRØR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
KQRNDM	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	
LRCØMM		X																	
MSKCØN				X	X														
SØRTELC				X	X														

802.5.1 /CØNPARS/
/KERROR/: System common blocks, described in
/KQRNDM/ section 100.2.

802.5.2 /LRCØMM/: Same as in ATLAS Library Routine LØDAREC

802.5.3 /CARDS/ This common block is similar to the system common block by the same name except for the lengths of the arrays and is used to simulate LØDAREC input of labels.

/C1/ Contains masking constants

/C2/ Contains the values for the number of words in a label subset and the number of finite element types recognized.

/EXMØDCM/ Common block for communication between secondary overlays. See comments in code.

/MSKCØN/ Contains masking constants.

/SØRTELC/ Contains information required by the reordering subroutines. See comments in code.

802.6 ATLAS DATA DIRECTORY

An engineering description of ITEM NAME in Table 802-1 is presented in reference 1-1.

Table 802-1. ATLAS Data Directory

NO.	ITEM NAME	ATTRIBUTE DEPENDENCY																	ELEMENT NO.	KODE NO.	SEQUENCE NO.	MATRIX GROUP
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18				
1)	BMA1		X							X										2	1	1	4
2)	BMA2		X							X										2	1	2	4
3)	BMA3		X							X										2	1	3	4
4)	BMA-VY(1)		X							X										2	2	1	4
5)	BMA-VY(2)		X							X										2	2	2	4
6)	BMA-VZ(1)		X							X										2	2	3	4
7)	BMA-VZ(2)		X							X										2	2	4	4
8)	BMA(1)		X							X										2	2	5	4
9)	BMA(2)		X							X										2	2	6	4
10)	BMCONSTR		X							X										2	2	7	4
11)	BMIC		X							X										2	0	1	4
12)	BMIY(1)		X							X										2	2	8	4
13)	BMIY(2)		X							X										2	2	9	4
14)	BMIZ(1)		X							X										2	2	10	4
15)	BMIZ(2)		X							X										2	2	11	4
16)	BMJ(1)		X							X										2	2	12	4
17)	BMJ(2)		X							X										2	2	13	4
18)	BMMAT		X							X										2	4	1	4
19)	BMMY(1)		X		X	X				X										2	3	1	6
20)	BMMY(2)		X		X	X				X										2	3	2	6
21)	BMMZ(1)		X		X	X				X										2	3	3	6
22)	BMMZ(2)		X		X	X				X										2	3	4	6
23)	BMN1		X							X										2	1	4	4
24)	BMN2		X							X										2	1	5	4
25)	BMP(2)		X		X	X				X										2	3	5	6
26)	BMSPS		X		X					X								X		2	5	1	9
27)	BMTEMP		X							X										2	4	2	4
28)	BMTPS		X		X					X								X		2	5	2	12
29)	BMT(2)		X		X	X				X										2	3	6	6
30)	BMVY(2)		X		X	X				X										2	3	7	6
31)	BMVZ(2)		X		X	X				X										2	3	8	6
32)	BRA1		X							X										8	1	1	4
33)	BRA2		X							X										8	1	2	4
34)	BRA3		X							X										8	1	3	4
35)	BRIC		X							X										8	0	1	4
36)	BRMAT		X							X										8	4	1	4
37)	BRN1		X							X										8	1	4	4
38)	BRN10		X							X										8	1	5	4
39)	BRN11		X							X										8	1	6	4
40)	BRN12		X							X										8	1	7	4
41)	BRN13		X							X										8	1	8	4
42)	BRN14		X							X										8	1	9	4
43)	BRN15		X							X										8	1	10	4
44)	BRN16		X							X										8	1	11	4
45)	BRN17		X							X										8	1	12	4
46)	BRN18		X							X										8	1	13	4
47)	BRN19		X							X										8	1	14	4
48)	BRN2		X							X										8	1	15	4
49)	BRN20		X							X										8	1	16	4
50)	BRN21		X							X										8	1	17	4
51)	BRN22		X							X										8	1	18	4
52)	BRN23		X							X										8	1	19	4
53)	BRN24		X							X										8	1	20	4
54)	BRN25		X							X										8	1	21	4
55)	BRN26		X							X										8	1	22	4
56)	BRN27		X							X										8	1	23	4
57)	BRN28		X							X										8	1	24	4
58)	BRN29		X							X										8	1	25	4
59)	BRN3		X							X										8	1	26	4
60)	BRN30		X							X										8	1	27	4
61)	BRN31		X							X										8	1	28	4
62)	BRN32		X							X										8	1	29	4
63)	BRN33		X							X										8	1	30	4
64)	BRN34		X							X										8	1	31	4
65)	BRN35		X							X										8	1	32	4
66)	BRN36		X							X										8	1	33	4
67)	BRN37		X							X										8	1	34	4
68)	BRN38		X							X										8	1	35	4
69)	BRN39		X							X										8	1	36	4
70)	BRN4		X							X										8	1	37	4
71)	BRN40		X							X										8	1	38	4
72)	BRN41		X							X										8	1	39	4
73)	BRN42		X							X										8	1	40	4
74)	BRN43		Y							X										8	1	41	4
75)	BRN44		X							X										8	1	42	4
76)	BRN5		X							X										8	1	43	4
77)	BRN6		Y							X										8	1	44	4
78)	BRN7		X							X										8	1	45	4
79)	BRN8		X							X										8	1	46	4
P01	BRN9		X							X										8	1	47	4

Table 802-1 (Continued)

NO.	ITEM NAME	ATTRIBUTE DEPENDENCY																	ELEMENT NO.	KODE NO.	SEQUENCE NO.	MATRIX GROUP
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18				
81)	BRSICPA1		X		X	X				X										8	3	1	6
82)	BRSICPA2		X		X	X				X										8	3	2	6
83)	BRSICPA3		X		X	X				X										8	3	3	6
84)	BRSMS		X		X					X								X		8	5	1	9
85)	BRTAL12		X		X	X				X										8	3	4	6
86)	BRTAL13		X		X	X				X										8	3	5	6
87)	BRTAL23		X		X	X				X										8	3	6	6
88)	BRTEMP		X							X										8	4	2	4
89)	BRTPS		X		X					X								X		8	5	2	12
90)	CCVAREF-L		X							X										13	2	1	4
91)	CCVAREF-U		X							X										13	2	2	4
92)	CCVEPS1L		X		X	X				X										13	3	1	6
93)	CCVEPS1U		X		X	X				X										13	3	2	6
94)	CCVEPS2L		X		X	X				X										13	3	3	6
95)	CCVEPS2U		X		X	X				X										13	3	4	6
96)	CCVCAM12L		X		X	X				X										13	3	5	6
97)	CCVCAM12U		X		X	X				X										13	3	6	6
98)	CCVIC		X							X										13	0	1	4
99)	CCVLAM01-L		X							X										13	2	3	4
100)	CCVLAM01-U		X							X										13	2	4	4
101)	CCVLAM02-L		X							X										13	2	5	4
102)	CCVLAM02-U		X							X										13	2	6	4
103)	CCVLAM03-L		X							X										13	2	7	4
104)	CCVLAM03-U		X							X										13	2	8	4
105)	CCVLAM04-L		X							X										13	2	9	4
106)	CCVLAM04-U		X							X										13	2	10	4
107)	CCVLAM05-L		X							X										13	2	11	4
108)	CCVLAM05-U		X							X										13	2	12	4
109)	CCVLAM06-L		X							X										13	2	13	4
110)	CCVLAM06-U		X							X										13	2	14	4
111)	CCVLAM07-L		X							X										13	2	15	4
112)	CCVLAM07-U		X							X										13	2	16	4
113)	CCVLAM08-L		X							X										13	2	17	4
114)	CCVLAM08-U		X							X										13	2	18	4
115)	CCVLAM09-L		X							X										13	2	19	4
116)	CCVLAM09-U		X							X										13	2	20	4
117)	CCVLAM10-L		X							X										13	2	21	4
118)	CCVLAM10-U		X							X										13	2	22	4
119)	CCVN1		X							X										13	1	1	4
120)	CCVN2		X							X										13	1	2	4
121)	CCVN3		X							X										13	1	3	4
122)	CCVN4		X							X										13	1	4	4
123)	CCVSIGMA1L		X		X	X				X										13	3	7	6
124)	CCVSIGMA1U		X		X	X				X										13	3	8	6
125)	CCVSIGMA2L		X		X	X				X										13	3	9	6
126)	CCVSIGMA2U		X		X	X				X										13	3	10	6
127)	CCVSMS		X		X					X								X		13	5	1	9
128)	CCVTAU12-L		X		X	X				X										13	3	11	6
129)	CCVTAU12-U		X		X	X				X										13	3	12	6
130)	CCVTEMP		X							X										13	4	1	4
131)	CCVTHICK-L		X							X										13	2	23	4
132)	CCVTHICK-U		X							X										13	2	24	4
133)	CCVTMS		X		X					X								X		13	5	2	12
134)	COVALPHAL		X							X										4	2	1	4
135)	COVALPHAU		X							X										4	2	2	4
136)	COVBETAL		X							X										4	2	3	4
137)	COVBETAU		X							X										4	2	4	4
138)	COVIC		X							X										4	0	1	4
139)	COVMAT		X							X										4	4	1	4
140)	COVN1		X							X										4	1	1	4
141)	COVN2		X							X										4	1	2	4
142)	COVN3		X							X										4	1	3	4
143)	COVN4		X							X										4	1	4	4
144)	COVSIGMA1L		X		X	X				X										4	3	1	6
145)	COVSIGMA1U		X		X	X				X										4	3	2	6
146)	COVSIGMA2L		X		X	X				X										4	3	3	6
147)	COVSIGMA2U		X		X	X				X										4	3	4	6
148)	COVSIG-S1L		X		X	X				X										4	3	5	6
149)	COVSIG-S1U		X		X	X				X										4	3	6	6
150)	COVSIG-S2L		X		X	X				X										4	3	7	6
151)	COVSIG-S2U		X		X	X				X										4	3	8	6
152)	COVSMS		X		X					X								X		4	5	1	9
153)	COVTAL12L		X		X	X				X										4	3	9	6
154)	COVTAU12U		X		X	X				X										4	3	10	6
155)	COVTEMP		X							X										4	4	2	4
156)	COVTMS		X		X					X								X		4	5	2	12
157)	COVTICIL		X							X										4	2	5	4
158)	COVTICIU		X							X										4	2	6	4
159)	COVTCI3L		X							X										4	2	7	4
160)	COVTCI3L		X							X										4	2	8	4

Table 802-1. (Continued)

NO.	ITEM NAME	ATTRIBUTE DEPENDENCY																	ELEMENT NO.	KODE NO.	SEQUENCE NO.	MATRIX GROUP
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18				
161)	CONTIZIL		X							X										4	2	9	4
162)	CONTIZIU		X							X										4	2	10	4
163)	CPLAREF		X							X										12	2	1	4
164)	CPLA1		X							X										12	1	1	4
165)	CPLA2		X							X										12	1	2	4
166)	CPLA3									X										12	1	3	4
167)	CPLA4		X							X										12	1	4	4
168)	CPLEPS1		X		X	X				X										12	3	1	6
169)	CPLEPS2		X		X	X				X										12	3	2	6
170)	CPLCAM12		X		X	X				X										12	3	3	6
171)	CPLID		X							X										12	0	1	4
172)	CPLLAM01		X							X										12	2	2	4
173)	CPLLAM02		X							X										12	2	3	4
174)	CPLLAM03		X							X										12	2	4	4
175)	CPLLAM04		X							X										12	2	5	4
176)	CPLLAM05		X							X										12	2	6	4
177)	CPLLAM06		X							X										12	2	7	4
178)	CPLLAM07		X							X										12	2	8	4
179)	CPLLAM08		X							X										12	2	9	4
180)	CPLLAM09		X							X										12	2	10	4
181)	CPLLAM10		X							X										12	2	11	4
182)	CPLN1		X							X										12	1	5	4
183)	CPLN2		X							X										12	1	6	4
184)	CPLN3		X							X										12	1	7	4
185)	CPLN4		X							X										12	1	8	4
186)	CPLSIGMA1		X		X	X				X										12	3	4	6
187)	CPLSIGMA2		X		X	X				X										12	3	5	6
188)	CPLSMS		X		X					X								X		12	5	1	9
189)	CPLTAU12		X		X	X				X										12	3	6	6
190)	CPLTEMP		X							X										12	4	1	4
191)	CPLTHICK		X							X										12	2	12	4
192)	CPLTMS		X		X					X								X		12	5	2	12
193)	DELX		X							X										0	2	1	1
194)	DELY		X							X										0	2	2	1
195)	DELZ		X							X										0	2	3	1
196)	GPLALPHA		X							X										6	2	1	4
197)	GPLA1		X							X										6	1	1	4
198)	GPLA2		X							X										6	1	2	4
199)	GPLA3		X							X										6	1	3	4
200)	GPLA4		X							X										6	1	4	4
201)	GPLA5		X							X										6	1	5	4
202)	GPLID		X							X										6	0	1	4
203)	GPLMA1		X							X										6	4	1	4
204)	GPLM1		X		X	X				X										6	3	1	6
205)	GPLM12		X		X	X				X										6	3	2	6
206)	GPLM2		X		X	X				X										6	3	3	6
207)	GPLN1		X							X										6	1	6	4
208)	GPLN2		X							X										6	1	7	4
209)	GPLN3		X							X										6	1	8	4
210)	GPLN4		X							X										6	1	9	4
211)	GPLSIGMA1		X		X	X				X										6	3	4	6
212)	GPLSIGMA2		X		X	X				X										6	3	5	6
213)	GPLSMS		X		X					X								X		6	5	1	9
214)	GPLTAU12		X		X	X				X										6	3	6	6
215)	GPLTEMP		X							X										6	4	2	4
216)	GPLTMS		X		X					X								X		6	5	2	12
217)	GPLT-PEND1		X							X										6	2	4	4
218)	GPLT-PEND2		X							X										6	2	3	4
219)	GPLT-PEND3		X							X										6	2	4	4
220)	GPLT-PEND4		X							X										6	2	5	4
221)	GPLT-PEND5		X							X										6	2	6	4
222)	GPLT-PEMB1		X							X										6	2	7	4
223)	GPLT-PEMB2		X							X										6	2	8	4
224)	GPLT-PEMB3		X							X										6	2	9	4
225)	GPLT-PEMB4		X							X										6	2	10	4
226)	GPLT-PEMB5		X							X										6	2	11	4
227)	LOCTAB		X		X															0	6	1	8
228)	LOADAR			X								X	X	X						0	0	1	7
229)	LUCTAB		X																	0	5	1	1
230)	MBMA1			X						X										2	1	1	10
231)	MBMA2			X						X										2	1	2	10
232)	MBMA3			X						X										2	1	3	10
233)	MBMID			X						X										2	0	1	10
234)	MBMN1			X						X										2	1	4	10
235)	MBMN2			X						X										2	1	5	10
236)	MCOVID			X						X										4	0	1	10
237)	MCOVN1			X						X										4	1	1	10
238)	MCOVN2			X						X										4	1	2	10
239)	MCOVN3			X						X										4	1	3	10
240)	MCOVN4			X						X										4	1	4	10

Table 802-1. (Continued)

NO.	ITEM NAME	ATTRIBUTE DEPENDENCY																	ELEMENT NO.	KODE NO.	SEQUENCE NO.	MATRIX GROUP
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18				
2411	MCOVN5			X						X										4	1	5	10
2421	MCOVN6			X						X										4	1	6	10
2431	MCOVN7			X						X										4	1	7	10
2441	MCOVN8			X						X										4	1	8	10
2451	MCOVN9			X						X										4	1	9	10
2461	MPLID			X						X										5	0	1	10
2471	MPLN1			X						X										5	1	1	10
2481	MPLN2			X						X										5	1	2	10
2491	MPLN3			X						X										5	1	3	10
2501	MPLN4			X						X										5	1	4	10
2511	MPLN5			X						X										5	1	5	10
2521	MPLN6			X						X										5	1	6	10
2531	MPLN7			X						X										5	1	7	10
2541	MPLN8			X						X										5	1	8	10
2551	MPLN9			X						X										5	1	9	10
2561	MRDID			X						X										1	0	1	10
2571	MRDN1			X						X										1	1	1	10
2581	MRDN2			X						X										1	1	2	10
2591	MSCAIC			X						X										9	0	1	10
2601	MSCAN1			X						X										9	1	1	10
2611	MSPRID			X						X										3	0	1	10
2621	MSPRN1			X						X										3	1	1	10
2631	MSPRN2			X						X										3	1	2	10
2641	NODAF		X							X										0	0	1	1
2651	NODIC		X							X										0	0	2	1
2661	PLALPHA		X							X										5	2	1	4
2671	PLA1		X							X										5	1	1	4
2681	PLA2		X							X										5	1	2	4
2691	PLA3		X							X										5	1	3	4
2701	PLA4		X							X										5	1	4	4
2711	PLBETA		X							X										5	2	2	4
2721	PLIC		X							X										5	0	1	4
2731	PLMAT		X							X										5	4	1	4
2741	PLN1		X							X										5	1	5	4
2751	PLN2		X							X										5	1	6	4
2761	PLN3		X							X										5	1	7	4
2771	PLN4		X							X										5	1	8	4
2781	PLSIGMAS1		X		X	X				X										5	3	1	6
2791	PLSIGMAS2		X		X	X				X										5	3	2	6
2801	PLSIGMA1		X		X	X				X										5	3	3	6
2811	PLSIGMA2		X		X	X				X										5	3	4	6
2821	PLSMS		X		X					X								X		5	5	1	9
2831	PLTAU12		X		X	X				X										5	3	5	6
2841	PLTEMP		X							X										5	4	2	4
2851	PLTMS		X		X					X								X		5	5	2	12
2861	PLTS(1)		X							X										5	2	3	4
2871	PLTS(2)		X							X										4	2	4	4
2881	PLT(0)		X							X											2	5	4
2891	RDA(1)		X							X											2	1	4
2901	RDA(2)		X							X											2	2	4
2911	RDIC		X							X										1	0	1	4
2921	RDMAT		X							X										1	4	1	4
2931	RDN1		X							X										1	1	1	4
2941	RDN2		X							X										1	1	2	4
2951	RDP		X		X	X				X										1	3	1	6
2961	RDP/A(1)		X		X	X				X										1	3	2	6
2971	RDP/A(2)		X		X	X				X										1	3	3	6
2981	RDSMS		X		X					X								X		4	5	1	9
2991	RDTEMP		X							X										1	4	2	4
3001	RDTMS		X		X					X								X		1	5	2	12
3011	RX		X		X	X				X										0	3	1	5
3021	RXBMODE		X		X	X	X			X										0	3	1	11
3031	RXVMODE		X		X		X			X										0	3	1	2
3041	RY		X		X	X				X										0	3	2	5
3051	RYAMODE		X		X	X	X			X										0	3	2	11
3061	RYVMODE		X		X		X			X										0	3	2	2
3071	RZ		X		X	X				X										0	3	3	5
3081	RZBMODE		X		X	X	X			X										0	3	3	11
3091	RZVMODE		X		X		X			X										0	3	3	2
3101	SCAA1		X							X										9	1	1	4
3111	SCAA2		X							X										9	1	2	4
3121	SCAF1		X		X	X				X										9	3	1	6
3131	SCAF2		X		X	X				X										9	3	2	6
3141	SCAF3		X		X	X				X										9	3	3	6
3151	SCAIC		X							X										9	0	1	4
3161	SCAR(R1)		X							X										9	2	1	4
3171	SCAR(R2)		X							X										9	2	2	4
3181	SCAR(R3)		X							X										9	2	3	4
3191	SCAR(T1)		X							X										9	2	4	4
3201	SCAR(T2)		X							X										9	2	5	4

Table 802-1. (Concluded)

NO.	ITEM NAME	ATTRIBUTE DEPENDENCY 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	ELEMENT NO.	KODE NO.	SEQUENCE NO.	MATRIX GROUP
3211	SCAK(13)		X							X										9	2	6	4
3221	SCAP1		X		X	X				X										9	3	4	6
3231	SCAP2		X		X	X				X										9	3	5	6
3241	SCAP3		X		X	X				X										9	3	6	6
3251	SCAR1		X							X										9	1	3	4
3261	SCASMS		X		X					X								X		9	5	1	9
3271	SCASMC		X		X					X								X		9	5	2	12
3281	SPLA1		X							X										11	1	1	4
3291	SPLA2		X							X										11	1	2	4
3301	SPLA3		X							X										11	1	3	4
3311	SPLA4		X							X										11	1	4	4
3321	SPLIC		X							X										11	0	1	4
3331	SPLPAT		X							X										11	4	1	4
3341	SPLA1		X							X										11	1	5	4
3351	SPLA2		X							X										11	1	6	4
3361	SPLA3		X							X										11	1	7	4
3371	SPLA4		X							X										11	1	8	4
3381	SPLC21		X		X	X				X										11	3	1	6
3391	SPLC23		X		X	X				X										11	3	2	6
3401	SPLC41		X		X	X				X										11	3	3	6
3411	SPLC43		X		X	X				X										11	3	4	6
3421	SPLC-EQUIV		X		X	X				X										11	3	5	6
3431	SPLSMS		X		X					X								X		11	5	1	9
3441	SPLT		X							X										11	2	1	4
3451	SPLTAL-MAX		X		X	X				X										11	3	6	6
3461	SPLTEMP		X							X										11	4	2	4
3471	SPLTMS		X		X					X								X		11	5	2	12
3481	SPLW-AVG		X		X	X				X										11	3	7	6
3491	SPRFAREA1L		X							X										3	2	1	4
3501	SPRFAREA1U		X							X										3	2	2	4
3511	SPRFAREA2L		X							X										3	2	3	4
3521	SPRFAREA2U		X							X										3	2	4	4
3531	SPRP-LMP1L		X							X										3	2	5	4
3541	SPRP-LMP1U		X							X										3	2	6	4
3551	SPRP-LMP2L		X							X										3	2	7	4
3561	SPRP-LMP2U		X							X										3	2	8	4
3571	SPRIC		X							X										3	0	1	4
3581	SPRMAT		X							X										3	4	1	4
3591	SPRA1		X							X										3	1	1	4
3601	SPRA2		X							X										3	1	2	4
3611	SPRC(3)L		X							X										3	2	9	4
3621	SPRC(3)U		X							X										3	2	10	4
3631	SPRC(2)L		X							X										3	2	11	4
3641	SPRC(2)U		X							X										3	2	12	4
3651	SPRP-CAPL		X		X	X				X										3	3	1	6
3661	SPRP-CAPU		X		X	X				X										3	3	2	6
3671	SPRP-LMPL		X		X	X				X										3	3	3	6
3681	SPRP-LMPU		X		X	X				X										3	3	4	6
3691	SPRC-EQUIV		X		X	X				X										3	3	5	6
3701	SPRSIGMA-L		X		X	X				X										3	3	6	6
3711	SPRSIGMA-U		X		X	X				X										3	3	7	6
3721	SPRSMS		X		X					X								X		3	5	1	9
3731	SPRTAL-MAX		X		X	X				X										3	3	8	6
3741	SPRTEMP		X							X										3	4	2	4
3751	SPRTMS		X		X					X								X		3	5	2	12
3761	SPRT-WER		X							X										3	2	13	4
3771	SRDA(1)		X							X										10	2	1	4
3781	SRDA(2)		X							X										10	2	2	4
3791	SRDIC		X							X										10	0	1	4
3801	SRDMAT		X							X										10	4	1	4
3811	SRDA1		X							X										10	1	1	4
3821	SRDA2		X							X										10	1	2	4
3831	SRDA3		X							X										10	1	3	4
3841	SRDP1		X		X	X				X										10	3	1	6
3851	SRDP2		X		X	X				X										10	3	2	6
3861	SRDP21/L		X		X	X				X										10	3	3	6
3871	SRDSMS		X		X					X								X		10	5	1	9
3881	SRDTEMP		X							X										10	4	2	4
3891	SRDTMS		X		X					X								X		10	5	2	12
3901	SRD(P/A)1		X		X	X				X										10	3	4	6
3911	SRD(P/A)2		X		X	X				X										10	3	5	6
3921	TA		X		X	X				X										0	3	4	5
3931	TABRCDE		X		X	X	X			X										0	3	4	11
3941	TAVMCDE		X		X		X			X										0	3	5	2
3951	TV		X		X	X				X										0	3	5	5
3961	TVBMCDE		X		X	X	X			X										0	3	5	11
3971	TVVMCDE		X		X		X			X										0	3	6	5
3981	TZ		X		X	X				X										0	3	6	11
3991	TZBRLDF		X		X	X	X			X										0	3	6	11
4001	TZVMDCE		X		X		X			X										0	3	6	11
4011	[illegible]		X		X					X										0	4	1	8
4021	[illegible]							X	X						X	X	X			0	0	1	1
4031	[illegible]		X							X										0	0	1	1
4041	[illegible]		X							X										0	1	2	1
4051	[illegible]		X							X										0	1	3	1

803. GRAPHICS POSTPROCESSOR

803.1 GENERAL INFORMATION

803.1.1 Purpose

To generate graphical displays using the extracted data residing on the file EXTRRNF, generated by the EXTRACT processor.

The GRAPHICS Postprocessor produces online and/or offline two- and three- dimensional windowing, rotation, two-dimensional clipping, and plot selection. Display types include: orthographic projection, contour, matrices in numerical form with "floating window," and execution menus.

Batch and/or interactive modes are provided to produce optional CØMp80, CALCØMP or GERBER offline plots while optional online graphical displays are generated on a Tektronix 4014 CRT.

The processor makes extensive use of the following software packages:

- BCS Interactive Graphics library.
- BCS Numerical Plotting System;
- Tektronix PLOT-10 Terminal Control System.

803.1.2 Access

This module is called from the ATLAS control program by the EXECUTE GRAPHICS statement (sec. 228, ref. 1-1)

803.2 MATRIX ACTIVITY

	Name	Description	File
Input			
1)	ADATDIR	Copy of ATLAS Data Directory	EXTRRNF
2)	DBEXCØN	Extract control matrix	EXTRRNF
3)	DBEXTNM	Extract name List	EXTRRNF
4)	DBINDEX	Attribute name list matrix	EXTRRNF
5)	DB001xx	Extracted data matrix	EXTRRNF
6)	DBINDxy	Low index list matrix	EXTRRNF
7)	ALNMLST	Altitude subset list matrix	EXTRRNF
8)	ALNm001	Altitude subset matrix	EXTRRNF
9)	CANMLST	Flutter case subset list matrix	EXTRRNF
10)	CANM001	Flutter case subset matrix	EXTRRNF

11)	CSNMLST	Changeset subset list matrix	EXTkkNF
12)	CSNM001	Changeset subset matrix	EXTRRNF
13)	CØNMLST	Condition subset list matrix	EXTRRNF
14)	CØNM001	Condition subset matrix	EXTRRNF
15)	CYNMLST	Cycle subset list matrix	EXTRRNF
16)	CYNM001	Cycle subset matrix	EXTRRNF
17)	LCNMLST	Loadcase subset list matrix	EXTRRNF
18)	LCNM001	Loadcase subset matrix	EXTRRNF
19)	MDNMLST	Mode subset list matrix	EXTRRNF
20)	MDNM001	Mode subset matrix	EXTRRNF
21)	RSNMLST	Retained subset list matrix	EXTRRNF
22)	RSNM001	Retained subset matrix	EXTRRNF
23)	SUBSLST	Node/element subset list matrix	EXTRRNF
24)	SPKddda	Ordered nodal subset matrix	EXTRRNF
25)	SDTNLST	Label subset list matrix	EXTRRNF
26)	SITM001	Label subset matrix	EXTRRNF

Output

1) The Graphics File - GRAPFIL

The graphics file, G/F, consists of user defined plot attributes input via ATLAS GRAPHICS statement(s) for generating on-screen and/or offline graphical displays. G/F consists of two sequential binary files, GRAFSQ1 and GRAFSQ2, written by subroutines WGRAH, WGRAD and optionally WGRAT under the control of the GRAPHICS (1,0) primary overlay. Records contained on G/F are written with the following write statements:

GRAFSQ1

```
WRITE (GRAFSQ1) NWX(IHD(I),I=1,NWX),NWY,
      (XIN(I),I=1,NWY),NWCL,(XCLVL(I),I=1,NWCL)
```

GRAFSQ1 consists of one data record per graphical display, while GRAFSQ2 consists of one data file (e.g., K data records followed by an EØF mark) for each record written on GRAFSQ1.

GRAFSQ2

Record Type One

```
WRITE (GRAFSQ2) KØD,LNGH,(IBUF(I),I=1,LNGH),NP,IDIMN,
      ((XYZ(I,J),I=1,NP),J=1,IDIMN),JKØDE,NL,LLL,
      ((LBLX(I,J),I=1,NL),J=1,LLL)
```

N records of this type are written for each graphical display, the last being indicated by setting KØD=3. Subroutine WGRAD writes these records, where $1 \leq N$.

Record Type Two - Optional

The second type is optional, depending upon whether special annotation is requested by the user via GRAPHICS statement(s), and is written by subroutine WGRAT.

```
WRITE (GRAFSQ2) KØD,NS,LØC,(JTEXT(I),I=1,LØC),J,
      (JCHAR(I),I=1,J) (CHSIZE(I),I=1,J),((XYPT(K,I),I=1,J),
      K=1,IDIM),(JWØRDS(I),I=1,J),(THETA(I),I=1,J)
```

M records of this type may be written for each graphical display, depending on user directives where 0 ≤ M. If records of this type are written for a display, then the Mth record is signalled by setting KØD=±3. Each display is terminated by an EØF mark being written on GRAFSQ2.

The contents of data records written on G/F by subroutines WGRAH, WGRAD and WGRAT is described in section 1001.

2) Device Independent Vector File - VECTFIL

V/F is a sequential binary file containing offline device (e.g., CALCØMP) independent commands. V/F is used by a user selected postprocessor as an input file to generate offline dependent commands for the CØMp80, CALCØMP or GERBER plotter. V/F is written by BIGS lower level routines, QXGRPT and QXHDR.

V/F is written in the following format:

```
WRITE (97) IR,NW,NC,NCHSZ,(ITEXT(KK),KK=1,NW),
      NPTS,(XX(J),YY(J),J=1,NPTS)
```

The file is terminated with an EØI.

803.3 PROGRAM METHOD

The GRAPHICS module consists of 2 primary overlays. The function of the first primary overlay is to access the information on EXTRFNF and generate the sequential files GRAFSQ1 and GRAFSQ2 (or G/F) according to the user specifications supplied in the EXECUTE GRAPHICS command.

The function of the second primary overlay is to interpret the contents of G/F, generate a device-independent vector file (V/F), generate online displays, and create a device-dependent plotting data file (P/F) for any of the 3 offline plotting devices e.g., CALCØMP, GERBER and CØMp80.

The purpose of the two primary overlays is to cut down on the core size requirements. The arrangement works as follows: On encountering an EXECUTE GRAPHICS command, the (0,0) overlay calls the (1,0) overlay from GRAPØLF in the normal manner. The (1,0) overlay interprets the contents of /CØNPARS/, generates the G/F file, and determines whether any displays are to be produced and if a V/F file is to be created. If neither of these is required, the (1,0) returns directly to (0,0). Otherwise it calls (2,0) itself. After completing the desired tasks the (2.0) returns to (0,0) and the ATLAS program execution continues normally.

The (1,0) primary overlay, GFGEN contains the eight secondary overlays viz; GRCØN,BALPLØT,GENMAT,GEØMPLT, NVECPLT,ESCØPLT,XYPLØT, and MTRXPLT.

803.3.1 ØVERLAY (GRAPØLF,1,0) - GFGEN

The function of GFGEN is to interpret the parameters of the EXECUTE GRAPHICS statement and generate the graphics file GRAPFIL which contains the basic information required to generate displays by the (2,0) primary overlay and its secondaries. The program reads the data from the random access file EXTRRNF, generated by the EXTRACT processor, and writes the same on the sequential file GRAPFIL using the subroutines WGRAH, WGRAD and WGRAT.

The file EXTRRNF contains several sections, each pertaining to an EXECUTE EXTRACT command and identified by the 'EXNAME' supplied by the user in his EXECUTE EXTRACT command. These EXNAME's are refe[illegible] to by the user in his EXECUTE GRAPHICS command implying that the data required to generate the specified displays is contained in that section of EXTRRNF which is identified by the EXNAME. One EXECUTE GRAPHICS command may refer to more than one EXNAME indicating more than one section of EXTRRNF can be used in generating displays.

Program GFGEN calls the overlay (1,21) which interprets the contents of the EXECUTE GRAPHICS command and determines the number of EXNAME's that are specified as well as the type of display that is to be generated from data related to each EXNAME. The value of the variable NUMPLT indicates the type of display to be generated and has the following significance.

NUMPLT = 1 - 3D orthographic plots of structural geometry.

NUMPLT = 2 - 3D orthographic plots of nodal displacements.

NUMPLT = 3 - 3D orthographic plots of scalar quantities related to finite elements.

NUMPLT = 4 - Contour plots of scalar quantities related to finite elements.

NUMPLT = 5 - V-G/V-F graph plots.

NUMPLT = 6 - Other graph plots

NUMPLT = 7 - Matrix plots

A separate secondary overlay is used to generate plots of type 1, 2 and 7, whereas one secondary overlay generates plots of types 3 and 4, and one generates plots of types 5 and 6.

The program GFGEN sets up a loop for the number of EXNAME's referred to in the EXECUTE GRAPHICS command. Using the value of NUMPLT associated with each of the EXNAME's it calls the required overlay to generate the desired display.

The generation of "loadability diagrams" is carried out entirely by the secondary overlay (1,20), program BALPLØT. The program BALPLØT also interprets the contents of the EXECUTE GRAPHICS statement. The indicator for this path is set in the variable LØDIND.

After the GRAPFIL is completely written the program writes four words on sequential file IDIRECT. The contents of these words are used by the primary overlay (2,0) program AGP to determine subsequent actions. The possible subsequent actions are as follows:

a) In interactive mode

- generate online displays only
- generate online displays and write them on VECTFIL
- Write the contents of GRAPFIL onto VECTFIL without display

- Write contents of VECTFIL onto TAPE99 for an offline plotting device.

b) In batch mode

- Write contents of GRAPFIL onto VECTFIL
- Write contents of VECTFIL onto TAPE99 for an offline plotting device.

If none of these actions is required, the (2,0) overlay is not called.

The following contains a step-by-step description of the tasks performed by GFGEN.

a) Opens and positions the two sequential files of the graphics file, GRAPFIL.

b) Initializes common blocks /GRMØDCM/ and /C1/

c) Opens the random files EXTRRNF and SC00RNF.

d) Calls overlay (1,2) which interprets the contents of /CØNPARS/

e) Calls overlay (1,20) which generates loadability plots if so indicated by the value of the variable LØDIND. If (1,20) is called, steps (f) thru (h) are omitted.

f) Checks the value of the variable IEXTDEF which indicates the total number of EXNAME's from which data is to be displayed. Omit steps (g) and (h) if IEXTDEF=0.

g) Calls overlay (1,21) which sets up matrices from the extracted data, if NUMPLT = 1, 2, 3 or 4. The matrices contain all the information that is required to produce all displays.

h) Sets up a loop 1 to IEXTDEF. Every pass of the loop checks the value of the variable NUMPLT and calls the associated overlay. The first value in NUMPLT is set up when the overlay (1,21) is called. Each secondary overlay called subsequently resets NUMPLT as required for the next cycle, before returning to (1,0).

i) Writes the variables IBATCH,ISWØFF,IDEVC and NPLØT on the file IDRCT.

j) Closes all files.

k) Returns if ICALL=0.

l) Otherwise calls Overlay (GRAPØLF,2,0)

803.3.2 ØVERLAY(GRAPØLF,1,2) - GRCØN

a) Sets up default plot parameters in common block /GRMØDCM/.

b) Checks if loadability plots are to be generated. If so, sets LØDIND=1, moves the contents of the matrix DBINDXX for the specified EXTRACT to array LØDPØS in common block /GRMØDCM/ and returns.

c) Interprets sequentially the contents of the parameter list in the EXECUTE GRAPHICS statement. The values are stored in the array MP in the common block /GRMØDCM/. Whenever an EXNAME is encountered, the contents of MP(1) thru MP(43) provide the specification for the displays to be generated from the data associated with the EXNAME. These are transferred to a column of the matrix with index name EXNMLST. After the command is completely interrogated the EXNMLST matrix is written on the scratch file SC00RNF. The matrix EXNMLST contains as many columns as the EXNAME's referred to the EXECUTE GRAPHICS command.

d) Writes the EXNMLST matrix on SC00RNF.

803.3.3 ØVERLAY GRAPØIF(1,20) - BALPLØT

Program BALPLØT directs the formation of loadability plots. These consist of a fan grid of weight vs vehicle cg-x in percent Mean Aerodynamic Chord, overlayed by passenger, cargo, and fuel vectors. Subroutine CHKDATA looks through CØNPARS for keywords, and sets values accordingly. GRID forms the basic fangrid, using various graphics utility routines, while VECTØR takes appropriate matrices from EXTRRNF and calculates the passenger, cargo and fuel vectors as required using various graphics utility routines.

803.3.4 ØVERLAY(GRAPØLF,1,21) - GENMAT

The program GENMAT generates certain matrices which facilitate generation of displays by subsequent overlays. The matrices, as identified later in this

section, are generated for information that is needed several times by subsequent overlays.

This avoids multiple accessing by keys of the non-contiguously stored data items. The data items are accessed only once by the keys and the desired information is then available as self contained matrices. These matrices are created for each of the EXNAME's referred to in the EXECUTE GRAPHICS command. The characters 6 and 7 in the index names for these matrices contain the integer number in display code associated with the EXNAME.

Step-by-step tasks performed by the program are as follows:

a) Reads the following matrices:

DBINDEX - EXTRRNF
DBEXTNM - EXTRRNF
SUBSLST - EXTRRNF
DTNMLST - EXTRRNF
DBEXCØN - EXTRRNF
EXNMLST - SC00RNF

b) Sets up a loop for number of columns in the EXNMLST matrix.

Finds plottype (2nd variable in the column) in NUMPLT.

If NUMPLT is greater than 4, omit steps (c) thru (g). Carries out steps (c) thru (g) for each pass in the loop.

c) Obtains the associated attribute information by calling the subroutine INDXUSE. The subroutine also obtains the number of nodes in NUMNØDS, the number of elements of each type in array NUMELMS, total number of elements in NØELT, the selection pattern for node-related items in array ISELPNØ and the same for element related items in array ISELPEL.

d) Sets elements 50 thru 57 of the current column of the EXNMLST matrix.

e) Calls GENØSØF to generate the following matrices:

VEDF001 - Each word in this matrix contains a pair of 30-bit integers

that are internal node numbers between which a line is to be drawn when generating a grid.

ELND001 - Each word in this matrix contains up to 5 packed 12-bit integers which are the structural nodes for an element. As many words per element are written as required.

The matrix in position MNØDSS is a nodal subset matrix for the element structural nodes for all elements in the data for the EXNAME.

f) Calls GENDUSS to generate the following two subset matrices.

The matrix in position MAF is the subset of nodes for which data is extracted, with respect to the internal nodes.

The matrix in position MNID is the subset of element nodes with respect to the nodes for which data is extracted.

g) Calls GEDUCØT twice to generate nodal correspondence tables for nodes represented by the subset matrices MAF and MNID.

A nodal correspondence table is essentially a renumbering scheme whereby the original node numbers that may not be contiguous are renumbered contiguously from 1 to N where N is the number of nodes. The nodal correspondence table contains information for obtaining both old-new and new-old correspondence.

For the MAF matrix the internal ids of the nodes are regarded as user IDs and a new sequence of internal IDs is generated and incorporated in the first correspondence table in the position MNCT. This scheme is used to get a compact connectivity matrix.

For the MNID matrix the internal IDs of nodes are regarded as user IDs and a new sequence of internal IDs is generated and incorporated in the second correspondence table in position MNCT1. This table helps in locating a given node in the extracted data.

h) Calls SELCØRD to obtain the nodal coordinates of the nodes in the extracted data and puts them in matrices at positions MXG, MYG, MZG, respectively.

i) The coordinates, as obtained by calling SELCØRD, are in the global coordinate system. The user specification in the EXECUTE GRAPHICS command and/or the default specification for the desired views may require that these coordinates be transformed. For this purpose the program calls GETTRIN to obtain the transformation specifications for the data and calls UTRANS to perform the transformations. The coordinates in positions MXG, MYG and MZG are transformed and stored in positions MXT, MYT and MZT.

j) If NUMPLT=3, the transformed coordinates are projected onto the Y-Z plane by zeroing out the contents of the matrix at MXT.

k) If NUMPLT=4, the transformed coordinates are projected onto the X-Y plane by zeroing out the contents of the matrix at MZT.

l) Calls GENCØNM to generate the connectivity matrix. This is an upper triangular bit form matrix. The bits that are turned on in a row indicate the connection of the node represented by that row to the other nodes.

m) Calls GENMAF to generate the matrix of analysis frames and a matrix of user IDs for the nodes in the extracted data.

n) Calls GENMEID to generate matrix of user IDs for the elements in the extracted data.

o) Calls GENECG to generate a matrix of element c.g. locations (based on structural nodes) for the elements in the extracted data.

p) Writes the local frames matrix on SCOORNF.

q) Sets up a loop for the numbers of EXNAME's. For each EXNAME the minimum and maximum values of transformed X, Y and Z coordinates are obtained.

If SCALE is specified as a parameter in the EXECUTE GRAPHICS statement, the coordinate matrices, the element c.g. location matrix and the min./max. values are scaled. The minimum and the maximum

values are put into elements 44 thru 49 of the current column of the EXNMLST matrix.

Steps (c) thru (q) result in writing on SCOORNF the following matrices for each EXNAME, from which data is to be displayed.

r) If the EXPLØDE option is specified by the user in his EXECUTE GRAPHICS command, it implies that the data from all the EXNAME's are to be displayed together in one single display. In that case the maximum and the minimum coordinate values are updated to reflect the maximum and the minimums of the combined total display.

s) Loads the first column of the EXNMLST matrix into array MP in the common block /GRMØDCM/ so that the primary overlay program GFGEN calls the next proper secondary overlay.

Index Name	Position Name	Contents
XGLØBXX	MXG	Global X-coordinates
YGLØBXX	MYG	Global Y-coordinates
ZGLØBXX	MZG	Global Z-coordinates
XTRANXX	MXT	Transformed X-coordinate
YTRANXX	MYT	Transformed Y-coordinate
ZTRANXX	MZT	Transformed Z-coordinate
AFRAMXX	MAF	Nodal analysis frames
MNØIDXX	MNID	Nodal User IDs
MELIDXX	MEID	Element user IDs
DCØRTXX	MNCT	First nodal correspondence table
MCØNNXX	MCØN	Connectivity matrix
MECGTXX	MECGT	Transformed element c.g. matrix
MLØCØXX	MLØCAL	Local reference frame table
DCØR1XX	NMCT1	Second nodal correspondence table

These matrices contain all the information required to generate a labeled grid and are used for that purpose by subsequent overlays.

803.3.5 ØVERLAY (GRAPØLF,1,22) - GEØMPLT

This overlay plots geometry.

a) Reads matrices DBINDEX and DEEXCØN from EXTRRNF.

b) Calls INDXUSE to obtain attribute usage in the array IINDEX. Numbers of elements of each type in array NUMELMS, number of nodes in NUMNØDS, total number of elements in NØELT, selection pattern for node-dependent items in array ISELPNØ and the same for element dependent items in array ISELPEL.

c) Calls GETMAT to read appropriate matrices from SC00RNF.

d) Calls WGRAH to write the header on G/F.

e) Calls GENTRAC to generate grid if required or calls GENPØIN to generate points only.

f) Calls GENDZ to generate dZ coordinates.

g) Calls GENNØID to generate node labels if required.

h) Calls GENELED to generate element labels if required.

i) Reads the EXNMLST matrix and loads the next column of the matrix into the array MP in common block /GRMØDCM/ so that the primary overlay program GFGEN calls the next proper secondary overlay.

803.3.6 ØVERLAY (GRAPØLF,1,23) - NVECPLT

This overlay plots nodal displacements for loadcases and vibration and buckling mode shapes.

a) Reads matrices DBINDEX, DBEXCØN and DBEXTNM from EXTRRNF.

b) Calls INDXUSE (see step (b) in overlay (1,22))

c) Calls GETMAT to read appropriate matrices from SC00RNF.

d) Reads matrix ADATDIR (ATLAS DATA DIRECTØRY) from EXTRRNF and determines the vector components that are to be plotted.

e) Determines whether the vector components are dependent on loadcases of mode shapes and finds out the index name of the IC/MØDE subset matrix that is associated with the data.

f) Reads the appropriate LC/MØDE subset matrix.

g) Generates the "INDEX" that is associated with the first vector to be plotted and calls GETREC to locate the key in the extracted data.

h) Calls GENNVEC to generate the displacement plots.

i) Loads the next column of EXNMLST matrix into the array MP in common block /GRMØDCM/ so that the primary overlay program GFGEN calls the next proper secondary overlay.

803.3.7 ØVERLAY (GRAPØLF,1,24) - ESCØPLT

This overlay plots stress components, property values or margins of safety associated with finite elements as 3-D orthographic plots or contour plots.

a) Reads matrices DBINDEX and DBEXCØN from EXTRRNF.

b) Calls INDXUSE (see step (b) in overlay (1,22))

c) Calls GETMAT to read appropriate matrices from SCØØRNF.

d) Reads matrix ADATDIR (ATLAS DATA DIRECTØRY) from EXTRRNF and determines the scalar quantity that is to be plotted.

e) If the scalar quantity is dependent on loadcases/cycles, the index name for the loadcase/cycle subset matrix is obtained and the loadcase/cycle subset matrix is read.

f) Calls GETREC to locate the first key for the scalar in the extracted data.

g) Calls GENESCA to generate 3-D scalar plots or,

h) Calls GENECØ to generate contour plots.

i) Loads the next column of EXNMIST matrix into the array MP in common block /GRMØDCM/so that the primary overlay program GFGEN calls the next proper secondary overlay.

803.3.8 ØVERLAY (GRAPØLF,1,25) - XYPLØT

This overlay plots V/G-V/F curves or scalar vs. loadcase or scalar vs. cycle graphs. For V/G-V/F curves:

a) Reads the first extracted data matrix.

b) Calls GENVGVF to generate plots.

The (2,0) primary overlay, AGP, has 13 secondary overlays viz. MENUEX, PLTXT2D, PLTXT, MTRX, CPLTXT, CPLTXT2, SC4020, CALCMP, GERBER, HCMENU, GDRCTRY, CNTUR2, and CNTUP3 associated with it. The purpose of the (2,0) and its secondary overlays is given below.

For scalar vs. loadcase/cycle graphs.

c) Reads matrices DBINDEX and DBEXCØN from EXTRRNF.

d) Calls INDXUSE (see step (b) in overlay (1,22))

e) Reads matrix ADATDIR (ATLAS DATA DIRECTØRY) from EXTRRNF.

f) Determines the attribute dependency of the scalar quantity to be plotted and checks that the quantity is related either to loadcases or design cycles.

g) Reads the loadcase/cycle subset matrix.

h) Calls subroutine SGRØDAT to have the required data selected and reordered so that the extracted data is stored by elements with results for all loadcases/cycles being stored sequentially for each element. The routine SGRØDAT is the same as used by overlay (EXTRØLF,1,4), program RESØRT in the EXTRACT processor.

i) Calls subroutine BARPLØT to generate graphs.

j) Loads the next column of EXNMLST matrix into the array MP in common block /GRMØDCM/ so that the primary overlay GPGEN calls the next proper secondary overlay.

803.3.9 ØVERLAY (GRAPØLF,1,26) - MTRXPLT

a) Reads matrix MDBINDX from EXTRRNF.

b) Sets up a loop for the entries of the matrix MDBINDX and calls subroutine GENMATR to generate plot of each matrix.

c) Loads the next column of EXNMLST matrix into the array MP in common block /GRMØDCM/ so that the primary overlay GFGEN calls the proper secondary overlay next.

803.3.10 ØVERLAY (GRAPØLF,2,6) - AGP

Program AGP monitors execution of the appropriate secondary overlays under it using the information supplied by the user in his EXECUTE GRAPHICS statement and that supplied via the keyboard during interactive operation.

The functions performed by the individual secondary overlays are described in the following sections.

803.3.11 ØVERLAY(GRAPØLF,2,1) - MENUEX

Program MENUEX is responsible for all menus pertaining to online displays and keyboard activities, excluding the crosshairs (graphic cursor), and is invoked only in the conversational mode. Specific functions include: deciphering user inputs entered via the keyboard for legal commands and/or data, displaying appropriate menus requested by the console operator, updating transformation parameters (e.g., zoom), and passing user directives ot the graphics executive, AGP, via labeled common (e.g., plot selection).

MENUEX, the menu executive, controls the display and functional process of the following menus and directories:

- Function Menu,
- Gname Directory,
- Plot ID Directory,
- Plot Transformation Menu.

Refer to reference 1-1 for a detailed explanation of these menus and the command structure required by each.

803.3.12 ØVERLAY(GRAPØLF,2,4) - PLTXT2D

Program PLTXT2D is called by AGP to generate two-dimensional graphical displays, including two-dimensional multiple projections. This overlay displays one, two, three or four displays on a single "frame." Graphical inputs are read off the G/F initially. Plot limits may be superseded online using the crosshairs or the plot transformation menu. The following features are supported via PLTXT2D:

- Plotting of any number of points or curves on a single plot;
- Superposition of text plotted in alphanumeric and/or vector form;
- Optional generation of axes or grid;
- Two-dimensional windowing via crosshairs for single plot per frame;
- Comprehensive array of plotting styles;
- Automatic generation of a vector file unless supressed by user;
- Clipping.

803.3.13 ØVERLAY(GRAPØLF,2,5) - PLTXT

Program PLTXT is called by AGP to generate either orthographic or perspective graphical displays. Graphical inputs are read off the graphics file initially. The display may be altered via the crosshairs or one of the transformation menus. The following features are supported by PLTXT:

- Transformation options such as windowing and rotation;
- Plotting of any number of points or curves on a single plot;
- Superposition of text displayed in alphanumeric and/or vector form;
- Optimum subject space computation option prior to displaying rotated object;

- Clipping;
- Order of rotation user-controlled;
- Automatic generation of vector file unless suppressed by user.

803.3.14 ØVERLAY(GRAPØLF,,,6) - MTRX

Program MTRX displays a user selected matrix in numerical form. MTRX is only called in the conversational mode. Matrices in numerical form are not produced offline. A key is printed on the CRT identifying the row and column number of the first element in each record read off G/F. Each element of a "page" is identified by column and row numbers. The user may view portions of a matrix not shown in the current "page" using a "floating window" option if the data is contained within the current data record. An option is also provided to skip to the next record if the matrix contains multiple records.

803.3.15 ØVERLAY(GRAPØLF,2,7) - CPLTXT

Program CPLTXT is called when a contour plot is to be generated using the following algorithm:

> Given a set of random x, y, z points, an M*M surface definition matrix of uniformly spaced z-values is generated over the x-y plane using an octant weighted average for each mesh point. Each rectangular grid element is examined to determine if any of the supplied control levels are contained with it. If so, an arithmetic average is used to compute a fifth point to subdivide the rectangle into 4 separate triangles. Linear interpolation is then used to compute isobar lines for those triangles containing contour level(s) within it. The set of random x, y, z points is not central memory bounded.

The primary purpose of CPLTXT is to compute the M*M surface definition matrix of uniformly spaced z-values as described in the above algorithm. The surface definition matrix is computed by CPLTXT, saved in labeled common, and utilized by ØVERLAY(GRAPØLF,2,10) to compute either a contour plot or orthographic projection both of which utilizes the surface definition matrix. The surface definition matrix is calculated only once after a user has selected a specific plot from the Gname directory even though several different graphical

displays (views) are generated by the user via transformation options (e.g., zoom, rotation).

803.3.16 ØVERLAY(GRAPØLF,2,10) - CPLTXT2

Program CPLTXT2 using the M*M matrix of uniformly spaced z-values computed by ØVERLAY(GRAPØLF,2,7), computes the isobar lines using the following method: Each rectangular grid element is examined separately to determine if any of the supplied contour levels is contained within it. If so, an arithmetic average is used to compute a fifth point to subdivide the rectangle into 4 separate triangles. Each triangle is examined separately to determine if contour levels are contained within it and linear interpolation is used to compute the isobar lines. The process is repeated for each rectangular grid element.

CPLTXT2 also displays the M*M surface definition matrix as a three-dimensional orthographic projection of the surface grid. The user is allowed to interactively define the orientation and scaling information of the display. The following method is used to generate the 3D display: 3D lines are drawn through each of the M rows and M columns of the matrix. Each grid point in a line is defined in x, y, z space by its row, column and matrix value. Scaling in the x, y dimensions is normalized to 30% of unity. The orientation is preset to RX = 5 degrees, RY = 15 degrees and RZ = 30 degrees.

803.3.17 ØVERLAY(GRAPØLF,2,12) - SC4020

Program SC4020 is the Stromberg Carlson 4020 postprocessor. The postprocessor is used to convert device independent commands written on the vector file (TAPE97) to the SC4020 or CØMp80 dependent commands written on TAPE99. The postprocessor utilizes the Boeing Numerical Plotting System (NPS) in the conversion process. The postprocessor also generates a file called ØFFLINE which contains directives for the CØMp80 plotter operator. The contents written on the ØFFLINE file are obtained either via EXECUTE GRAPHICS statement(s) and/or generated by user directives defined via the Hardcopy Questionnaire Menu (refer to ØVERLAY(GRAPØLF,2,15)). Both TAPE97 and ØFFLINE are used for queuing data during plot postprocessing.

803.3.18 ØVERLAY(GRAPØLF,2,13) - CALCMP

Program CALCMP is the CALCØMP model 763 postprocessor. The postprocessor is used to convert device independent

commands written on the vector file (TAPE97) to CALCØMP dependent commands written on TAPE99. The postprocessor utilizes extensively the Boeing Numerical Plotting System (NPS) in the conversion process. The postprocessor also generates a file called ØFFLINE which contains directives for the CALCØMP operator (e.g., bond or vellum). The contents written on the ØFFLINE file are obtained either via EXECUTE GRAPHICS statement(s) and/or generated by user directives defined via the Hardcopy Questionnaire Menu (refer to ØVERLAY(GRAPØLF,2,15)). Both TAPE97 and ØFFLINE are used for queuing data during plot postprocessing.

803.3.19 ØVERLAY(GRAPØLF,2,14) - GERBER

Program GERBER is the GERBER drafting-machine postprocessor. The postprocessor is used to convert device independent commands written on the vector file (TAPE97) to GERBER drafting-machine dependent commands written on TAPE99. The postprocessor utilizes extensively the Boeing Numerical Plotting System (NPS) in the conversion process. The postprocessor also generates a file called ØFFLINE which contains directives for the GERBER operator (e.g., pen type). The contents written on the ØFFLINE file are obtained either via EXECUTE GRAPHICS statement(s) and/or generated by user directives defined via the Hardcopy Questionnaire Menu (refer to ØVERLAY(GRAPØLF,2,15)). Both TAPE97 and ØFFLINE are used for queuing data during plot postprocessing.

803.3.20 ØVERLAY(GRAPØLF,2,15) - HCMENU

Program HCMENU only displays the Hardcopy Questionnaire Menu when running in the conversational mode. The menu allows the user to define the following information interactively:

- Offline plotting device (e.g., CØMp80, CALCØ...);
- Mailing and operator directives;
- Plot stacking option.

This information is transmitted to the appropriate offline postprocessor (e.g., CALCØMP) via labeled common.

The Hardcopy Questionnaire Menu is not currently used.

803.3.21 ØVERLAY(GRAPØLF,2,30) - GDRCTRY

Program GDRCTRY's primary function is to build a directory of the graphics file, G/F. The directory is used to display the Gname Directory and Plot ID Directory. The directory provides a summary of the graphical displays written on G/F and sufficient information to allow efficient references to any display selected by the user via the Gname Directory and/or Plot ID Directory. The directory also contains the plot type (e.g., contour) which is used to determine which overlay(s) is to be called to generate the display.

The directory is written on file IDRCT, one binary record per subject index (e.g., V/G - V/G).

803.3.22 ØVERLAY(GRAPØLF,2,31) - CNTUR2

Program CNTUR2 is called when a contour plot is to be generated using the surface triangularization algorithm and associated software described in reference 1-1 to compute the connectivity data given a set of boundary points and an arbitrary number of data points. The total number of points should not exceed 1000 and the total number of resulting triangles not exceed 1600.

The surface triangularization algorithm is stated below:

1. Start the triangularization using any two adjacent boundary points i and j.

2. Determine the point k which yields the largest angle between ik and jk.

3. Define the first triangle as ijk and set a flag indicating whether k is to the left or right of the line ij.

4. Check whether the side ik is on the boundary or the side of a previously generated triangle. If so, go to step 5, if not, proceed as follows:

   a. Find the point l which yields the largest angle between il and lk and is on the proper side of ik. If k is to the left of ij, l must be to the left of ik.

b. If the side li is part of a previous triangle lim, set l=m and repeat the test for li.

c. If the side kl is part of a previous triangle klm, set l=m and go to step 4b.

d. If neither condition 4b or 4c is met, define ikl as the next triangle in the subdivision.

5. Repeat step 4 for the side kj. It is important that the order ik and kj be used (rather than ki of jk) to generate triangles which all have the same rotational sense as ijk.

6. Find all adjacent triangles for the next triangle in the list. If the nodes of the next triangle are labeled ijk, step 4 must be repeated for each side ji, kj and ik.

7. Repeat step 6 for each triangle in the subdivision.

This algorithm will normally result in a complete set of adjacent triangles whose union consists of the entire region when completed, each side of each triangle will either

1. be on the boundary of the region,

2. be a common side between two adjacent triangles, or

3. have no neighboring point which will form another triangle with the correct rotational sense. This condition is considered an error in the program, but the program will continue to process the triangles which have been found.

The connectivity data is passed to ØVERLAY(GRAPØLF,2,32) for further processing.

803.3.23 ØVERLAY(GRAPØLF,2,32) - CNTUR3

Program CNTUR3 is automatically called when the surface triangularization algorithm is to be used to generate a contour plot (refer to ØVERLAY(GRAPØLF,2,31)).

When this overlay is called two different graphs may be generated.

1. Contour plot generated from the surface triangularization computed by program CNTUR2 using boundary points and an arbitrary set of interior points written on G/F. Contour lines are computed using linear interpolation.

2. Optional plot showing the surface triangularization computed by CNTUR2.

803.4 COMMON BLOCK USAGE

	ØVERLAY: GRAPØLF (1,X)							
NAME	(1,0)	(1,2)	(1,20)	(1,21)	(1,22)	(1,23)	(1,24)	(1,25)
KERRØR	x	x	x	x	x	x	x	x
CØNPARS	x	x	x	x	x	x	x	x
KØRNDM	x	x	x	x	x	x	x	x
GRMØDCM	x	x	x	x	x	x	x	x
FILE99	x		x		x	x	x	x
WGRADX	x		x		x	x	x	x
WGRATX	x		x		x	x	x	x
CHCKGF	x		x		x	x	x	x
XCLVL	x							
IØBUF9B	x							
IØBUF9	x		x		x	x	x	x
RØRDABG	x							
C1	x			x	x	x	x	x
PLTSTUF			x					
LABEL9			x		x	x	x	x
TEXT99			x		x	x	x	x
HDR999			x		x	x	x	x
PLTBUF			x		x	x	x	x
PLTCØM				x	x	x	x	x

NAME	ØVERLAY: GRAPØLF(2,X)							
	(2,0)	(2,1)	(2,4)	(2,5)	(2,6)	(2,7)	(2,10)	(2,12)
KERRØR	x							
FILE99	x	x	x	x	x	x	x	
ØFFLNE	x							x
XCLVL	x		x	x	x	x	x	
NPSCØM								x
ZMCØNTR	x					x	x	x
QXGDØA	x	x	x	x	x		x	
QXSCCØ	x		x					
QXCØRD	x	x	x	x	x	x	x	
CØNLIM	x					x	x	
CØNLIM2	x					x	x	
FETBUFF	x							
IØBUF9B	x							
IØBUF9	x							
ITYPLT	x	x						
HDR999	x	x	x	x	x	x	x	
NUMEL99	x							
HCDATA	x							
BATCHM	x							
KEEPST	x		x	x		x	x	
XBLWUP	x		x	x		x	x	
RØRDABG	x	x	x	x	x	x	x	
CALCST	x		x	x		x		
STLMTS	x		x	x	x	x		
ØFFLN	x	x						
QXSTRG	x	x	x	x			x	
QXPLSC	x		x	x			x	
PLTYPE	x		x	x			x	
QCVIEW	x		x	x			x	
QXDUMY	x		x		x			
KEY10	x		x	x				
SCREENP	x	x						
HDRPØS	x		x	x	x	x		
ISELECT	x	x	x	x	x	x	x	
USERSP	x	x	x	x	x	x	x	
ERASE9	x		x	x		x		
PLT999	x	x	x	x	x	x		
DRCTRY	x	x						
TRANSF	x	x	x	x	x	x	x	

NAME	ØVERLAY: GRAPØLF (2,X) Continued					
	(2,13)	(2,14)	(2,15)	(2,30)	(2,31)	(2,32)
KERRØR						
FILE99	x	x		x	x	x
ØFFLNE	x	x				
XCLVL					x	x
NPSCØM						
ZMCØNTR	x	x			x	x
QXGDØA						x
QXSCCØ						x
QXCØRD					x	x
CØNLIM					x	x
CØNLIM2						
FETBUFF						
IØBUF9B						
IØBUF9						
ITYPLT						
HDR999					x	x
NUMEL99					x	x
HCDATA						
BATCHM						
KEEPST						x
XBLWUP						x
RØRDABG					x	x
CALCST						
STLMTS					x	x
ØFFLN						
QXSTRG						x
QXPLSC						x
PLTYPE						x
QCVIEW						
QXDUMY						x
KEY10						
SCREENP						
HDRPØS					x	x
ISELECT					x	x
USERSP					x	x
ERASE9						
PLT999					x	x
DRCTRY				x		
TRANSF					x	x

803.4.1 /CØNPARS/, /KLRRØR/, /KQRNDM/ are ATLAS system common blocks, described in section 100.2.

803.4.2 Module Common Blocks

/GRMØDCM/	Secondary overlay communication common block for overlays under (1,0).
/C1/	Masking constants common block
/PLTCØM/	Common block for constant arrays required as calling parameters for routines WGRAD, WGRAH and WGRAT.
/FILE99/	System common block defining G/F file names and diagnostic file name.
/WGRADX/	Contains indices and buffer, permitting G/F generation in multiple secondary overlays.
/WGRATX/	Used to save indices for buffering text parameters when a plot is being created and written on G/F in different secondary overlays.
/CHCKGF/	Indices for checking logical sequence of writing GRAFSQ1 and/or GRAFSQ2.
/LABEL9/	Buffer for reading and writing labeling information for current plot being written on G/F.
/TEXT99/	Contains alphanumeric information and attributes of how and where that information is to be displayed on current plot being written on G/F.
/HDR999/	GRAFSQ1 (header file) read and write common block - contains user defined plot attributes.
/PLTBUF/	Read and write common block of X,Y,Z coordinates and plotting method written on GRAFSQ2.
/ØFFLNE/	CØMp80, GERBER or CALCØMP postprocessor directives for offline operator.
/XCLVL/	Number of contour levels and values at which isobar lines are computed.

/NPSCØM/	Common block used by NPS routines called by CØMp80 postprocessor.
/ZMCØNTR/	Surface definition matrix computed by (2,7) or connectivity data computed by (2,31) if contour plot is displayed.
/QXGDØA/	Graphic data set identifiers defined by BIGS.
/QXSCCØ/	Defaulted object space limits in rasters, initialized prior to creating a plot.
/QXCØRD/	Current screen object space and subject space limits, respectively.
/CØNLIM/	Subject space limits read off G/F when contour overlays involving the octant-weighted average algorithm are executed (i.e., (2,7) and (2,8)).
/CØNLIM2/	Incremental X,Y distances between contiguous mesh points when contour overlays involving the octant-weighted average algorithm are executed.
/FETBUFF/	I/Ø buffer for vector file (TAPE97).
/IØBUF9B/	I/Ø buffer for storing G/F directory, and for storing offline device dependent commands (TAPE99). Single buffer is used for both functions since TAPE99 is created just prior to exiting AGP, overwriting directory.
/IØBUF9/	I/Ø buffers for G/F (i.e., GRAFSQ1 and GRAFSQ2).
/ITYPLT/	Plot type code (e.g., 4 for perspective).
/HDR999/	Buffer for reading/writing G/F header records.
/NUMEL99/	Number of resulting triangles computed when the surface triangularization algorithm is used to compute contour plots.
/HCDATA/	Offline plotting device (e.g., 7HCALCØMP).
/BATCHM/	Code identifying execution mode: 0 = Conversational 1 = Batch Mode, Compute G/F, V/F and P/F. 2 = Generate P/F only from existing V/F.

/KEEPST/	Orthographic projection subject space limits if blowup option is used.
/XBLWUP/	Code indicating if blowup option was used or not (1 or 0).
/RØRDABG/	Orthographic projection order of rotation for RX, RY and RZ.
/CALCST/	Orthographic projection subject space limits computed if Transformation Menu was invoked without previously invoking the crosshairs.
/STLMTS/	Code to indicate if a data record has been read off G/F for current display being generated. If code >0, code = number of records read; otherwise, code = 0.
/ØFFLN/	Indicate which plots the user has selected interactively to generate on the vector file without generating online.
/QXSTRG/	Buffer for transforming user coordinates to screen coordinates.
/QXPLSC/	Used by scissoring routine to determine if beam position is to be redefined prior to plotting next set of points.
/PLTYPE/	Plot type (e.g., linear-linear type plot) used by BIGS.
/QCVIEW/	Rotation matrix and rotation angles used to determine the orientation of a 3D display (refer to BIGS subroutine GVIEW).
/QXDUMY/	Used for storing row and column numbers when displaying a matrix in numerical form, or for labeling grids.
/KEY10/	Current screen subject space limits (converted to RST system from XYZ) after invoking the Transformation Menu.
/SCREENP/	Screen position (IX,IY) where the alphanumeric cursor is currently displayed or will be displayed next.
/HDRPØS/	Record number (on GRAFSQ1) locating active plot or matrix header record for a specific subject index (refer to G/F Directory Menu).

/ISELECT/	Plot subject index name, plot title, and file number locating coordinates written on file GRAFSQ2.
/USERSP/	Contains selected subject index, selected plot index, error code, active Function Menu option number, and code identifying current graphic state for the menu executive (MENUEX).
/ERASE9/	Code indicating if previous plot is complete or being generated.
/PLT999/	Defines basic graphic state of current display (e.g., grid already displayed).
/DRCTRY/	Contains the number of subject indexes and the names of each written on G/F.
/TRANSF/	Current Plot Transformation Menu parameters (e.g., eyepoint, center-of-interest).

900. INTERACTIVE CONTROL

900.1 GENERAL INFORMATION

900.1.1 Purpose

To prepare, interpret and execute ATLAS commands in an interactive environment. ATLAS commands may be specified and processed one by one, or several at a time in what is called command procedures. ATLAS program files may be created, modified or interrogated through an edit facility internal to the ATLAS code.

900.1.2 Access

The interactive control processor is called from the ATLAS control program by the INTERACTIVE CØNTRØL statement (sec. 200, ref. 1-1).

900.1.3 Description

The interactive control capability consists of a set of interactive commands, and the code to generate and process, or invoke processing of these commands. Logically the CØNTRØL processor code performs three major functions; console communication, execution of command procedures, and monitoring of operations. These functions, together with the command language will be described in the following text in sufficient detail to support subsequent sections of this documentation. A more comprehensive discussion related to the use of the interactive capability is contained in reference 1-1, section 200.

1. Interactive Control Language. There are two types of interactive commands; one which will be referenced to as regular commands, and another which is called priority commands. The latter type is used entirely within the console communication function and will therefore be discussed in conjunction with this function.

   The regular commands must adhere to the following general format (the characters <> are used to enclose components of the command and are not part of the command itself):

   <label> <:> <id> <d_1> <parameter list> <d_2> <$>

label - Command label consisting of from 1 to 7 alphanumeric characters. The first character must be a letter.

: - Command label delimiter.

id - Command identifier consisting of a primary ID, and in some cases an additional secondary ID. Both IDs can have from 1 to 8 alphanumeric characters.

d_1 - ID delimiter consisting of either the character <(>, left parenthesis, or of the character <,>, comma.

parameter list - Actual parameters to be transmitted to the executing code. The parameter list syntax is described in section 200.2.

d_2 - Parameter list delimiter consisting of either the character <)>, right parenthesis, or of the character <.>, period.

$ - End of command character.

Blank characters interspersed in the command have no effect, except in conjunction with the TØ-BY parameter list statement as described in reference 1-1, section 200.

2. Console Communication. The console communication can occur in either of two modes; the A-mode (ATLAS), or the I/E-mode of the edit facility. The A-mode is the default case, the edit mode is entered by issuance of a special priority command.

In the A-mode regular interactive commands are typed in line-by-line and transmitted to a temporary core buffer. This input mode is interrupted either by issuing a priority command, or by giving the signal to start execution of the specified commands. A priority command is detected and processed out-of-order in relation to the regular commands. Once execution of the command is

completed, the control will revert to the A-mode, ready to resume preparation of regular commands. Upon detecting the execute-signal the command buffer will be transmitted to a reserved disk file (TEXTFIL) and control will be directed to the execute function. The commands residing on TEXTFIL are collectively called a command procedure.

The edit facility is equivalent to the BCS/CMEDIT program as of June, 1975. All capabilities of the CMEDITor with respect to generating, modifying or scanning files are available from inside the ATLAS program. In particular it is anticipated that the edit facility in ATLAS will be used to generate command procedures and ATLAS input data files.

Priority commands have the following general format:

<&> <id> <,> <argument>

& - The ampersand character which signifies the priority command.

id - Command identifier consisting of from 1 to 10 alphanumeric characters.

, - Command ID delimiter.

argument - A number or an alphanumeric string of characters.

Blank characters interspersed in a priority command are ignored.

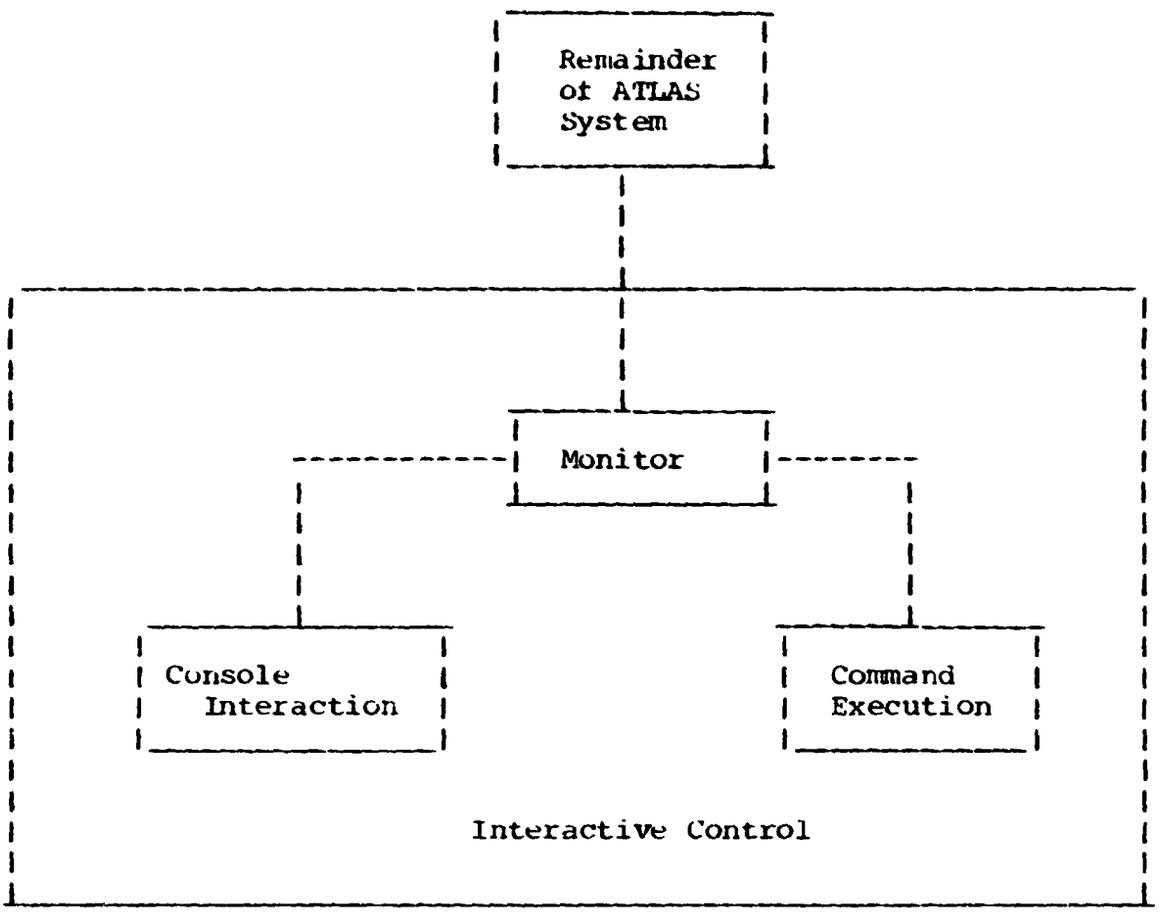
3. Command Execution. The execution function operates on named command procedures each of which may contain any number of commands. The actual processing of a procedure takes place in two stages. In stage 1 the procedure is interrogated command by command, and the source code is translated into a defined object code format (sec. 900.2.2.2). The source code and the object code reside on separate sequential files. In stage 2, provided the "compilation" taking place in stage 1 was successful, the command execution is actually carried out. The code to perform these operations may be located anywhere in the ATLAS system depending on the type of command to be processed. Execution of the ATLAS MASS module, for instance,

will require transfer to the MASSØLF overlay. The code to process other types of commands may be local to the interactive control processor. Thus, the execute function must contain the logic required to perform the necessary branching to, and recovery from externally and internally located code.

Command procedures can be executed in two different modes. In the FULL mode all commands in the procedure are processed without any interaction allowed from the terminal. In the STEP mode the control returns to the terminal after execution of each command. At this point any action may be initiated, including processing of new command procedures, before processing of the current procedure is resumed.

Command procedures may be nested through procedure calls (RUN-commands) up to a certain level defined in the code. The highest level procedure will always be the one currently being processed. Once a procedure processes a revert to a lower level procedure, the procedure itself and any intermediate procedures will be purged from the procedure linkage definition.

4. Monitoring of operations. The relationship between the three interactive control processor functions and the remainder of the ATLAS system is shown in the illustration.



The monitor function controls the interaction between the console communication function, the command execution, and the external functions of the ATLAS system.

900.2 FILES AND FORMATS

900.2.1 Reserved Files

The following file names are reserved for use by the interactive control processor:

TLXTFIL

ØBJECTx (Where x may be any alphanumeric character)

HISTFIL

These are all sequential files with usage as described below.

TEXTFIL contains source code for the (last) command procedure prepared in the A-type console mode.

ØBJECTx contains object (compiled) code of source procedures from TEXTFIL. The last character of the name will be the display code equivalent of the procedure level.

HISTFIL contains the "dayfile" of the interactive control activity. Just before the execution of a command is initiated, a copy of the command is written onto the HISTFIL.

Apart from these reserved files the interactive control processor will accept any source file prepared in the editor, and generate object code files according to specifications on CØMPILE or RUN commands.

900.2.2 File Formats

1. Source File Format

   Commands are written on the source file according to the format defined in section 900.1.2.1. End-of-line is indicated by a zero 12-bit byte. If a line ends at the 9th or 10th character of a word, an additional zero word will be added to the input record.

2. Object File Format

   word 1 - 10HØBJECTFILE

   word 2 - Number of instructions in file (=NN)

   word 3 - Status = I

   I = -2 - Fatal errors detected during compilation.

I = -1 - Warning conditions detected during compilation.

I = 0 - No warning conditions or fatal errors detected. Procedure is inactive.

I > 0 - Procedure is active. I is the number of the last command being executed from the procedure.

word 4 - Execution mode (=4HFULL or 4HSTEP)

word 5 - Name of calling procedure (revert file).

word 6 - Total number of warning conditions detected during compilation.

word 7 - Total number of fatal errors detected during compilation.

word 8 - 0 (not used)

word 9 - 0 (not used)

word 10 - 0 (not used)

word 11 - Bits 59-18: Contains the label of first command.

Bits 17-0: Pointer to first command (=KK)

word 12 - Same as word 11 for second command.

. . . . . .

word 11+NN - Length of object file + 1

From here on the commands are stored sequentially according to the following format:

First command

word KK - Status for first command (=J)

J = -2 - Fatal compilation errors

J = -1 - Warning conditions encountered

J = -0 - Compilation ok

word KK+1 - Pointer to source copy of command (=LL)

word KK+2 - Pointer to object code of command (=MM)

word KK+3 - Number of warning conditions encountered for command

word KK+4 - Number of fatal errors encountered for command

word KK+5 - Room for up to 10 warning or error codes.

word KK+14

word LL - Source copy of command

. . . . . .

word MM - Compiled format of command. The format for each command is described in section 900.2.3.

. . . . . .

The second command follows, etc.

900.2.3 Compiled Command Formats

The source form of the commands is shown here to clarify the compiled formats. For a description of the effect of each command, parameter options and default values, etc., the reference 1-1, section 200. should be used.

1. CØMPILE (I=afile, Ø=bfile, F=format, M=mode)

Compiled format is:

word 1 - Bits 59-12: 8LCØMPILEb
Bits 11-0 : Command type (=1)

word 2 - Length of compiled format

word 3 - Compile input file (afile)

word 4 - Compile output file (bfile)

word 5 - Source code format

word 6 - Procedure execution mode

2. RUN (I=afile, Ø=bfile, F=format, M=mode)

Compiled format is:

word 1 - Bits 59-12: 8LCØMPILEX

Bits 11-0 : Command type (=2)

words 2-6 - Same as for CØMPILE command

3. RUN (Ø = bfile, M = mode)

Compiled format is:

word 1 - Bits 59-12: 8LRUNbbbbb

Bits 11-0: Command type (=3)

word 2 - Length of compiled format

word 3 - Procedure name (object code file)

word 4 - Procedure execution mode

4. REVERT (Ø = bfile, M = mode)

Compiled format is:

word 1 - Bits 59-12: 8LREVERTbb

Bits 11-0 : Command type (=4)

word 2 - Length of compiled format

word 3 - Revert procedure name

word 4 - Execution mode to be assumed by the revert procedure

5. RETURN

Compiled format is:

word 1 - Bits 59-12: 8LRETURNbb

Bits 11-0 : Command type (=5)

word 2 - Length of compiled format

6. CØRE (FL = nnn)

Compiled format is:

word 1 - Bits 59-12: 8LCØREbbbb

Bits 11-0 : Command type (=6)

word 2 - Length of compiled format

word 3 - Requested field length (=nnn)

7. QUIT

Compiled format is:

word 1 - Bits 59-12: 8LQUITbbbb

Bits 11-0 : Command type (=7)

word 2 - Length of compiled format

8. EXECUTE module (plist)

Compiled format is:

word 1 - Bits 59-12: 8LEXECUTEb

Bits 11-0 : Command type (=8)

word 2 - Length of compiled format

word 3 - Overlay file name of module

word 4 - Primary overlay number

word 5 - KCØNPAR of common block /CØNPARS/

word 6 on - Array CØNPARS of common block/CØNPARS/

9. READ INPUT (plist)

Compiled format is:

word 1 - Bits 59-12: 8LREADbbbb
Bits 11-0: Command type (=9)

word 2 on - Same as for EXECUTE command

10. LØAD name (plist)

Compiled format is:

word 1 - Bits 59-12: 8LLØADbbbb

Bits 11-0: Command type (=10)

word 2 on - Same as for EXECUTE command

11. SAVE name (plist)

Compile format is:

word 1 - Bits 59-12: 8LSAVEbbbb

Bits 11-0 : Command type (=11)

word 2 - Same as for EXECUTE command

12. PRINT name (plist)

Compile format is:

word 1 - Bits 59-12: 8LSAVEbbbb

Bits 11-0 : Command type (=12)

word 2 - Same as for EXECUTE command

13. INDEX FILES (plist)

Compiled format is:

word 1 - Bits 59-12: 8LINDEXbbb

Bits 11-0 : Command type (=13)

word 2 on - Same as for EXECUTE command

14. PRØBLEM ID (text)

Compiled format is:

word 1 - Bits 59-12: 8LPRØBLEMb

Bits 11-0 : Command type (=14)

word 2 - Length of compiled format

word 3 on - Contains "text"

15. CHANGE ID (text)

Compiled format is:

word 1 - Bits 59-12: 8LCHANGEbb

Bits 11-0 : Command type (=15)

word 2 on - Same as for PRØBLEM ID command

16. PURGE name (plist)

Compiled format is:

word 1 - Bits 59-12: 8LPURGEbbb

Bits 11-0 : Command type (=16)

word 2 - Length of compiled format

word 3 - KCØNPAR of common block /CØNPARS/

word 4 on - Array CØNPARS of common block /CØNPARS/

17. RENAME MATRIX (plist)

Compiled format is:

word 1 - Bits 59-12: 8LRENAMEbb

Bits 11-0 : Command type (=17)

word 2 on - Same as for PURGE command

18. GØTØ (label)

Compile format is:

word 1 - Bits 59-12: 8LGØTØbbbb

Bits 11-0 : Command type (=18)

word 2 - Length of compiled format

word 3 - Contains the "label"

19. IF (logical expression)

Compile format is:

word 1 - Bits 59-12: 8LIFbbbbbb

Bits 11-0 : Command type (=19)

word 2 - Length of compile format

word 3 on - Contains the logical expression as a compressed string of characters

20. INTEGER (A, B(nn), . . .)

Compile format is:

word 1 - Bits 59-12: 8LINTEGERb

Bits 11-0 : Command type (=20)

word 2 - Length of compiled format

word 3 - Name of first parameter (=A)

word 4 - Index of first parameter (=0)

word 5 - Name of second parameter (=B)

word 6 - Index of second parameter (=nn)

word 7 - Etc.

21. REAL (A,), ..)

Compiled format is:

word 1 - Bits 59-12: 8LREALbbbb

Bits 11-0 : Command type (=21)

word 2 on - Same as for INTEGER command

22. LØGICAL (A, B(nn), ...)

Compiled format is:

word 1 - Bits 59-12: 8LLØGICALb

Bits 11-0 : Command type (=22)

word 2 on - Same as for INTEGER command

23. CØNTINUE

Compiled format is:

word 1 - Bits 59-12: 8LCØNTINUE

Bits 11-0 : Command type (=23)

word 2 - Length of compiled format

24. LIST option (A, B(nn), ...)

Compiled format is:

word 1 - Bits 59-12: 8LLISTbbbb

Bits 11-0 : Command type (=24)

word 2 - Length of compiled format

word 3 - Contains "option"

word 4 - Number of parameter list statements

word 5 - Name of first parameter (=A)

word 6 - Index of first parameter (=0)

word 7 - Name of second parameter (=B)

word 8 - Index of second parameter (=nn)

word 9 - Etc.

25. Arithmetic Expression

Compile format is:

word 1 - Bits 59-12: 8LAEXPRbbb

Bits 11-0 : Command type (=25)

word 2 - Length of compiled format

word 3 - Contains the arithmetic expression as a compressed string of characters

900.3 PROGRAM METHOD

900.3.1 General

Figure 900-1 depicts the functional structure of the interactive control processor, showing the relationship of the major components/subroutines of the system. The three major functions of the processor discussed in section 900.1.3; terminal communications, command execution, and monitoring of operations, are performed by the three subroutines CØNSØLE, EXECUTE and MØNITØR. These routines along with some important subordinate functions will be discussed in the following paragraphs.

The control processor is invoked from the CØNTRØL (1,0) overlay by a call to the MØNITØR subroutine. This call, along with some additional control code is generated from precompiling the ATLAS control statement INTERACTIVE CØNTRØL.

The control processor uses blank common as its primary work area in order to accommodate variable size information. The program maintains its own core size requirements. In general it will work on a minimum-size blank common area of 300 octal locations, however, when a larger requirement is detected at a certain stage of the operations, the field length will be changed for the duration of that requirement. The job field length specified by the user will be maintained for processing external to the control processor. The code used to manage the fieldlength is contained in the subroutine CHANGFL and the common block/CØRELEN/.

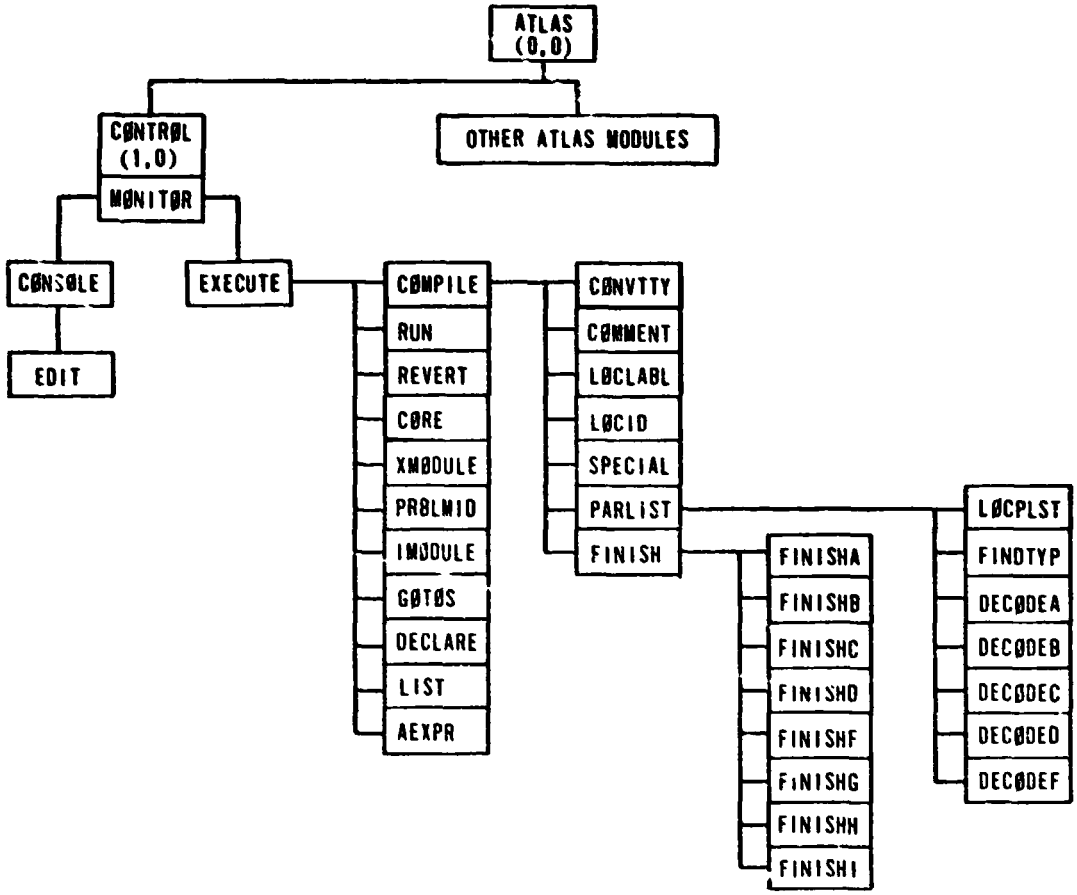


Figure 900-1. Functional Structure of the Interactive Control Processor

900.3.2 The MØNITØR Routine

The monitor directs the "traffic" between the terminal activity, the local command execution, and the external ATLAS code. Its actions depend on the value of the following logical variables:

IACTIVE - This variable has the value true for the duration of interactive processing. The value is set in the CØNTRØL (1,0) overlay. The monitor will turn the value to false in the following two occasions; 1) when a QUIT command is processed, and 2) if a batch job attempts to access the control processor.

READ - When READ = true the CØNSØLE subroutine will be called causing interaction with the terminal.

EXEC - When EXEC = true the monitor will call up the EXECUTE function.

EXTL - When EXTL = true this variable directs the monitor to branch to an external ATLAS program module. Logically the control processor is still in execute mode while this is going on, and control will return to the EXECUTE routine once the ATLAS module is completed.

QUIT - When QUIT = true the monitor will terminate the interactive processing mode.

900.3.3 The CØNSØLE Routine

a. Initialize.

aa. Rewind the source code file (TEXTFIL)

ab. Set minimum field length.

ac. Set input buffer pointers.

b. Read one record from terminal.

ba. Write A-prompt (A>), then read record.

baa. Find position of first and last non-blank character in input record (routine NØNBLNK)

bab. If no non-blank characters found, go to (ba) for new record.

bac. If first character is ampersand (&) we have a priority command. Branch to (c).

bad. If last character is number symbol (#) it is a signal to terminate terminal mode. In that case - -

bada. Set terminal termination flag (FINI = true).

badb. Replace # character by $ character in case user forgot to indicate end-of-command himself.

bb. Store input record in source code buffer.

bba. Add end-of-line symbol to input record (12 bit zero byte)

bbb. Increase fieldlength if required.

bbc. Copy input record to buffer.

bbd. If termination flag is set branch to (d). If not branch to (b) for more.

c. Process priority commands

ca. Compress command, ie. take out all blank characters (routine CØMPRES).

cb. Find command type (routine CØMPARE)

cba. If type not recognized write message and branch to (b).

cbb. Branch according to command type.

cc. &DELETE/&PRINT commands

cca. Find beginning of argument list (position of comma)

ccb. If no comma found, interpret that as argument missing. In that case ---

ccba. Set default argument value =1

ccbb. Branch to (ccd)

ccc. Decode argument

ccca. Check that argument is positive integer

ccd. Space back in source code buffer the specified number of lines

ccda. If buffer is empty write message and branch to (b)

ccdb. Locate the specified number of zero bytes. If no zero bytes found, or fewer than specified, position at beginning of source buffer.

cce. If &DELETE command ---

ccea. Reset source buffer pointers at position found in (ccdb)

cceb. Branch to (b)

ccf. If &PRINT command ---

ccfa. Print from position found in (ccdb) to end of source buffer

ccfb. Branch to (b)

cd. &CØRE Command

cda. Find beginning of argument List (position of comma)

cdb. If comma missing, interpret that as argument missing. Write message and branch to (b)

cdc. Decode argument

cdca. Check that argument is an integer within reasonable range

cdcb. Reset the value for user fieldlength

cdd. Branch to (b)

ce. &EDIT command

cea. Find beginning of argument List (position of comma)

ceb. If comma is missing - no argument list

ceba. The default edit file is TEXTFIL

cebb. Branch to (ced)

cec. Decode argument list

ceca. Check that name is legal; ie, length ≤ 7 characters and first character being a letter

ced. Write the source code buffer to disk file TEXTFIL

cee. If edit file ≠ TEXTFIL, open the file

cef. Call the EDIT subroutine

ceg. Return from edit mode

cega. Close the edit file, unless it is TEXTFIL

cegb. Check length of TEXTFIL to see if fieldlength requirements have changed.

cegc. Read TEXTFIL

cegd. Branch to (b)

d. Return to monitor

da. First handle the case when the source code buffer is empty (no new commands specified)

daa. If no procedures are active, write message and branch to (b)

dab. Set up return to current procedure

ab. Source code buffer contains new commands. Nest this procedure to current procedure

dba. Set up object code file name. (First 6 characters are ØBJECT. The 7'th character is the display code equivalent of the procedure level)

dbb. Set up compile-execute command to process the new procedure

dc. Write source code buffer to TEXTFIL

dd. Set control to transfer to execute mode

de. Return

900.3.4 The EXECUTE Routine

a. Initialize

aa. Set minimum fieldlength

b. Branch according to entry mode

ba. If processing of external ATLAS code has just been terminated (variable EXTL = true) then branch to (d)

bb. Otherwise execution of a new command is to be initiated. Branch to (c)

c. Start execution of new command

ca. Set control variable XBST = false (XBST is set true if, during execution of one command all the control information for executing next command is set up)

cb. Branch according to command type to one of cc through cq

cc. CØMPILE and CØMPILEX commands (subroutine CØMPILE)

cd. RUN commands (subroutine RUN)

ce. REVERT commands (subroutine REVERT)

ct. RETURN commands

cfa. Set control variable READ = true.

cfb. Branch to (e)

cg. CØRE command (subroutine CØRE)

ch. QUIT command

cha. Set control variable QUIT = true

chb. Branch to (e)

ci. EXECUTE, READ, LØAD, SAVE, PRINT and INDEX commands (subroutine XMØDULE)

cj. PRØBLEM ID and CHANGE ID commands (subroutine PRBLMID)

ck. PURGE and RENAME commands (subroutine IMØDULE)

cl. GØTØ commands (subroutine GØTØS)

cm. IF commands (subroutine IFS)

cn. INTEGER, REAL and LØGICAL commands (subroutine DECLARE)

co. CØNTINUE commands

coa. Branch to (e)

cp. LIST commands (subroutine LIST)

cq. Arithmetic statements (subroutine ALXPR - for the time being this is a dummy routine)

cr. Check if any of the subroutines above have requested transfer to external ATLAS code (check value of EXTL). If so, branch to (e)

d. Finish execution of one command

da. If error flag (KLKRØR) has been set ---

daa. Reset error flag (KLRRØR = 0)

dab. Request transfer to terminal (READ = true)

db. If transfer flag to terminal set, branch to (e)

dc. If execution buffer (EXBUF) has been set by last command (XBST = true), branch to (c) to have it executed

dd. If current procedure is executed in FULL mode (no interrupt from terminal) ---

dda. Set up a REVERT command to continue processing that procedure

ddb. Branch to (c) to have the REVERT command executed

de. If current procedure is executed in STEP mode (interrupt from terminal after each command) ---

dea. Set transfer flag READ = true

deb. Branch to (e)

df. If none of da - de above something is drastically wrong. Write message and transfer to terminal

e. Terminate execute mode (Set EXEC = false). Return to monitor

900.3.5 The COMPILE routine

The COMPILE routine translates a source code procedure (sec. 900.2.2.1) into an object code procedure (sec. 900.2.2.3/4)

a. Initialization

aa. Set minimum field length

ab. Open the source code file and the object code file

ac. Check that fieldlength is big enough to read the source file

ad. If source file is empty exit with error flag set

ae. Read source file

af. If source file is written in BATCH mode (ie. FØRTRAN compatible format) convert it to TTY format defined in sec. 900.2.2.1 (subroutine CØNVTTY)

ag. Count number of commands on source file. Count number of $ signs (routine CHARCNT)

ah. Initialize core buffer to contain object code

aha. Adjust fieldlength so the object buffer is "reasonably" big (it should at least be MINBUF big)

ahb. Set buffer to zero

ahc. Set 1'st through 4'th word of object code header.

b. Decode one command at a time

ba. Check if all commands have been decoded (are we past the last $ sign on the source file?)

baa. If decoding done branch to (c)

bb. Bound position of next command, i.e., position leading and trailing $ sign (subroutine LØCCHAR)

bc. Check if next command is a comment (logical function CØMMENT). A comment either starts with the two characters */, or it is a blank or an empty command.

bca. If command is a comment, branch to (ba)

bd. Increment command counter

be. Place copy of command in obj code buffer (see sec. 900.2.2.2, word LL)

bf. Set address of word MM of object code buffer

bg. Check that fieldlength is big enough. There must be room for max number of parameter list parameters (LPARAM)

bh. Locate and decode the command label (subroutine IØCLABL)

bha. Locate position of label delimiter (:)

bhaa. If no colon, branch to (bhe)

bhb. Locate position of equal sign (=). (We have to see if the colon is part of "set equal" (:=) of an arithmetic expression)

bhba. If no equal sign branch to (bhd)

bhc. Both colon and equal sign exist, do we have a label?

bhca. No label if : and = occur in that order, and are either consecutive or are separated by blank characters only. In that case branch to (bhe)

bhd. Decode label

bhda. Compress label to the left of the field

bhdb. Check that label is not empty, or longer than 7 characters

bhe. Terminate LØCLABL

bi. Locate and decode the command identification (subroutine LØCID)

bia. Locate position of id delimiter. It is either the first left parenthesis, or the first comma following the label delimiter. If none of these is found, end-of-ID is set to end of command

bib. Compress ID towards the label delimiter

bic. Locate primary ID (subroutine CØMPARE)

bica. If no primary ID found the command is either an arithmetic expression or un-recognizable. Branch to (bie)

bid. Look for secondary ID (subroutine CØMPARE)

bida. First mask off the primary ID so we can find out where the secondary ID starts. The primary ID need not be complete, so mask off as many characters as match with the complete ID (stored in array NLISTB)

bidb. If secondary ID empty branch to (bie)

bidc. Locate secondary ID (subroutine CØMPARE)

bie. Terminate LØCID

bj. Decode the special commands PRØBLEM ID, CHANGE ID, IF, and arithmetic expressions (subroutine SPECIAL). The parameter list of these commands does not follow the syntax of other commands, so they must be treated separately.

bja. Call the SPECIAL routine

bjb. Branch to (bm)

bk. Decode the parameter list of all regular commands according to the syntax rules and translated forms described in section 200.3 (subroutine PARLIST)

bka. Locate the position of the command parameter list (subroutine LØCPLST). The position of the leading parameter list delimiter is the command ID delimiter. The position of the trailing parameter list delimiter (a right parenthesis or a period) is found by a character search from the trailing command delimiter towards the leading command delimiter.

bkb. Decode one statement at a time. Statement delimiters are commas. Statement types are designated as follows:

type A: keyword(index)

ABCD(123)

type B: keyword = parameter

AA = 67, BB = VALUE

type C: keyword = multiparameter

SAM = (4, 9K, 13 TØ 21)

type D: keyword = list expression

ØAD = 1 TØ 9 BY 2

type E: keyword = matrix expression

MAT = [A - B * C]

type F: Matrix expression

[(A + B) * C = 0]

bkc. Locate leading and trailing delimiter for next statement (subroutine LØCCHAR)

bkd. Find the statement type (subroutine FINDTYP)

bke. Branch according to statement type to one of bkf through bkj

bkf. Decode statement type A (subroutine DECØDEA)

bkg. Decode statement type B (subroutine DECØDEB)

bkh. Decode statement type C (subroutine DECØDEC)

bki. Decode statement type D (subroutine DECØDED)

bkj. Decode statement type E and F (subroutine DECØDEF)

bkk. Branch to (bkc) if more statements to be decoded

bkl. Terminate PARLIST

bl. Complete decoding of parameter list for command (subroutine FINISH). In this section all unique options for a particular command will be tested, and the final object code for the command produced.

bla. Complete compilation of CØMPILE, CØMPLEX and RUN commands (subroutine FINISHA)

blb. Complete compilation of REVERT command (subroutine FINISHB)

blc. Complete compilation of CØRE command (subroutine FINISHC)

bld. Complete compilation of EXECUTE, READ, LØAD, SAVE, PRINT and INDEX commands (subroutine FINISHD)

ble. Complete compilation of PURGE and RENAME commands (subroutine FINISHF)

blf. Complete compilation of GØTØ commands (subroutine FINISHG)

blg. Complete compilation of INTEGER, REAL, and LØGICAL commands (subroutine FINISHH)

blh. Complete compilation of LIST commands (subroutine FINISHI)

bli. Terminate FINISH

bm. Set words KK through KK+4 of object code of command (sec. 900.2.2.2)

bn. Increment total error and warning counters

bo. Branch to (ba)

c. Complete compilation of command procedure.

ca. Set values of the object code header (first 10 words)

cb. Write summary error or warning messages if necessary

cc. Write object code buffer to disk

cd. If command is COMPILEX, prepare execution buffer to execute the compiled procedure

ce. Close source code file and object code file. (Not if source code file = TEXTFIL)

cf. Terminate COMPILE

900.4 COMMON BLOCK USAGE

900.4.1 /COMPARS/ /KERROR/ - ATLAS system common blocks described in section 100.2

900.4.2 /COMRELEN/ - Contains parameters for managing dynamic fieldlength during the execution of the interactive control processor

/DATABAS/ - Contains a core resident version of the data base

/LRRSECT/ - Contains variables related to error reporting for the interactive control code

/IACTIVE/ - Contains basic control variables for the interactive control processor, which must be kept in the root overlay during execution.

/NLISTA/ - Contains the priority command identifications

/NLISTB/ - Contains the primary command identifications

/NLISTC/ - Contains the secondary command identifications

/PARAM/ - Communication area for subroutines PARLIST and FINISH. It contains the decoded parameter list for one command.

/PFILES/ - Contains the standard file names used by the control processor

/XPARAM/ - Contains basic control parameters used internally by the interactive control code

/XBST/ - Contains control information internal to the execution mode.

1000. ATLAS LIBRARY ROUTINES

All modules in ATLAS are supported by the ATLASL library which is a combination of the ALIB routines and the SNARK library routines. ALIB routines are described in section 1001. Users should not call SNARK library routines directly but should use the SNARK language instead (ref. 10-1). Additional routines which are available only to control programs are contained in CLIB. These routines are described in section 1002. The ATLAS/NASTRAN interface routines described in sections 1003 and 1004 are also part of CLIB since they are used by control programs. ILIb supports the interactive capabilities of the CØNTRØL module as described in section 900. VLIB is used by control programs to supply the VERSIØN routine which initializes the VIPER common block. The creation of VLIB is described in section 502.6.

1001. ATLAS ALIB LIBRARY ROUTINES

ROUTINE: AINTG

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: Generates displacements and slopes at unsteady aerodynamics control points specified in the global axis system.

USAGE:
```
DIMENSION X(NPTS),Y(NPTS),Z(NPTS),DELZ(NRØWZ,NCOLS),
1     SA(NVARY),GAMMA(NPTS),DZ1(NRØWZ,NCOLS),
1     DZ2(NRØWZ,NCOLS)
CALL AINTG(X,Y,Z,NPTS, DELZ,NRØWZ,NCØL1,NCØLS,SA,
1     INDG,GAMMA,INDD,DZ1,DZ2)
```

PARAMETERS: I N P U T

X,Y,Z,NPTS - (X,Y,Z) Location of NPTS output points in global system

NCØL1,NCØLS - First column and number of columns to be used
If NCØLS=0, all columns≥NCØL1 will be used

SA - Interpolation coefficient array

INDG - Indicator for local orientation of normal component,
=0 - GAMMA assumed zero for all points
=1 - GAMMA(1) applies to all points (GAMMA in radians)
=2 - GAMMA (I) applies to point I

GAMMA - Array of normal orientations
GAMMA has same sign as DZ/DY of surface at point

INDD - Indicator for generation of slopes at output points.
=0 - No slopes, DZ1 and DZ2 not required
=1 - DZ1 = DZ/DX, DZ2 not required
=2 - DZ1 = DZ/DY, DZ2 not required
=3 - DZ1 = DZ/DX, DZ2 = DZ/DY

NROWZ - ≥ NPTS = Row dimension of the matrices that are to contain the output

O U T P U T

DELZ	-	Displacements at output points
NCØLS	-	No. columns returned
LZ1,DZ2	-	Slope at output points
INDD	-	Error indicator
	≥0 -	No error
	<0 -	error of the form 10H*NNNNNNNXX, NNNNNNN=routine name, XX=error no.
=10H*AINTG	1 -	Interpolation coefficient array type not recognizable

COMMON: None

SUBROUTINES: AINTT,PLATEØ,BEAMØ,MØTAXØ,MØTPTØ,PØLYØ

LANGUAGE: FØRTRAN

DISCUSSION:

(1) Transform output points to local interpolation coordinate system by calling GTØLT entry point of AINTT.

(2) Select interpolation output routine associated with interpolation information array.

(3) Generate displacements (optionally, slopes) at output points by calling:

PLATEØ for surface spline method
BEAMØ for beam spline method
MØTAXØ for motion axis method
MØTPTØ for motion point (rigid body) method
PØLYØ for polynomial method

(4) Modify displacements (slopes) by difference between dihedral at output point and orientation of local system with respect to the global system.

(5) Transform output points back to global system by calling LTØGT entry point of AINTT.

ROUTINE: AINTL

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: Generates displacements and slopes at unsteady aerodynamics control points specified in the local axis system

USAGE:
```
DIMENSIØN X(NPTS),Y(NPTS),Z(NRØWZ,NCOLS),SA(NVARY),
1     DZ1(NRØWZ,NCOLS),DZ2(NRØWZ,NCOLS)
CALL AINTL( X,Y,NPTS, Z,NRØWZ,NCØL1,NCØLS,SA,
1     INDD,DZ1,DZ2)
```

PARAMETERS:

I N P U T

X,Y,NPTS	-	(X,Y) Location of NPTS output points in local system
NCØL1,NCØLS	-	First column and number of columns to be used
INDD	-	Indicator for generation of slopes at output points =0 - No slopes. DZ1,DZ2 not required =1 - DZ1 = DZ/DX, DZ2 not required =2 - DZ1 = DZ/DY, DZ2 not required =3 - DZ1 = DZ/DX,DZ2 = DZ/DY
NROWZ	-	≥ NPTS The row dimension of the matrices containing the output displacements and slopes.

O U T P U T

Z	-	Displacements at output points
NCØLS	-	No. columns returned
DZ1,DZ2	-	Slope at output points
INDD	-	Error indicator ≥0 - No error <0 - error of the form 10H*NNNNNNNXX, NNNNNNN=routine name, XX=error no. =10H*AINTL 1 - Interpolation coefficient array type not recognizable

COMMON: None

SUBROUTINES: PLATEØ,BEAMØ,MØTAXØ,MØTPTØ,PØLYØ

LANGUAGE: FØRTRAN

DISCUSSION: (1) Select interpolation output routine associated with interpolation coefficient array

(2) Generate displacements (optionally, slopes) at output points by calling:

PLATEØ	for surface spline method
BEAMØ	for beam spline method
MØTAXØ	for motion axis method
MØTPTØ	for motion point (rigid body) method
PØLYØ	for polynomial method

ROUTINE: AINTT

AUTHOR: M. C. Redman

DATE: January 1974

PURPOSE: Coordinate transformation routine for unsteady aerodynamics interpolation routine AINTG

USAGE:
```
DIMENSION X(NPTS),Y(NPTS,Z(NPTS),R(3,3),T(3)
CALL AINTT(X,Y,Z,NPTS,R,T)
```

PARAMETERS:

I N P U T

NPTS - Number of points
X,Y,Z - Coordinates of points
R - Rigid rotation matrix, global to local
T - Translation vector, global to local

O U T P U T

X,Y,Z - Transformed coordinates

COMMON: None

SUBROUTINES: None

LANGUAGE: FORTRAN

DISCUSSION: Entry Points:

GTOLT - Transforms points from global to local axis.

LTOGT - Transforms points from local to global axis.

ROUTINE: ARCL

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: Determine the arc length along a cubic, between two points

USAGE: S = ARCL(X0,Y0,C1,C2,C3,X,Y)

PARAMETERS: I N P U T

X0	-	x-coordinate of initial point
Y0	-	y-coordinate of initial point
C1,C2,C3	-	Cubic coefficients in Y defining the curve (note: C0 is not required)
X	-	x-coordinate of final point
Y	-	y-coordinate of final point

O U T P U T

S	-	Arc length from (X0,Y0) to (X,Y) along the curve $x = C0+C1*y+C2+y^2+C3*y^3$

COMMON: None

SUBROUTINE: None

LANGUAGE: FORTRAN

DISCUSSION: If the curve is a cubic, i.e., C3 $\neq$ 0, a 4 point Gauss-Legendre quadrature is applied. Otherwise a closed form solution to the integral is evaluated.

ROUTINE:	BEAMØ
AUTHOR:	Unknown
DATE:	Unknown
PURPOSE:	Calculate modal displacements and optionally slopes at a set of output points, given an interpolation information array generated for a beam-spline system.
USAGE:	See listing for a complete definition of all parameters
COMMON:	None
SUBROUTINES:	ARCL, CØMCUB, LATSRT, HERMINT, ZERØCØL
LANGUAGE:	FØRTRAN

ROUTINE: BITCAT

AUTHOR: W. J. Erickson

DATE: December 1973

PURPOSE: To count the on bits in the masked portion of the words in an array

USAGE:
```
DIMENSION ARRAY(N)
CALL BITCAT(ARRAY,MASK,N,NBITS)
```

PARAMETERS: I N P U T

ARRAY - Array with bits to be counted
MASK - Masking constant
N - Dimension of array

O U T P U T

NBITS - Number of bits counted

COMMON: None

SUBROUTINES: None

LANGUAGE: COMPASS

DISCUSSION: Use BITCAT in preference to NBITS1 when an array is to be counted, or when the use of a masking word is indicated.

ROUTINE: CHKEØR

AUTHOR: S. H. Gadre

DATE: January 1974

PURPOSE: To convert "MØDE2" LØDAREC input to "MØDE1" input

USAGE:
```
DIMENSIØN DATA(9)
CALL CHKEØR(DATA)
```

PARAMETERS: I N P U T

DATA - Card image (8A10) as read by LØDAREC in DATA(1) - DATA(8) and blank filled word DATA(9)

O U T P U T

DATA - Record converted to "MØDE1" format

COMMON: None

LANGUAGE: FØRTRAN

DISCUSSION: The 90 characters in "DATA" are scanned right to left. If the first non-blank character found is any character other than + (plus sign), a blank and a slash are written to its right. If it is a plus sign, it is replaced by a blank.

ROUTINE: CØISHAP

AUTHORS: G. vonLimbach and S. Wahlstrom

DATE: December 1973

PURPOSE: Generation of isoparametric shape functions and their first derivatives for all corner nodes of quadrilaterals or hexahedrons.

USAGE: (quad)

```
      DIMENSIØN EVAL(3), ICØØ(ID,4), NEDGE(4),
     1      XSI(5,4),NEDPØI(4),EDCØRN(2,4),FUNC(2,4),
     1      PARTIAL(ID,4)
```

(hexa)

```
      DIMENSIØN EVAL(3),ICØØ(ID,8),NEDGE(12),
     1      XSI(5, ) NEDPØI(12),EDCØRN(2,12),FUNC(2,8),
     1      PARTIAL(ID,8)

      CALL CØISHAP(ID,EVAL,ICØØ,NEDGE,XSI,NEDPØI,
     1      EDCØRN,FUNC,PARTIAL)
```

PARAMETERS: I N P U T

- ID - Row dimension of ICØØ and PARTIAL (2 for quadrilateral, 3 for hexahedron)
- EVAL - Array of coordinates of the point at which the function and its partials are computed. (Dimension -3)
- ICØØ - Array of corner coordir tes. ICØØ(i,j) is coordinate X(i) of corner j
- NEDGE - Array of interior nodes. NEDGE(i), where i=NEDPØI(e), is the number of interior nodes on edge e.
- XSI - Array of local edge coordinates. XSI(k,i), where i=NEDPØI(e), is the local edge coordinate of node k on edge e.
- NEDPØI - NEDPØI(i) is the pointer to EDGE i in NEDGE and XSI. Thus NEDGE and XSI must be stored in the same order, but not necessarily in order of increasing edge number.
- EDCØRN - Array of corner nodes. EDCØRN(j,i) is the jth corner node of edge i (j=1,2)

INPUT/OUTPUT

FUNC - FUNC(1,i) is output as the value of the shape function N(i).

FUNC(2,i) is replaced with normalization value (1.125) if input as zero. Otherwise, not changed.

O U T P U T

PARTIAL - PARTIAL(i,j) = $\frac{dN(j)}{dX(i)}$

COMMON: None

SUBROUTINES: None

LANGUAGE: FORTRAN

DISCUSSION: Isoparametric shape functions are treated in almost all texts on elementary finite element analysis, e.g., O. C. Zienkiewicz, The Finite Element Method in Engineering Science, 2nd edition, McGraw Hill, London, 1971, p. 107ff.

ROUTINE: CØMCU

AUTHOR: G. W. Erwin, Jr.

DATE: October 1962

PURPOSE: To fit a composite cubic through n(≤400) points.

USAGE:
```
DIMENSIØN S(400),X(400),Y(400)
CALL CØMCU(DA,DB,S,X,Y,L,M,N,NDA,NDB)
```

PARAMETERS: I N P U T

X - Array of independent variables
Y - Array of dependent variables
DA - Values of the derivative at x(1)
DB - Values of the derivative at (n)
L - Ignored
N - The number of points
NDA - The order of the derivative at x(1) (1 or 2)
NDB - The order of the derivative at x(n) (1 or 2)

O U T P U T

S - The array of slopes at the given points
M - Error condition
= 0 if success
= 1 if floating-point overflow (77462B)
= 2 if overflow light on
= 4 if divide check

A positive value of M other than 1,2, or 4 indicates more than one of these error conditions in which case M will be the sum of their individual returns.

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: CØMCU establishes a cubic between each pair of adjacent points. The cubic matches its neighbors in function value and in the first two derivatives at its end points.

ROUTINE: CUBIC2

AUTHOR: W. R. Smith

DATE: March 1961

PURPOSE: To fit a cubic to two points, given the slope at each.

USAGE:
```
DIMENSION X(2),Y(2),D(2),C(4)
CALL CUBIC2 (X,Y,D,C,M)
```

PARAMETERS: I N P U T

X(1),X(2) - X - coordinates of given points
Y(1),Y(2) - Y - coordinates of given points
D(1),D(2) - Slopes (dy/dx) at given points

O U T P U T

C - Array of cubic coefficients
M - Error condition
= 1 if success
= 2 if accumulator overflow
= 3 if X(1) = X(2)

COMMON: None

SUBROUTINES: None

LANGUAGE: FORTRAN

DISCUSSION: The subroutine sets up the system of four simultaneous equations expressing the four given conditions, and solves it for the coefficients of the cubic:

$$y = C1+C2*x+C3*x^2+C4*x^3$$

ROUTINE: DATSRT

AUTHOR: Gary C. Peterson

DATE: March 1975

PURPOSE: Given an array of real numbers, sort the array into ascending order and create a mapping array

USAGE: CALL DATSRT (A,AMAP,N,DIS,NØTUSD)

PARAMETERS:

I N P U T

A	The array to be sorted
N	Length of A
DIS	A value used to determine when two elements of A are sufficently close to be considered identical
NØTUSD	A value, which, when found, will be omitted from the ordered array

O U T P U T

A	The ordered array
AMAP	An array of integers stored as floating point numbers mapping the input A to the output A
N	Length of A, AMAP after ordering

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: Called by the routine BEAMØ.

ROUTINE: DECØDIR

AUTHOR: C. A. Felippa
L. J. Schmid

DATE: December 1969

PURPOSE: DECØDIR is a support routine to LØDAREC. It updates error counters if fatal error 78 occurs.

USAGE:
```
CØMMØN/CARDS/REC(250),ID(250),KRECS,KCDS,ITEMS,
1     KERR(3),JBL
CALL DECØDIR
```

COMMON:
```
CØMMØN/CARDS/REC(250),ID(250),KRECS,KCDS,ITEMS,
1     KERR(3),JBL
```
See the LØDAREC routine description for definitions of the variables.

SUBROUTINE: None

LANGUAGE: FØRTRAN

DISCUSSION: KERR(3) ≠ 0 implies an occurrence of error 78.

ROUTINE: DERIV1

AUTHOR: L. F. Wade

DATE: October 1961

PURPOSE: To find the first derivative of the quadratic through three given points at a specified one of these points. This provides a good approximation to the slope of a function at a point, particularly if the other two points used are nearby.

USAGE:
```
DIMENSION X(3), Y(3)
D1 = DERIV1(X,Y,N)
```

PARAMETERS:

I N P U T

X - The array of x-coordinates
Y - The array of y-coordinates
N - 1,2, or 3, the point at which the derivative is wanted.

O U T P U T

D1 - The first derivative at the requested point

COMMON: None

SUBROUTINE: None

LANGUAGE: FORTRAN

DISCUSSION: The subroutine finds the unique polynomial of degree two through the given points, then evaluates its first derivative at the specified point.

The x(i) must be distinct, but may be in any order and unevenly spaced. The points may be colinear.

ROUTINE: DERIV2

AUTHOR: G. W. Erwin, Jr.

DATE: May 1961

PURPOSE: To find the second derivative of the cubic through four given points (X(i), Y(i)) at an arbitrary point whose x-coordinate is given.

USAGE:
```
DIMENSION X(4), Y(4)
D2 = DERIV2 (X,Y,XX)
```

PARAMETERS: I N P U T

X - The array of x-coordinates
Y - The array of y-coordinates
XX - The x-coordinate of the point at which the second derivative is wanted.

O U T P U T

D2 - The second derivative at the point x.

COMMON: None

SUBROUTINES: None

LANGUAGE: FORTRAN

DISCUSSION: The subprogram finds the unique polynomial of degree three through the given points, then evaluates its second derivative at the desired x, which need not be one of the four given x coordinates.

The x(i) must be distinct (the subprogram returns with D2 = 0, if they are not). The x(i) may be in any order and unevenly spaced.

ROUTINE: DHRMINT

AUTHOR: M. C. Redman

DATE: August 1974

PURPOSE: Given F(X1),F(X2),DF(A1)/DX, and DF(X2)/DX, Calculate DF(X)/DX, X2≥X≥X1 for N modes where Hermite interpolation is used to approximate F(X).

USAGE: DIMENSION DFX(NR1,N), FX1(NR2,N), DFX1(NR2,N), FX2(NR2,N), DFX2(NR2,N) CALL DHRMINT (X,DFX,NR1, X1,FX1,DFX1,X2,FX2,DFX2,NR2,N)

PARAMETERS: See listing for complete explanation of parameters.

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: Used by MØTAXØ.

ROUTINE: EDISHAP

AUTHOR: G. von Limbach
S. Wahlstrom

DATE: January 1974

PURPOSE: Generation of isoparametric shape functions and their first derivatives for all interior nodes on one edge of a hexahedron or quadrilateral.

USAGE: (quad)
```
DIMENSION ID(4),EVAL(3),CØØ( ), PARTIAL (3,2),
1     FUNC(2, )
```

(hexa)
```
DIMENSION ID(5), EVAL(3),CØØ( ), PARTIAL (3,3),
1     FUNC(2, )
```

```
CALL EDISHAP (ID,EVAL    ,PARTIAL,FUNC)
```

PARAMETERS: I N P U T

ID - ID(1)= 2 for quadrilateral
3 for hexahedron
ID(2)= Number of interior nodes on edge
ID(3)= N=ID(1)+2
ID(N) Direction indicators of ± 1 or 0. When ID(2+i) is ± 1, this indicates that X(i) has this value for the edge. When it has the value of zero, it indicates that the edge is in the X(i) direction.

EVAL - Evaluation point. EVAL(i) is x(i) for the point at which the function and its partials are evaluated.

CØØ - Edge coordinates. CØØ(k) is node k on the edge. Corners are included.

INPUT/OUTPUT

FUNC - FUNC(1,i) is output as the value of the shape function N(i).

FUNC(2,i) is replaced with normalization value if input as zero. Otherwise not changed.

O U T P U T

PARTIAL - PARTIAL(i,j) = $\frac{dN(j)}{dX(i)}$

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: Isoparametric shape functions are treated in almost all texts on elementary finite element analysis. See for example: O. C. Zienkiewicz, The Finite Element Method in Engineering Science, 2nd edition, McGraw Hill, London, 1971, p. 107ff.

ROUTINE: ELEPICK

AUTHOR: L. C. Carpenter

DATE: July 1971

PURPOSE: To pick selected flexible elements out of a KSF matrix and store it in the new order into a scratch array.

USAGE:
```
DIMENSION MFLEX( ),LCT( ),ISCRAT( )
COMMON/SORTELC/IL,IU,NPASS,INTSCR,MAXSCR,ISCRFUL,
1   LASTELW,FLEXR
COMMON/MSKCON/MSK01,MSK02,...MSK60
CALL ELEPICK(MFLEX,LCT,ISCRAT)
```

PARAMETERS:

I N P U T

MFLEX - KSF matrix
LCT - Correspondence table (old to new)

O U T P U T

ISCRAT - Intermediate KSF matrix in new order

COMMON:
```
COMMON/SORTELC/IL,IU,NPASS,INTSCR,MAXSCR,ISCRFUL,
1     LASTELW,MFLEXR
```

IL - Lower bound for selection
IU - Upper bound for selection
NPASS - Number of elements that will fit in ISCRAT
INTSCR - Next free word for element data block in scratch
MAXSCR - Total size of scratch
ISCRFUL - Scratch overflow indicator
LASTELW - Number of last element written
MFLEXR - Size of KSF matrix

COMMON/MSKCON/MSK01,MSK02,...MSK60

masking constants

SUBROUTINE: None

LANGUAGE: FORTRAN

DISCUSSION: Row I of LCT contains the new address (J) of the old Itn element. A pass is made through the KSF matrix. If the 'J' for an element lies within the limits IL - IU, the data for the element is inserted into ISCRAT.

ROUTINE: ELEPUT

AUTHOR: L. C. Carpenter

DATE: July 1971

PURPOSE: To construct the final KSF matrix from the intermediate matrix ISCRAT, generated by ELEPICK.

USAGE:
```
DIMENSION ISCRAT( ),MFLEX( )
COMMON/SORTELC/IL,IU,NPASS,INTSCR,MAXSCR,ISCRFUL,
1   LASTELW,MFLEXR
COMMON/MSKCON/MSK01,MSK02,...MSK60
CALL ELEPUT(ISCRAT,MFLEX,NIR,NWORDS)
```

PARAMETERS: I N P U T

ISCRAT - Intermediate KSF matrix

O U T P U T

MFLEX - The KSF matrix
NIR - Number of elements in the KSF matrix
NWORDS - Size of KSF matrix

COMMON:
```
COMMON/SORTELC/IL,IU,NPASS,INTSCR,MAXSCR,ISCRFUL,
1LASTELW,MFLEXR
```

IL - Lower bound for selection
IU - Upper bound for selection
NPASS - Number of elements in ISCRAT
INTSCR - Next tree word for element data block in scratch
MAXSCR - Total size of scratch
ISCRFUL - Scratch overflow indicator
LASTELW - Number of last element written
MFLEXR - Size of KSF matrix

```
COMMON/MSKCON/MSK01,MSK02,...MSK60
```

masking constants

SUBROUTINE: None

LANGUAGE: FORTRAN

DISCUSSION: Data for elements are picked up sequentially from ISCRAT and put into KSF matrix in appropriate form.

ROUTINE: ESCAN

AUTHOR: W. J. Erickson

DATE: November 1973

PURPOSE: Locate the row of a matrix which has the specified occurrence of an item.

USAGE:
```
DIMENSIØN MAT(MATDIM)
CALL ESCAN(MAT,MASK,MATDIM,NØCCUR,ITEM,NRØW)
```

PARAMETERS: I N P U T

MAT - Matrix to be searched
MASK - Masking constant
MATDIM - Dimension of MAT
NØCCUR - Occurrence of item to be found
ITEM - Item to be found

O U T P U T

NRØW - Row number found or 0

COMMON: None

SUBROUTINES: None

LANGUAGE: CØMPASS

DISCUSSION: By specifying the mask, items packed in a word are located.

ROUTINE: ESØRT1

ENTRY: BSØRT1, ESET1, ESØRT1

AUTHOR: Unknown

DATE: Unknown

PURPOSE: To sort a vector in non-descending order.

USAGE:
```
DIMENSIØN V(N)
CALL ESØRT1(V,N)
CALL BSØRT1(V,N)
CALL ESET1(K)
```

PARAMETERS:

I N P U T

V - Vector to be sorted
K - New criterion for bubble sort
N - Number of elements in vector

O U T P U T

V - Sorted vector

COMMON: None

SUBROUTINES: None

LANGUAGE: CØMPASS

DISCUSSION: The algorithm used in ESØRT1 is an adaptation of that due to M. H. Vanemden. (See article on sorting by William A. Martin in ACM Computing Surveys, Vol. 3, No. 4, (Dec. 1973) pg. 147-174). BSØRT1 sorts the vector by using the bubble sort technique. Entry point ESET1 alters the minimum vector length for which the Vanemden sort is used from a default of 25 to K.

ROUTINE: FILEADD

AUTHOR. G. von Limbach

DATE: December 1973

PURPOSE: To open files for access by ATLAS overlays.

USAGE:
```
DIMENSIØN FET(    )
CALL FILEADD(FET,FILE1,...,FILEN)
```

PARAMETERS: I N P U T

FET - Array dimensioned 100*N for FET tables
FILE1 - File names (either integer or L format, N≤20)
FILEN

COMMON: See discussion.

SUBROUTINES: LØCF,FETEDIT,NPARMS,SYSTEM

LANGUAGE: FØRTRAN

DISCUSSION: FETADD is used by FILEADD to open unblocked files. Files which do standard FØRTRAN I/Ø (READ,WRITE) should use FETEDIT to open files with appropriate buffer sizes and blocking parameters.

ROUTINE:	FILEDEL
AUTHOR:	G. von Limbach
DATE:	December 1973
PURPOSE:	Delete and close all files not on the program card of ATLAS.
USAGE:	CALL FILEDEL
PARAMETERS:	None
COMMON:	None
SUBROUTINES:	LØCF,CLØSER,FETDEL
LANGUAGE:	FØRTRAN
DISCUSSION:	The routine is called by ATLAS after calling each primary overlay.

ROUTINE: FINDIFU

AUTHOR: F. Nelson

DATE: December 1973

PURPOSE: Find an internal element number from a user element number

USAGE:
```
LØGICAL MATCHED
DIMENSIØN KLCT00(1)
CALL FINDIFU (KLCT00,NFT,NØELE,IELENØ,MATCHED)
```

PARAMETERS:

I N P U T

KLCT00 - Data portion of KLCT00a matrix
NFT - Row dimension of KLCT00
NØELE - User element number

O U T P U T

IELENØ - Internal element number (if any)
MATCHED - True if NØELE appears in KLCT00
False if NØELE appears in KLCT00

COMMON: None

SUBROUTINE: None

LANGUAGE: FØRTRAN

DISCUSSION: Given a candidate user element number, the routine performs a binary search for the user element number in matrix KLCT00a to attempt a match. If successful, the match flag is set TRUE and the corresponding internal element number is returned. If unsuccessful, the match flag is set FALSE.

ROUTINE: FLIP

AUTHOR: G. von Limbach

DATE: October 1973

PURPOSE: Transform a vector given a transformation matrix from local to global.

USAGE: CALL FLIP(C,G,P,X)

PARAMETERS: I N P U T

C - Appropriate column of local coordinate system matrix
G - Original vector in local frame
P - Nodal coordinates

O U T P U T

X - Transformed vector in global frame

COMMON: None

SUBROUTINES: GRAB

LANGUAGE: FØRTRAN

ROUTINE: FLØP

AUTHOR: G. von Limbach

DATE: October 1973

PURPOSE: Transform a vector given a transformation matrix from global to local.

USAGE: CALL FLØP(C,G,P,X)

PARAMETERS: I N P U T

C - Appropriate column of local coordinate system matrix
G - Original vector in global frame
P - Nodal coordinates

O U T P U T

X - Transformed vector in local frame

COMMON: None

SUBROUTINES: GRAB

LANGUAGE: FØRTRAN

ROUTINE: GMISRT

AUTHOR: H. Hansteen

DATE: October 1974

PURPOSE: To define an entry for a new user matrix in the ATLAS user matrix name catalog.

USAGE:
```
CALL GMISRT(IRR,MNAME,FNAME,NRØW,NCØL,MTYP,
1     NDTYP,MBLS,NMBL)
```

PARAMETERS:

I N P U T

MNAME - Name of new matrix
FNAME - File name where matrix will be stored (L-format)
NRØW - Matrix row dimension
NCØL - Matrix column dimension
MTYP - Matrix type indicator
=1 - rectangular matrix
=2 - symmetric matrix
=3 - diagonal matrix
=4 - null matrix
NDTYP - Storage format (=1)
MBLS - Maximum matrix block size
NMBL - Number of matrix blocks

O U T P U T

IRR - Error flag
= 0 - standard return
=-2 - matrix with same name has been over-written.
=-3 - illegal filename (not an ATLAS file)
=-4 - errors in remaining argument list
=-5 - checksum error for disk read operation

COMMON: CØMMØN/MATTAB/ - ATLAS executive system common block
CØMMØN/KQRNDM/ - ATLAS executive system common block

SUBROUTINES: INCRB,SHIFT,REDER,WRTER,LCKSUM

LANGUAGE: FØRTRAN

DISCUSSION: The user matrix name catalog is generally saved on the system file SYSRNF. If the size of the catalog is small, however, a copy of the catalog is kept in core only. GMISRT works as follows:

a) It checks if the specified matrix name is previously defined in the catalog. If so it returns with the error flag set.

b) It checks that the specified file name coincides with one of the ATLAS data files. Error return if check is negative.

c) Checks range of remaining parameters as follows:

$0 \leq NRØW \leq 2^{19}-1$
$0 \leq NCØL \leq 2^{18}-1$
$1 \leq MTYP \leq 3$
NDTYP=1
$0 \leq MBLS \leq 2^{18}-1$
$0 \leq NMBL \leq 2^{18}-1$

Error return if check is negative.

d) Generates the new entry into the catalog.

No error messages are printed from the routine.

ROUTINE: GMLKUP

AUTHOR: H. Hansteen

DATE: October 1974

PURPOSE: To extract a specified entry in the ATLAS user matrix name catalog

USAGE: CALL GMLKUP(IRR,MNAME,FNAME,NRØW,NCØL,MTYP,NDTYP,
1 MBLS,NMBL)

I N P U T

MNAME - Name of matrix for which information is sought

INPUT/OUTPUT

IRR(in) - = 1- Checks if matrix exists and returns output parameters
≠ 1- Checks if matrix exists only

IRR(out) - = 0- Standard return, matrix found
=-1- Matrix name not in catalog
=-2- Catalog empty
=-5- Checksum error for disk read operation

O U T P U T

FNAME - File name where matrix is stored
NRØW - Row dimension of matrix
NCØL - Column dimension of matrix
MTYP - Matrix type indicator
= 1 - rectangular matrix
= 2 - symmetric matrix
= 3 - diagonal matrix
= 4 - null matrix
NDTYP - Storage format (=1)
MBLS - Maximum matrix block size
NMBL - Number of matrix blocks

COMMON: CØMMØN/MATTAB/ - ATLAS executive system common block

SUBROUTINES: INCRB,SHIFT,REDER,LCKSUM

LANGUAGE: FØRTRAN

DISCUSSION: The user matrix name catalog is gener. lly saved on che system file SYSkNF. If the size of the catalog is small, however, a copy of the catalog is kept in core only.

No error messages are printed from the routine.

ROUTINE: GRAB

AUTHOR: G. von Limbach

DATE: September 1973

PURPOSE: To compute the global-to-local transformation matrix for a given local coordinate system at a given point.

USAGE:
```
DIMENSION A(13),X(3),T(3,3)
CALL GRAB(A,X,T)
```

PARAMETERS: I N P U T

A - Mass or stiffness local coordinate matrix.
X - Global coordinates of point

O U T P U T

T - Transformation matrix V(local) = TV(global)

COMMON: None

SUBROUTINE: None

LANGUAGE: FORTRAN

DISCUSSION: For rectangular systems the coordinate transformation is constant and that at the origin is used. For cylindrical and spherical systems the coordinate system at the point is chosen to be right handed with the x axis along the local R axis and y and z axes tangential to constant local coordinate lines. Thus the mapping from a point to the coordinate system is continuous except for the poles of the spherical system where arbitrarily the y axis is the local x at the north pole and the local -x at the south pole.

ROUTINE: HERMINT

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: Perform Hermite interpolation for F(X), X1≤X≤X2, given F(X1), F(X2), DF(X1)/DX, and DF(X2)/DX, for N modes

USAGE:
```
      DIMENSION FX(NR1,N),FX1(NR2,N),DFX1(NR2,N),
     1          FX2(NR2,N),DFX2(NR2,N)
      CALL HERMINT(X,FX,NR1,X1,FX1,DFX1,X2,FX2,DFX2,NR2,N)
```

PARAMETERS: I N P U T

X	-	Values for which interpolated results are desired
NR1	-	Row dimension of FX array
X1,X2	-	Independent variable values bracketing X.
FX1	-	The dependent variable array corresponding to X1
FX2	-	The dependent variable array corresponding to X2
DFX1	-	The array of slopes at X1
DFX2	-	The array of slopes at X2
NR2	-	Row dimension of FX1,FX2,DFX1, and DFX2
N	-	The number of columns in FX1,FX2, DFX1, DFX2, and FX

O U T P U T

FX	-	The interpolated results at X for N columns

COMMON: None

SUBROUTINES: None

LANGUAGE: FORTRAN

DISCUSSION: This subroutine calculates the function

$$FX(I) = C1*(C2*C3*DFX1(I) + C2*C4*FX1(I) - C5*C6*DFX2(I) + C5*C7*FX2(I))$$

Where

$$C1 = \frac{1}{(X2-X1)^3}$$

$$C2 = (X2-X)^2$$
$$C3 = (X-X1)*(X2-X1)$$
$$C4 = 2*(X-X1) + (X2-X1)$$
$$C5 = (X-X1)^2$$
$$C6 = (X2-X)*(X2-X1)$$
$$C7 = 2*(X2-X) + (X2-X1)$$

ROUTINE: IDCHECK

AUTHOR: F. P. Gray

DATE: September 1972

PURPOSE: Check element data for duplicated user labels and print duplicates found.

USAGE: CALL IDCHECK(MELNØ,IWARN)

PARAMETERS: I N P U T

MELNØ - Element nodal matrix (KMELNØ,MMELNØ)

O U T P U T

IWARN - 0 no duplicate labels found
1 duplicate labels exist

COMMON: None

SUBROUTINES: SHIFT

LANGUAGE: FØRTRAN

ROUTINE: IFIND

AUTHOR: W. J. Erickson

DATE: October 1975

PURPOSE: To determine whether a number is contained in an array.

USAGE:
```
DIMENSION A(N)
I=IFIND(A,AI,N)
```

I N P U T

A - Array in which to search
AI - Number being sought
N - Number of words in A to search

O U T P U T

I - Result
= 0 AI not present in A
= I AI is equal to A(I)

COMMON: None

SUBROUTINES: None

LANGUAGE: COMPASS

ROUTINE: INCBCD

AUTHOR: C. A. Felippa and L. J. Schmid

DATE: December 1969

PURPOSE: To increment the numerical portion of a left-adjusted BCD item of the form, AAAXXXXFFF, where A..A is a non-numeric portion, X...X is the numeric portion and F..F is right fill of the word. The first two portions must be present. X...X may contain only characters 0,1,...,9.

USAGE: CØMMØN/LRCØMM/NINT,NØUT,NØUR,CCC(2)
A = INCBCD (ØLDBCD,INC,ØLDNØC,NEWNØC,LØØP,KERR)

PARAMETERS:

I N P U T

ØLD	-	BCD item to be incremented
INC	-	Integer increment
ØLDNØC	-	Number of characters in ØLDBCD, excluding right fill of word
NINT	-	See
NØUT	-	description
NØUR	-	in
CCC	-	LØDAREC

INPUT/OUTPUT

LØØP(in)	-	If LØØP = 0, decoding of numeric portion of ØLDBCD is skipped (can be used when INCBCD is called in a loop to recursively increment a BCD item).
LØØP(out)	-	LØØP = 1 unless an error condition has occurred.
KERR	-	See description in LØDAREC

O U T P U T

A	-	Incremented BCD item
NEWNØC	-	Number of characters in A

COMMON: CØMMØN/LRCØMM/NINT,NØUT,NØUR,CCC(2)

SUBROUTINES: GETT,STRMØV

LANGUAGE: FØRTRAN

ROUTINE: INCR

AUTHOR: W. J. Erickson

DATE: August 1974

PURPOSE: To increment the specified numeric portion of a 7 character record name

USAGE: N = INCR(INDEX,ILCB,IRCB,IDEL)

PARAMETERS: I N P U T

INDEX - Record name to be incremented
ILCB - Leftmost character position of numeric portion
IRCB - Rightmost character position of numeric portion
IDEL - Size of increment

O U T P U T

N - Incremented record name

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: ILCB,IRCB restricted to $2 \leq ILCB \leq IRCB \leq 7$

ROUTINE: INCRB

AUTHOR: H. Hansteen

DATE: August 1973

PURPOSE: To increment a fixed point number which occupies bit positions 6 through 17 of a word. The routine was specifically designed to modify the record number field of an ATLAS random index.

USAGE: IB = INCRB(IA,IDEL,IRR)

I N P U T

IA - Index name to be modified
IDEL - The positive or negative index modifier

O U T P U T

IB - The modified index name
IRR - Error flag
= 0 - standard return
= 2 - increment value exceeds range of numeric field (|IDEL|≥4096)
= 3 - incremented index exceeds range of numeric field (≥4096)

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: The position of the numeric field is set internally in the routine to allow future redefinition of the ATLAS random index format.

If an error condition is encountered the output variable remains unchanged. No error messages are printed.

ROUTINE: ISUB

AUTHOR: Harold Hansteen

DATE: July 1975

USAGE: J = ISUB(SUB,N)

PARAMETERS: I N P U T

SUB - Element or nodal subset matrix

N - Internal element or node number

O U T P U T

J - 0 N is not in the subset
1 N is in the subset

COMMON: None

SUBROUTINES: SHIFT

LANGUAGE: FØRTRAN

ROUTINE: ITEMLST

AUTHOR: Sharad H. Gadre

DATE: September 1975

PURPOSE: Load values into the ATLAS data directory matrix.

USAGE: CALL ITEMLST(MAT)

PARAMETERS: O U T P U T

MAT - A 2-column matrix with the data directory packed in it. The number of rows is currently 405.

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

ROUTINE: JACØBI

AUTHOR: R. L. Dreisbach

DATE: February 1974

PURPOSE: Compute eigenvalues and/or eigenvectors of a real symmetric matrix via the threshold Jacobi diagonalization method. Optionally, each eigenvector may be normalized relative to the largest element in the vector; the magnitude of the largest element in each of the resulting vectors is ±1.0.

USAGE:
```
DIMENSIØN A(N,N),B(N,N)
CALL JACØBI(A,B,N,ØPTV)
```

PARAMETERS:

I N P U T

A - Matrix for which an eigensolution is to be performed. This matrix is destroyed during computation. Resultant eigenvalues are generated in the diagonal of this matrix.

N - Order of matrices A and B

ØPTV - Eigenvalue/eigenvector computation options.
1 - Compute eigenvalues only
2 - Compute eigenvalues and eigenvectors
3 - Compute eigenvalues and normalized eigenvectors

O U T P U T

A - Resultant eigenvalues stored on diagonal.

B - Resultant matrix of eigenvectores stored columnwise in sequence corresponding to eigenvalues.

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: If an off-diagonal element of A is greater than the threshold, it is annihilated via a plane rotation transformation. This process is continued until each off-diagonal element is less than a final norm defined as follows:

RNØRMI - Off-diagonal norm. Initially, this is the square root of the sum of squared off-diagonal elements of A.
FN - Final norm equal to (RNØRMI*10^{-6}/N)
THRESH - Threshold defined as (RNØRMI/N^3). If after a complete sweep, THRESH is greater than FN, THRESH is re-set to (THRESH/N^3) and another sweep is performed.

The resulting matrix is the diagonal matrix of eigenvalues D. B is the multiple product of orthogonal matrices which result in the following relationships:

B(T)*A*B=D
A*B=B*D
A=B*D*B(T)

Thus, A^{-1}=B*D^{-1}*B(T)

Reference: A. Ralston, A First Course in Numerical Analysis, McGraw-Hill Book Co., 1965.

ROUTINE: KSFREØR

AUTHOR: K. Yagi

DATE: October 1974

PURPOSE: To reorder KSF matrices.

USAGE:
```
DIMENSIØN CØRR(N),PØS(2)
CALL KSFREØR (INFILE,ININDEX,ØUTFILE,ØUTIND,CØRR,
1             N,PØS,IRR)
```

PARAMETERS:

I N P U T

INFILE - The file name of the input KSF matrices

ININDEX - The index name of the first input matrix

CØRR - The correspondence table which holds the new order for the KSF matrices. J in row I means the new Jth element is the old Ith element.

N - The number of elements to be reordered.

PØS - An array in which PØS (1) and PØS(2) each contain available position references for use by KSFREØR.

O U T P U T

ØUTFILE - The file name of the reordered KSF matrices.

ØUTIND - The index name for the first reordered KSF matrix.

IRR - Error flag
=0 - standard return
=1 - SNARK error encountered

COMMON: None

SUBROUTINES: ELEPICK,ELEPUT

LANGUAGE: FØRTRAN

DISCUSSION: It is not advisable to use the same file and index names for the input and output matrices.

This routine is coded using the SNARK language for matrix manipulation.

ROUTINE:	LINLMT
AUTHOR:	W. J. Erickson
DATE:	June 1976
PURPOSE:	To make line limit exceeded a non-fatal error.
USAGE:	CALL LINLMT
PARAMETERS:	None
COMMON:	None
SUBROUTINES:	SYSTEMC
LANGUAGE:	FØRTRAN

ROUTINE: LØDAREC

AUTHORS: C. A. Felippa
L. J. Schmid

DATE: November 1970

PURPOSE: To read and decode one free field data record (for data record conventions see sec. 100, ref. 1-1)

USAGE:
```
CØMMØN/CARDS/REC(250),ID(250),KRECS,KCDS,ITEMS,
1     KERR(3),JBL
CØMMØN/LRCØMM/NINT,NØUT,NØUR,CCC(2)
DIMENSIØN IREC(250)
EQUIVALENCE(REC,IREC)
CALL LØDAREC
```

COMMON:
```
CØMMØN/CARDS/REC(250),ID(250),KRECS,KCDS,ITEMS,
1     KERR(3),JBL
CØMMØN/LRCØMM/NINT,NØUT,NØUR,CCC(2)
```

PARAMETERS:

I N P U T

JBL - a) If JBL=0, alphanumeric items are stored left-adjusted with zero fill.
b) If JBL=(blank), alphanumeric items are stored left-adjusted with blank fill.
Default value for JBL is blank.

NINT - File name/number of the card input file from which LØDAREC reads data records Default is NINT=5.

NØUT - File name/number of a coded output file onto which LØDAREC writes the image of the data input cards. Default is NØUT=6. If NØUT=0 the data records are not written (useful for teletype work)

NØUR - File name/number of a coded output file onto which error diagnostics are written. Default is NØUR=6.

INPUT/OUTPUT

KRECS - The data record counter. Empty records (comments only) are not counted. Generated records are counted.

KCDS - The data card counter

ITEMS - Counter of decoded data items loaded in REC

REC - Vector into which LØDAREC loads decoded card data. On input REC should contain the previously decoded record. Upon return REC contains the new items. LØDAREC sets REC (ITEMS+1)...REC(250) to zero.

ID - Vector into which LØDAREC loads identification for each data item. If data item is;
=5HFIXED - fixed point number
=8HFLØATING - floating point number
=N - alphanumeric string,
where N is the number of characters in the string. If N>10,ID=10.
LØDAREC sets ID(ITEMS+1)...ID(250) to zero.

CCC - Array used for internal communication between LØDAREC and routines DECØDIR and INCBCD.

O U T P U T

KERR(1) - Count of number of items in the current record which caused fatal error 78, illegal data in field, on decoding.

KERR(2) - Count of all other errors detected by LØDAREC

KERR(3) - Set to 1 whenever fatal error 78 occurs

SUBROUTINES: DECØDIR,LØCF,GETT,STRMØV,INCBCD,CHKEØR

LANGUAGE: FØRTRAN

ROUTINE: MAATTCH

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: Given, the definition of a motion axis; For an arbitrary point, determine the associated motion axis reference point, the distance from the point to the reference point, and the arc length along the motion axis to the reference point.

USAGE:
```
DIMENSION XMA(NMADP),YMA(NMADP),DYDXRL(NMADP),
1         SMA(NMADP),CMA(NSEG,4),XMAP(NSEG),
1         YMAP(NSEG)
CALL MAATTCH (X,Y,XMA,YMA,DYDXRL,SMA,NMADP,NSEG,
1             CMA,XMAP,YMAP,XR,DR,SR,THETARL)
```

PARAMETERS: I N P U T

X	-	x-coordinate of arbitrary point
Y	-	y-coordinate of arbitrary point
XMA	-	x-coordinates of motion axis definition points
YMA	-	y-coordinates of motion axis definition points
DYDXRL	-	dy/dx of reference lines through each motion axis definition point
SMA	-	Arc length along motion axis to definition point
NMADP	-	Number of motion axis definition points
NSEG	-	Number of motion axis segments (=NMADP-1)
CMA	-	Cubic coefficients in Y defining the motion axis on each segment
XMAP	-	x-coordinate of mapping point for a motion axis segment
YMAP	-	y-coordinate of mapping point for a motion axis segment (Note - XMAP(I),YMAP(I) is the intersection of the reference line through XMA(I),YMA(I) defined by DYDXRL(I) and the reference line for I+1)

O U T P U T

XR	-	x-coordinate of reference point on motion axis
YR	-	y-coordinate of reference point on motion axis
DR	-	Distance from (X,Y) to (AR,YR)
SR	-	Arc length along motion axis to (XR,YR)
THETARL	-	Angular difference between the reference line and tree stream

COMMON: None

SUBROUTINES: REFPT,ARCL

LANGUAGE: FØRTRAN

DISCUSSION: The motion axis segment associated with (X,Y) is determined. The segment is the first for which the distance from the reference line at the outboard definition point to (X,Y) is negative. The cubic coefficients for the segment are accessed, or calculated if the attachment will be inboard of the first or outboard of the last definition point. In the latter cases a linear extension of the motion axis definition at the definition point is made. Routine REFPT is called to determine (XR,YR), routine ARCL is called to determine SR, and DR and THETARL are calculated within MAATCH.

ROUTINE: MAMULT

AUTHOR: L. C. Carpenter

DATE: December 1973

PURPOSE: Perform one of the following matrix products:

1. C = C+AB
2. C = C-AB
3. C = AB
4. C = -AB

USAGE:
```
DIMENSION C(M,N),A(M,P),B(P,N)
COMMON /MACOM/ID,M,N,P,CROWSP,AROWSP,BROWSP,CCOLSP,
1     ACOLSP,BCOLSP
CALL MAMULT(C,A,B)
```

PARAMETERS:

I N P U T

A,B - Factor matrices

O U T P U T

C - Result matrix

COMMON:
```
COMMON/MACOM/ID,M,N,P,CROWSP,AROWSP,BROWSP,CCOLSP,
1     ACOLSP,BCOLSP
```

variables:

ID	-	1,2,3,4 selects option above
M	-	Number of rows in C and A
N	-	Number of columns in C and B
P	-	Number of columns in A and rows in B
CROWSP AROWSP BROWSP CCOLSP ACOLSP BCOLSP	-	Skip factors for locating next row/column where XROWSP = Number of elements to skip to get to a new row in matrix X, and XCOLSP = Number of elements to skip to get to a new column in matrix X.

SUBROUTINES: None

LANGUAGE: COMPASS

DISCUSSION: MAMULT is optimized for outer products, where P is small and N is large and M is between.

TIME = 2.3*M*N*P + M + 50 Microseconds

ROUTINE: MASKIT

AUTHOR: L. J. Schmid

DATE: December 1969

PURPOSE: To convert a left adjusted, blank filled portion of a word to zero fill.

USAGE: N = MASKIT(NAMEIN)

PARAMETERS: I N P U T

NAMEIN - The word with right adjusted blank fill

O U T P U T

N - The same word with right adjusted zero fill

SUBROUTINES: SHIFT

COMMON: None

LANGUAGE: FORTRAN

ROUTINE: MATAINT

AUTHOR: S. Wahlstrom

DATE: October 1970

PURPOSE: Calculate interpolated standard material properties for a given temperature

USAGE: I = MATAINT(ITEMP,PRØPMAT,NRØWS,IFIRST,ILAST,RESULT)

PARAMETERS: INPUT

ITEMP - Temperature for which properties are desired
PRØPMAT - Matrix of material properties
NRØWS - Number of temperatures (row dimension) in PRØPMAT
IFIRST - First property desired
ILAST - Last property desired

OUTPUT

RESULT - Vector of interpolated properties, at least (ILAST-IFIRST+1) elements long
I - Error flag
=0, Normal return
=1, Temperature out of range
=2, IFIRST>ILAST

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: No extrapolation is allowed

ROUTINE: MATCINT

AUTHOR: J. M. Held

DATE: April 1975

PURPOSE: Calculate interpolated composite material properties for a given temperature

USAGE: I = MATCINT(TEMP, CMAT, MCØDE, NFIRST, NLIST, RESULT)

PARAMETERS: INPUT

T - Temperature for which properties are desired
CMAT - Composite material matrix
MCØDE - Material code for current material
NFIRST - First property desired
NLAST - Last property desired

O U T P U T

RESULT - Vector of interpolated properties
I - >0, Thickness of lamina * 1000
-1, NFIRST>NLAST or
NFIRST<1 or
NLAST> allowable
-3, Temperature out of range
-4, MCØDE not present in CMAT

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: No extrapolation is allowed

ROUTINE: MATEXP

AUTHOR: L. J. Schmid

DATE: December 1969

PURPOSE: To rewrite a matrix within a given storage area so it will conform with new dimensions to be allocated to it.

USAGE: CALL MATEXP(MAT1,MR1,MC,MAT2,MR2)

PARAMETERS: I N P U T

MAT1 - The old matrix
MR1 - Row dimension of MAT1
MC - Column dimension of MAT1 and MAT2
MR2 - Row dimension of MAT2

O U T P U T

MAT2 - The compressed/expanded version of MAT1

COMMON: None

SUBROUTINE: None

LANGUAGE: FØRTRAN

DISCUSSION: If MR2 < MR1 it is assumed that MAT1 contains (MR1-MR2+1) zero rows at the lower end of the matrix. MAT2 is then formed by deleting these rows. If MR2 > MR1, MAT2 is formed by adding MR2-MR1+1 zero rows to the existing rows of MAT1.

ROUTINE: MØTAXØ

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: Given an interpolation coefficient array for a motion axis system generated in routine MØTAXI, calculate modal displacements and slopes at a set of output points.

USAGE:
```
DIMENSIØN X(NPTS),Y(NPTS),Z(NRØWZ,NMØDES),SA(NVARY),
1     DZ1(NRØWZ,NMØDES),DZ2(NRØWZ,NMØDES)
CALL MØTAXØ (X,Y,NPTS,Z,NRØWZ,NCØL1,NCØLS,SA,INDD,
1     DZ1,DZ2)
```

PARAMETERS:

I N P U T

NPTS	-	Number of output points
X,Y	-	Coordinates of output points
NRØWZ	-	Row dimension of Z,DZ1,DZ2
NCØL1	-	First input column to be used
NCØLS	-	Number of columns to be used - if NCØLS=0, all columns ≥NCØL1 will be used
SA	-	Interpolation coefficient array
INDD	-	Derivative option indicator, =0, no derivatives, DZ1,DZ2 not required =1, DZ1=DZ/DX, DZ2 not required =2, DZ1=DZ/DY, DZ2 not required =3, DZ1=DZ/DX, DZ2=DZ/DY

O U T P U T

Z	-	Modal displacements at output points
NCØLS	-	Number of columns returned
DZ1,DZ2	-	Derivatives as indicated by INDD above
INDD	-	Error return ≥0, No error =10H*MØTAXØ 1, SA has wrong format =10H*MØTAXØ 2, illegal number of columns requested =10H*MØTAXØ 3, illegal number of output points, NPTS≤0 =10H*MØTAXØ 4, illegal row dimension for Z, NRØWZ<NPTS

COMMON: None

SUBROUTINES: ZERØCØL,MAATTCH,HERMINT

LANGUAGE: FØRTRAN

DISCUSSION: For each point specified, the routine:

1. Calls MAATTCH, which determines motion axis reference point, arc length along motion axis to reference point, and angle from reference line to free stream.

2. Determines associated motion station segment of motion axis.

3. Calculates displacements and rotations at reference point for all modes:

   a. If reference point arc length is not within (S(1),S(NMS)) - use linear extrapolation for displacements and endpoint S(1) or S(NMS) values for rotations.

   b. If reference point arc length is within (S(1),S(MNS)) - call HERMINT for Hermite interpolation.

4. Generates displacement and (optionally) slopes at the output points.

NMØDES, used in the DIMENSIØN statement is given by:

$$NMØDES \geq NCØLS-NCØL1+1$$

ROUTINE: MØTPTØ

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: Given an interpolation coefficient array generated in MØTPTI associated with the motion of a rigid body datum, calculate the displacements at a set of output points.

USAGE:
```
DIMENSIØN  X(MPTS),Y(MPTS),Z(MPTS),DELZ(NRØWZ,NMØDES),
1          SA(NVARY),GAMMA(MPTS),DZ1(NRØWZ,NMØDES),
1          DZ2(NRØWZ,NMØDES)
CALL MØTPTØ(X,Y,Z,MPTS,DELZ,NRØWZ,NCØL1,NCØLS,SA,
            INDG,GAMMA,INDD,DZ1,DZ2)
```

PARAMETERS: I N P U T

X,Y,Z	-	Coordinates of output points.
ABS(MPTS)	-	Number of output points. If MPTS<0, Z is assumed to be zero for all points.
NRØWZ	-	Row dimension of DELZ, DZ1, DZ2
NCØL1	-	First input column to be used
NCØLS	-	Number of columns to be used. If NCØLS≤0, all columns ≥NCØL1 will be used
SA	-	Interpolation coefficient array
INDG	-	Indicator for orientation of normal displacement =0, same as global Z, no GAMMA used =1, angle GAMMA(1) applies to all points =2, angle GAMMA(I) applies to point I
GAMMA	-	Angular orientation (radians positive in positive X sense) of normal displacements, as indicated by INDG
INDD	-	Derivative option indicator =0, no derivatives, DZ1 and DZ2 not required =1, DZ1=DZ/DX, DZ2 not required =2, DZ1=DZ/DY, DZ2 not required =3, DZ1=DZ/DX, DZ2=DZ/DY

O U T P U T

DELZ - Normal displacements at output points
NCØLS - Number of columns returned
DZ1,DZ2 - Derivatives as specified by INDD
INDD - Error indicator,
≥0 - No error
= 10H*MØTPTØ 1, SA has wrong format
= 10H*MØTPTØ 2, illegal number of output points, MPTS=0
= 10H*MØTPTØ 3, illegal row dimension for DELZ, (NRØWZ<ABS(MPTS))
= 10H*MØTPTØ 4, illegal number of columns requested

COMMON: None

SUBROUTINE: ZERØCØL

LANGUAGE: FØRTRAN

DISCUSSION: For each output point specified, the routine generates displacements and (optionally) slopes by a rigid transformation of the motion of a datum.

NMØDES, used in the DIMENSIØN statement is given by:

NMØDES ≥ NCØL-NCØL1+1

ROUTINE: MØVRØW

AUTHOR: L. J. Schmid

DATE: December 1969

PURPOSE: To copy a submatrix of a matrix into another matrix.

USAGE:
```
DIMENSIØN  MAT1(MRS1,1),MAT2(MRS2,1)
CALL  MØVRØW (MAT1,MRS1,MAT2,MRS2,I1,J1,I2,J2,LRS,LCS)
```

PARAMETERS: I N P U T

MAT1 - Source matrix
MRS1 - Row size of MAT1
MRS2 - Row size of MAT2
I1 - Row location INMAT1 of first element to be moved
J1 - Column location in MAT1 of first element to be moved
I2 - Row location in MAT2 of first element of submatrix
J2 - Column location in MAT2 of the first element of submatrix
LRS - Number of rows to be moved
LCS - Number of columns to be moved

O U T P U T

MAT2 - Result matrix

COMMON: None

SUBROUTINE: None

LANGUAGE: FØRTRAN

DISCUSSION: The appropriate elements are copied from MAT1 to MAT2 using a double do loop.

ROUTINE: MØVSTG

AUTHOR: O. L. Anderson

DATE: May 1969

PURPOSE: Move the contents of a block of core to a new location.

USAGE: CALL MØVSTG(TØ,FIRST,LAST)

PARAMETERS: I N P U T

TØ - New location of first word of block
FIRST - First word of block
LAST - Last word of block

COMMON: None

SUBROUTINES: None

LANGUAGE: CØMPASS

DISCUSSION: This routine is the same as WØMSTG, part of the SNARK package.

Restriction: FIRST ≤ LAST

ROUTINE: NBITS1

ENTRY: NBITS1,NBIT1

AUTHOR: O. L. Anderson

DATE: May 19[illegible]

PURPOSE: Entry NBITS1 - To count the number of 1 bits of a partial word.
Entry NBIT1 - To count all 1 bits of a word.

USAGE: N=NBITS1(WØRD,I,J)
N=NBIT1(WØRD)

PARAMETERS: I N P U T

WØRD - Word in which 1 bits are counted.
I - Bit to start counting (≤59)
J - Bit to stop counting (I≥J≥0)

O U T P U T

N - Number of 1 bits in specified portion of word

CØMMØN: None

SUBROUTINES: None

LANGUAGE: CØMPASS

DISCUSSION: If 59≥I≥J≥0 is not satisfied, all 1 bits are counted.

ROUTINE: NCUIR

ENTRY: NCUIR,NCIUR

AUTHOR: G. von Limbach

DATE: May 1973

PURPOSE: NCUIR - Obtain the row in the nodal data table and the internal node number, given the user number
NCIUR - Obtain row and user number, given internal number

USAGE:
```
DIMENSION NCT( )
CALL NCUIR(NCT,NUSER,NINT,IRØW)
CALL NCIUR(NCT,NINT,NUSER,IRØW)
```

PARAMETERS:

I N P U T

NCT - Nodal correspondence table

I N P U T / O U T P U T

NUSER - User node number
NINT - Internal node number

O U T P U T

IRØW - Row pointer to nodal data table

COMMON: None

SUBROUTINES: None

LANGUAGE: COMPASS

DISCUSSION: If NUSER is not defined, NINT and IRØW are set to zero.

ROUTINE: NØDPRNT

AUTHOR: L. J. Schmid

DATE: December 1969

PURPOSE: Print the nodal data matrix

USAGE:
```
DIMENSIØN XX(MR,MC),NCT(NCTR)
CALL NØDPRNT(XX,MR,MC,NCT,NCTR)
```

PARAMETERS: I N P U T

XX - The nodal data matrix
MR - Row dimension of XX
MC - Column dimension of XX
NCT - The nodal correspondence table
NCTR - Row dimension of NCT

COMMON: None

SUBROUTINES: SHIFT

LANGUAGE: FØRTRAN

ROUTINE: ØPTFØR

AUTHOR: S. Wahlstrom

DATE: September 1970

PURPOSE: Compute the optimum format for a given field width and number

USAGE: CALL ØPTFØR(A,W,F)

PARAMETERS: I N P U T

A - Some real number
W - Field width (integer number)

O U T P U T

F - Format as a Hollerith constant

COMMON: None

SUBROUTINES: ABS

LANGUAGE: FØRTRAN

DISCUSSION: ØPTFØR can be used when setting up a variable format in an array. F has the format for printing A using W columns. The last character in F is a comma.

ROUTINE: PLATEØ

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: Given the spline coefficients for a set of functions as determined in routine PLATEI, and the associated input points, calculate the values of the functions and their derivatives at a set of output points.

USAGE:
```
DIMENSIØN XØ(NØPTS),YØ(NØPTS),ZØ(NRØWZ,NMØDES),
1         SA(NVARY),DZ1(NRØWZ,NMØDES),DZ2(NRØWZ,NMØDES)
CALL PLATEØ (XØ,YØ,NØPTS,ZØ,NRØWZ,NCØL1,NCØLS,
1         SA,INDD,DZ1,DZ2)
```

PARAMETERS:

I N P U T

NØPTS	-	Number of output points
XØ,YØ	-	Coordinates of output points
NRØWZ	-	Row dimension of ZØ.DZ1,DZ2
NCØL1	-	First input column to be used
NCØLS	-	Number of columns =0 - all columns ≥ NCØL1
SA	-	Spline coefficient array as generated in PLATEI
INDD	-	Derivative option indicator =0, no derivatives, DZ1,DZ2 not required =1, DZ1=DZ/DX, DZ2 not required =2, DZ1=DZ/DY, DZ2 not required =3, DZ1=DZ/DX, DZ2=DZ/DY

O U T P U T

ZØ	-	Interpolated function values
NCØLS	-	Number of columns returned
INDD<0	-	Error condition 10H*PLATEØ 1 = spline scratch array has wrong format 10H*PLATEØ 2 = illegal number of columns requested
DZ1	-	Not produced if INDD=0 DZ/DX if INDD=1 or 3 DZ/DY if INDD=2
DZ2	-	Not produced if INDD=0,1 or 2 DX/DY if INDD=3

COMMON: None

SUBROUTINES: ZERØCØL, VIP

LANGUAGE: FØRTRAN

DISCUSSION: NMØDES, used in the DIMENSIØN statement is given by:

$$NMØDES \geq NCØLS-NCØL1+1$$

ROUTINE: PØLYØ

AUTHOR: M. Y. Hirayama

DATE: August 1973

PURPOSE: To obtain displacements and slopes from coefficients of a polynomial with independent X and Y variables and a maximum order of 5.

```
DIMENSIØN X(NPTS),Y(NPTS),Z(NRØWZ,NMØDES),
1         DZ1(NRØWZ,NMØDES),DZ2(NRØWZ,NMØDES),A(NVARY)
CALL PØLYØ(X,Y,NPTS,Z,NRØWZ,NCØL1,NCØLN,A,INDD,DZ1,DZ2)
```

PARAMETERS:

I N P U T

X - x-coordinates (local) of output points
Y - y-coordinates (local) of output points
NPTS - number of output points
NRØWZ - row dimension of Z - must be ≥ NPTS
NCØL1 - number of modes
= 0, pick up number of modes in coefficient array
A - Coefficients array
INDD - Indicator for derivative output
= 0, no derivatives
= 1, dZ/dX
= 2, dZ/dY
= 3, both dZ/dX and dZ/dY

O U T P U T

Z - Matrix of displacements at output points
DZ1 - Not used if INDD = 0 or 2
dZ/dX if INDD = 1 or 3
DZ2 - Not used if INDD = 0 or 1
dZ/dY if INDD = 2 or 3
INDD - Error return
= 0 No error
= 10H*PØLYØ X, where
X = 1 - Polynomial name incorrect
2 - Too many modes specified
3 - NRØWZ < NPTS
4 - INDD <0 or >3

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: NMØDES, used in the DIMENSIØN statement is given by:

$$NMØDES \geq NCØLS-NCØL1+1$$

ROUTINE: PRNTØCT

AUTHOR: Unknown

DATE: Unknown

PURPOSE: Print a matrix in octal format with 5 numbers per line.

USAGE: CALL PRNTØCT (MAT,MATR,MATC)

PARAMETERS: I N P U T

MAT - Matrix to print
MATR - Row dimension
MATC - Column dimension

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

ROUTINE: REFPT

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: Find the intersection of a given straight line and a cubic.

USAGE:
```
CALL REFPT(X,Y,XM,YM,DYDXRL,YMA1,YMA2,C0,C1,C2,C3,
1     XR,YR)
```

PARAMETERS:

I N P U T

X	-	x-coordinate of arbitrary point
Y	-	y-coordinate of arbitrary point
XM	-	x-coordinate of mapping point
YM	-	y-coordinate of mapping point (Note YM = indefinite if (XM,YM) undefined)
DYDXRL	-	dY/dX of reference line if (XM,YM) defined
YMA1	-	y-coordinate of inboard motion axis definition point
YMA2	-	y-coordinate of outboard motion axis definition point
C0,C1,C2,C3	-	cubic coefficients in y defining motion axis on (YMA1,YNA2)

O U T P U T

XR	-	x-coordinate of reference point
YR	-	y-cocrdinate of reference point

COMMON: None

SUBROUTINE: None

LANGUAGE: FØRTRAN

DISCUSSION: Depending upon the values of C0,C1,C2,C3 the solutions to a cubic, quadratic, or linear equation are determined. That solution for which YMA1 ≤ YR ≤ YMA2 is used.

Cubic is: $x = C0*C1*y + C2*y^2 + C3*y^3$

ROUTINE: REØRSUB

AUTHOR: Kandace K. Yagi

DATE: August 1974

PURPOSE: Reorder a subset matrix.

USAGE: CALL REØRSUB (INSUB,ØUTSUB,CØRR,N)

PARAMETERS: I N P U T

INSUB - Old subset matrix
CØRR - Sorting correspondence where CØRR(J)=J means that new element J is the same as old element I
N - Number of elements

O U T P U T

ØUTSUB - Sorted subset matrix

COMMON: None

SUBROUTINES: SHIFT

LANGUAGE: FØRTRAN

ROUTINE: SCAMP4

AUTHOR: G. W. Erwin, Jr.

DATE: November 1962

PURPOSE: Fit a chain of cubic equations through a set of points

USAGE: DIMENSION X(N),Y(N),C(4,K) or C(7,K) where K≥N-1
CALL SCAMP4(X,Y,N,NDA,NDB,DA,DB,C,S,M)

PARAMETERS: I N P U T

- X - The array of x-coordinates
- Y - The array of y-coordinates
- N - The number of given points
- NDA - The order (first or second) of the derivative at X(1); input as a negative integer if the derivative is to be computed by SCAMP4
- NDB - The order of the derivative at X(N)
- DA - The value of the derivative at X(1); not used or changed if NDA <0
- DB - The value of the derivative at X(N),
- M≠12 - The cubic coefficients are to be stored in a four by (n-1) array
- M=12 - The cubic coefficients are to be stored in a singly subscripted array according to the (2D) composite curve format

O U T P U T

- C - The array of cubic coefficients
- S - The array of derivatives $y(i)^1$ at the given x(i)
- M - Error return
  =0 success
  1≤M≤7 implies error return from COMCU
  M large implies error return K on Jth cubic from CUBIC2 (M=100*J+K)
  =-1 implies N<2

COMMON: None

SUBROUTINES: COMCU,CUBIC2,DERIV1,DERIV2

LANGUAGE: FORTRAN

ROUTINE:	SCREW
AUTHOR:	Unknown
DATE:	Unknown
PURPOSE:	Print a graphical display on the output file to indicate the status of execution.
USAGE:	CALL SCREW
PARAMETERS:	None
COMMON:	None
SUBROUTINES:	None
LANGUAGE:	FORTRAN

ROUTINE: SHELL

AUTHOR: Unknown

DATE: May 1967

PURPOSE: To reorder an array of numbers in ascending order.

USAGE:
```
DIMENSION X(N),K(N)
CALL SHELL(X,K,N)
```

PARAMETERS: I N P U T

N - The length of the arrays X and K

INPUT/OUTPUT

X - The array which is reordered by the routine

O U T P U T

K - An integer array of size N. K(I) contains the original position of new element X(I)

COMMON: None

SUBROUTINES: None

LANGUAGE: FORTRAN

ROUTINE: SHELLX

AUTHOR: Unknown

DATE: May 1967

PURPOSE: To reorder the elements of an array Y according to an index array K such that Y(K(I)) is moved to position Y(I).

USAGE:
```
DIMENSION Y(N),K(N),DUM(N)
CALL SHELLX(Y,K,N,DUM)
```

PARAMETERS:

I N P U T

K - The index array
DUM - An array used for intermediate storage
N - The length of arrays Y, K and DUM

INPUT/OUTPUT

Y - The array which is reordered by the routine

COMMON: None

SUBROUTINES: None

LANGUAGE: FORTRAN

ROUTINE: SMASK

AUTHOR: W. J. Erickson

DATE: October 1974

PURPOSE: First mask, then shift every entry in an array.

USAGE: CALL SMASK (MAT,MATDIM,ISHIFT,MASK)

PARAMETERS: I N P U T

MAT - Array to be modified
MATDIM - Array dimension
ISHIFT - Number of bits to left shift
MASK - Mask to use prior to shifting

O U T P U T

MAT - Array with each word masked and shifted

COMMON: None

SUBROUTINES: None

LANGUAGE: COMPASS

DISCUSSION: Useful for extracting packed data from an array.

ROUTINES: SØERD
SØES01 - SØES05

AUTHOR: G. von Limbach

DATE: August 1974

PURPOSE: SØERD and its subroutines SØES01 through SØES05, a general sort package for element data, sort variable size blocks of data and the associated index table using a correspondence table.

USAGE: COMMØN/SØRTPAR/IPAR(40)
CALL SØERD

COMMON: COMMØN/SØRTPAR/IPAR(40)
IPAR is initialized by the calling program with file names, index names, and SHIFT and INCR arguments for input and output data. For precise current definition of contents of IPAR see comments in SØERD.

SUBROUTINES: ESØRT1, MØVSTG, INCR, SHIFT, FETEDIT

LANGUAGE: SØERD - SNARK
SØES01 - SØES05 - FØRTRAN

DISCUSSION: SØERD clears blank common and adds its own files. Scratch files will be dropped and any files added by SØERD will be deleted. Sorting of index tables is done by ESØRT1 and blocks of data are moved by MØVSTG. The input index table contains packed block numbers, row numbers and data size numbers in input order. The input data blocks may contain data not referenced in the index table. The correspondence table is a list of the output positions, designated in the input order. The output will be in increasing order, but need not be sequential.

ROUTINE: STAR

AUTHOR: F. P. Gray

DATE: June 1972

PURPOSE: This subroutine interrogates a matrix index name and counts the number of asterisks (*) present. A test index and masking index are also formed.

USAGE: CALL STAR (INDEX,ITEST,IMSK,ISTAR)

PARAMETERS: I N P U T

INDEX - 7 character index name, left adjusted, zero fill

O U T P U T

ITEST - 7 character test index, left adjusted, zero fill
IMSK - 7 character test index, left adjusted, zero fill
ISTAR - Count of stars in index

COMMON: None

SUBROUTINE: None

LANGUAGE: FORTRAN

DISCUSSION: The test index is formed by converting the stars (*) in the index name to zeros.
The masking index is formed from the octal word 777777777777700000 by replacing the sevens by zeros in the locations corresponding to the stars (*) in the index name.

ROUTINE: TRANS

AUTHORS: S. Andreassen
S. Wahlstrom

DATE: January 1969

PURPOSE: Establish 3X3 orthogonal coordinate transformation matrix between a global and a local cartesian rectangular coordinate system.

USAGE:
```
DIMENSION T(3,3),DX(3),DYZ(3)
CALL TRANS(T,DX,DYZ,M,S)
```

PARAMETERS:

I N P U T

DX - Global coordinates of local x axis
DYZ - Global coordinates of a vector in local xy or xz plane
M - <0 if DYZ in xz plane
=0 if yz orientation is arbitrary
>0 if DYZ in xy plane

O U T P U T

T - Transformation matrix such that V (local) = TV (global)
S - >0 length of DX
-1 DX is shorter than 10^{-15}
-2 DYZ is parallel to DX

COMMON: None

SUBROUTINES: None

LANGUAGE: FORTRAN

DISCUSSION: First, the x-axis is obtained by normalization of DX; then, the normalized cross product of DX and DYZ defines the second axis (either y or z depending on M). Finally the third axis is obtained as a cross product of the first two. If M=0, the last two rows of T are set to zero.

ROUTINE: VECPRØD

AUTHOR: S. Wahlstrom

DATE: April 1972

PURPOSE: Compute the vector (cross) product

USAGE:
```
DIMENSIØN A(3),B(3),C(3)
CALL VECPRØD(A,B,C,D)
```

PARAMETERS:

I N P U T

B - First factor vector
C - Second factor vector

O U T P U T

A - Result vector - A=(B)X(C)
D - Length of A

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: The equations used are:

$$A(1) = B(2)C(3) - B(3)C(2)$$
$$A(2) = B(3)C(1) - B(1)C(3)$$
$$A(3) = B(1)C(2) - B(2)C(1)$$
$$D = SQRT(A(1)^2 + A(2)^2 + A(3)^2)$$

ROUTINE: VTMEI

AUTHORS: H. Schmeising
C. Berner

DATE: August 1969

PURPOSE: Compute the product vector of either:

1. A matrix stored by columns, post multiplied by a column vector, or

2. A matrix stored by rows, pre-multiplied by a row vector

USAGE:
```
DIMENSION  V(L),R(M)
DIMENSION  A(M,K)  where  K ≥ L  for 1.
DIMENSION  A(K,M)  where  K ≥ L  for 2.
CALL  VTMEI(V,A,N,L,M,R)
```

PARAMETERS: I N P U T

V - Factor vector
A - Factor matrix
N - Column to column skip factor for 1.
Row to row skip factor for 2.
L - Dimension of vector
M - Column dimension of matrix for 1.
Row dimension of matrix for 2.

O U T P U T

R - Result vector

COMMON: None

SUBROUTINES: None

LANGUAGE: COMPASS

DISCUSSION: Time = 40+4.6(M/2-1)+1.8((M-L)/2-1) microseconds

ROUTINE: VTMVI

AUTHORS: H. Schmeising
C. Berner

DATE: August 1969

PURPOSE: Compute the product vector of either:

1. A matrix stored by rows, post multiplied by a column vector, or

2. A matrix stored by columns, pre-multiplied by a row vector

USAGE:
```
DIMENSION  V(L),R(M)
DIMENSION  A(M,K)  where  K ≥ L  for 1.
DIMENSION  A(K,M)  where  K ≥ L  for 2.
CALL  VTMVI(V,A,N,L,M,R)
```

PARAMETERS: I N P U T

V - Factor vector
A - Factor matrix
N - Row to row skip factor for 1.
    Column to column skip factor for 2.
L - Dimension of vector
M - Row dimension of matrix for 1.
    Column dimension of matrix for 2.

O U T P U T

R - Result vector

COMMON: None

SUBROUTINES: None

LANGUAGE: COMPASS

DISCUSSION: Time = 40+4.6(M/2-1)+1.8((M-L)/2-1) microseconds

ROUTINE: VTMVIA

AUTHOR: H. Hansteen

DATE: August 1969

PURPOSE: Compute the product vector of either:

1. A matrix stored by rows, post multiplied by a column vector, or

2. A matrix stored by columns, pre-multiplied by a row vector

and add it to the existing contents of the result vector.

USAGE:
```
DIMENSIØN  V(L),R(M)
DIMENSIØN  A(M,K)  where  K ≥ L  for 1.
DIMENSIØN  A(K,M)  where  K ≥ L  for 2.
CALL  VTMVI(V,A,N,L,M,R)
```

PARAMETERS: I N P U T

V - Factor vector
A - Factor matrix
N - Row to row skip factor for 1.
    Column to column skip factor for 2.
L - Dimension of vector
M - Row dimension of matrix for 1.
    Column dimension of matrix for 2.

O U T P U T

R - Result vector

COMMON: None

SUBROUTINES: None

LANGUAGE: CØMPASS

DISCUSSION: This routine is identical to VTMVI, except for its accumulation feature.

ROUTINE: WGRAD

AUTHOR: C. Mounier

DATE: October 1975

PURPOSE: To write data records on file GRAFSQ2 (second file in common block /FILE99/)

USAGE: CALL WGRAD(KØD,X,Y,Z,IBFR,LKØDE,XL)

PARAMETERS: I N P U T

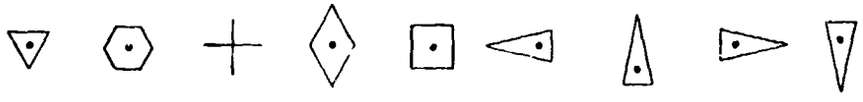
KØD - Input code for data record
= 1 directs data in current call to be written to GRAFSQ2 with more data to follow.
= 3 indicates that this is the last call to the routine for the current display.

x - Array of X coordinates to be plotted.
y - Array of Y coordinates to be plotted
z - Array of Z coordinates to be plotted.

IBFR(1) - Curve plotting code (other than contour plots)

= 1 solid lines
= 2 points
= 3 dashed lines (dash length = 0.1 inch)
= 4 solid lines and symbol at points symbol denoted by value of LBFR(2)
= 5 dashed lines and symbol at points
= 6 symbols (at points)
= 8 dashed lines - length specified by the user in IBFR(4).

IBFR(2) - Plot symbol code
= 1HA thru 1HZ and 1H1 thru 1H9 results in the specified character being plotted as the symbol
= 1 thru 9 results in the following 9 symbols being plotted, respectively



IBFR(3) - Plot symbol code for user-specified boundary points on a contour plot only
= 1 thru 9 results in the above 9 symbols being plotted, respectively.

IBFR(4) - User specified dash length in 0.01 inch units (IBFR(k)=10 implies 0.1 inch dash length)

IBFR(5) - Skip factor for points in the X,Y,Z arrays.
= 1 implies all points
= n implies point NOS. 1, 1+n, 1+2n, ...

IBFR(7) - Number of trace types (NT) per record
NT ≤ 10

IBFR(8) - Number of data points defining first trace type (≤ 200)

IBFR(7+NT) - Number of data points defining NT-th trace type.

IBFR(8+NT) - Number of curves to plot for first trace type, each consisting of IBFR(8) points.

IBFR(7+2*NT) - Number of curves to plot for NT-th trace type, each consisting of IBFR(7+NT) points.

LKØDE - plot labelling code
= 0 labelling not required
= 1 plot user-defined plot labels (see XL)
= 2 plot program-generated numeric labels. (See XL)

XL - The following data required if LKODE = 1
XL(1) = Number of labels N (≤ 100)
XL(2) thru XL(4) - X,Y,Z positions of the first label.
XL(3N-1) thru XL(3N+1) - X,Y,Z positions of the N-th label.
XL(3N+2) = First label (Holerith ≤ 10 characters)
XL(4N+1) = N-th label (Holerith ≤ 10 characters)

The following data required if LKODL = 2.

XL(1) thru XL(3) - X, Y, Z offsets for labels in user units.

XL(4) - first label
XL(5) - increment for successive labels
XL(6) - first point to be labelled for a given curve.
XL(7) - skip factor for points on curves
XL(8) - first curve to be labelled
XL(9) - skip factor for curves
XL(10) - number of points to be labelled per curve.

COMMON: CØMMØN/FILE99/ - ATLAS system common block

SUBROUTINES: LBLBUF,PCKTRC,PCKXYZ,SPTS

LANGUAGE: FØRTRAN

DISCUSSION: Subroutines WGRAH, WGRAD and WGRAT together formulate interfacing software for the ATLAS graphics capability. Using these routines the user may define plots that may be displayed interactively or may be plotted offline. All features of the ATLAS interactive graphics capability are applicable to these user-defined plots. The mode of usage is as follows:

File Buffers

```
DIMENSIØN BUF1(600),BUF2(600),BUF3(117)
Arrays for WGRAH routine (Maximum dimensions)
DIMENSIØN IHD(29),XHD(22),CL(99)
Dimension of CL=number of contour levels specified.
CL may be a simple variable for other types of
plots
DIMENSIØN X(1000),Y(1000),Z(1000),IBFR(27),XL(401)
Dimensions of x,y,z = the maximum number
of points to be displayed per call to WGRAD.
Dimension of XL = (4*N+1) where N is the number
of labels to be displayed with LKODE=1
Dimension of XL = 10. IF LKODE = 2,
XL may be a simple variable if no labels are to be
displayed.
DIMENSIØN ITXT(90),NCH(10),CS(10),PØS(10,3),
NWS(10),ANG(10)
Dimension of ITXT = number of characters to be
displayed/2.
Dimension of NCH,CS, NWS and ANG = Number of set
points (NSP)
Dimension of PØS = (NSP,2) for 2-D data
                 = (NSP,3) for 3-D data
Open files
CALL FILLADD(GRAFSQ1,BUF1,600,1,0,IRP)
```

```
CALL FILEADD(GRAFSQ2,BUF2,600,1,0,IRR)
CLL FILEADD(DIAGFIL,BUF3,117,1,0,IRR)
REWIND GRAFSQ1
REWIND GRAFSQ2
User code to generate arrays for routines
WGRAH,WGRAD and WGRAT
CALL WGRAH(IHD,XHD,CL)
CALL WGRAD,(KØD,X,Y,Z,IBFR,LKØDE,XL)
CALL WGRAT(KØD,ITXT,NSP,NCH,CS,PØS,NWS,ANG)
Return files
CALL FETDEL(GRAFSQ1,IRR)
CALL FETDEL(GRAFSQ2,IRR)
CALL FETDEL(DIAGFIL,IRR)
Execute GRAPHICS module, using only the ØFFLINE
parameter. This parameter is not required if
interactive displays only are required
EXECUTE GRAPHICS (ØFFLINE =     )
```

ROUTINE: WGRAH

AUTHOR: C. Mounier

DATE: October 1975

PURPOSE: To write a header record on the GRAFSQ1 (first file in common block /FILE99/)

USAGE: CALL WGRAH(IHD,XHD,CL)

PARAMETERS: I N P U T

Array IHD

IHD(1) - Holerith blank-filled plot-group identifier

IHD(2) thru IHD(4) - Holerith blank-filled plot identifier.

IHD(5) - Number of title words for titling the axes for x-y graphs or for titling the display for orthographic, and contour plots. (≤ 4)

IHD(6) - Graph format code:
= 0, do not display grid
= 1, draw axes with tick marks.
= 2, draw full grid.

IHD(7) - Number of grid divisions along X axis. If set to zero, the program will automatically generate suitable divisions.

IHD(8) - Similar to IHD(7), for Y axis.

IHD(9) - Header code:
= 0, one header record for a display
= 1, multiple header records for a display. This feature is used for obtaining multiple plots on a single frame. Applicable only for graphs and contour plots.

IHD(10) - Plot type code
= 1, X-Y graph
= 2, orthographic projection
= 3, contour using rectangular boundary
= 4, perspective
= 5, contour using user-specified boundary

IHD(11) - Grid code
= 0, linear X and Y
= 1, logarithmic X, linear Y
= 2, linear X, logarithmic Y
= 3, logarithmic X, logarithmic Y
= 4, polar grid with radial axis
= 5, polar grid with logarithmic radial axis.

IHD(12) - Text option code
= 0, subroutine WGRAT will not be called
= 1, subroutine WGRAT will be called.

IHD(13) - Number of divisions along each side of the rectangular boundary for a contour plot.

IHD(14) - Number of contour levels (≤ 99)

IHD(15) thru IHD(14+3*IHD(5))
X,Y plot titles or orthographic and contour display titles.

IHD(15+4*IHD(5))
Order of rotation about X axis.

IHD(16+4*IHD(5))
Order of rotation about Y axis.

IHD(17+4*IHD(5))
Order of rotation about Z axis.
Each of the above 3 variables may be assigned one of the numbers 1, 2 or 3. and are meaningful only for orthographic projections.

Array XHD

XHD(1) - Horizontal plot size for offline plots

XHD(2) - Vertical plot size for offline plots

XHD(3) - Number of boundary points specified if user-specified boundary option is used for contour plots.

XHD(4) - Rotation angle about X axis

XHD(5) - Rotation angle about Y axis

XHD(6) - Rotation angle about Z axis

XHD(4) thru XHD(6) are meaningful only for orthographic projections.

XHD(7) - X coordinate of eyepoint

XHD(8) - Y coordinate of eyepoint

XHD(9) - Z coordinate of eyepoint

XHD(10) - X coordinate of center of interest

XHD(11) - Y coordinate of center of interest

XHD(12) - Z coordinate of center of interest

XHD(7) thru XHD(12) are meaningful only for perspective plots

XHD(13) - Code for displaying the user axis orientation and scales for orthographic plots
= 0, display orientation and scales
≠ 0, do not display the above
XHD(14) - not used
XHD(15) - Code for size of labelling characters
= 0, size = 0.1067" - default
= 1., size = .25"
= 2., size = .125"
= 3., size = .1067"
= 4., size = .0625"
= floating point number indicating size in inches.
CL(1) thru CL(IHD(14)) -
Contour level values CL can be a single dummy variable if the plot type is not contour.
XHD(16) - Scale value for scaling orthographic projections
XHD(17) - Maximum X value for subject space
XHD(18) - Maximum Y value for subject space
XHD(19) - Maximum Z value for subject space
XHD(20) - Minimum X value for subject space
XHD(21) - Minimum Y value for subject space
XMD(22) - Minimum Z value for subject space
CL(4) thru CL(IHD(14))
Contour level values CL may be a dummy variable if IHD(14) = 0 i.e. the plot is not a contour plot.

COMMON: COMMON/FILE99/ - ATLAS system common block

SUBROUTINES: None

LANGUAGE: FORTRAN

DISCUSSION: See WGRAD

ROUTINE: WGRAT

AUTHOR: C. Mounier

DATE: October 1975

PURPOSE: To write special annotation data records.

USAGE: CALL WGRAT(KØD,ITXT,NSP,NCH,CS,PØS,NWS,ANG)

PARAMETERS: I N P U T

KØD - Input code for data record
= ± 1, directs data in current call to be written to GRAFSQ2 with more data to follow
= ± 3, indicates that this is the last call to the routine for the current display. For the above options + indicates Tektronix - drawn characters, - indicates vector - drawn character.

ITXT - Array of annotation information (≤ 96 words)

NSP - Number of set points (locations to commence a string of characters)

NCH(4) - Number of characters to be displayed at first set point.

NCH(NSP) - Same for NSP-th set point

CS(1) - Character size in inches for first set point (required only if KØD < 0)

CS(NSP) - Same for NSP-th set point

POS(1,1) thru POS(1,3) -
Subject space coordinates for first set point. (X , Y , Z)
POS(NSP,1) thru POS(NSP,3) -
Same for NSP-th set point

Only X and Y coordinates required for 2 dimensional displays.

NCT(1) - Number of characters to be displayed at first set point.
NCT(NSP) - Same for NSP-th set point

The characters are stored contiguously in the array ITXT.

ANG(1) - Orientation angle in degrees, measured counterclockwise from horizontal direction for the first set point (required only if KOD < 0)
ANG(NSP) - Same for NSP-th set point.

COMMON: COMMON/FILE99/ - ATLAS system common block

SUBROUTINES: PCKTXT

LANGUAGE: FORTRAN

DISCUSSION: See WGRAD

ROUTINE: ZERØCØL

AUTHOR: M. C. Redman

DATE: January 1973

PURPOSE: To initialize specified rows and columns to zero.

USAGE:
```
DIMENSIØN  Z(NRØWZ,NL),DZ1(NRØWZ,NL),DZ2(NRØWZ,NL)
CALL ZERØCØL (M,NF,NL,Z,NRØWZ,INDD,DZ1,DZ2)
```

PARAMETERS:

I N P U T

M - Number of rows to zero out
NF - Starting column to zero out
NL - Last column to zero out
NRØWZ - Row dimension size of Z,DZ1 and DZ2
INDD - Indicator for zeroing out DZ1 and DZ2
=0, Zero out only Z
=1, Zero out only Z and DZ1
=2, Zero out only Z and DZ2
=3, Zero out only Z,DZ1 and DZ2

O U T P U T

Z,DZ1,DZ2 Matrices with submatrices zeroed out

COMMON: None

SUBROUTINE: None

LANGUAGE: FØRTRAN

DISCUSSION: The columns NF to NL for the first M rows are initialized to zero for matrices indicated by INDD indicator.

ROUTINE: ZERØUT

AUTHOR: O. L. Anderson

DATE: Unknown

PURPOSE: Set the contents of a block of core to zero.

USAGE: CALL ZERØUT(FIRST,LAST)

PARAMETERS: I N P U T

FIRST - Address of first word of block
LAST - Address of last word of block

COMMON: None

SUBROUTINES: None

LANGUAGE: CØMPASS

DISCUSSION: This routine is the same as WQZERØ, part of the SNARK package.

1002. ATLAS CLIB LIBRARY ROUTINES

ROUTINE: CUTNE

AUTHOR: Stig Wahlstrom

DATE: July 1975

PURPOSE: Generate a stiffness data set from a subset of an existing set.

USAGE: CALL CUTNE(SET,NEWSET NØDSET,DATAFIL, KSFSIZE,IRR)

PARAMETERS: I N P U T

SET - Existing set number

NEWSET - New set number

NØDSET - Nodal subset number

ELEMSET - Element subset number

DATAFIL - File name (use DATAKNF)

KSFSIZE - KSF block size in NEWSET

O U T P U T

COMMON: IRR - LE 0 successful
1 SET not available
2 NEWSET exists already
3 NØDSET not available
4 ELEMSET not available
5 NØDSET and ELEMSET not compatible

SUBROUTINES: CUT1,CUT2,CUT3,CUT4,CUT5,INCR

LANGUAGE: SNARK

ROUTINE: DESCØNF

AUTHOR: F. D. Nelson

DATE: April 1976

PURPOSE: Scan the minimum/maximum margins of safety array and check for convergence on element allowables.

USAGE: SNARK CALL DESCØNF(DATAMIN,=NØLC,=IØPT1)

PARAMETERS:

I N P U T

DATAMIN - Minimum & maximum margins of safety matrix position

NØLC - Number of loadcases in current stage

BEGCYCL - First design cycle in a do loop range

ENDCYCL - Last design cycle in a do loop range

CURCYCL - Current design cycle in a do loop series

CØNVERG - Array containing the users convergence constraint values. Cells 1 thru 13 are for element types 1 thru 13.

ISET - Data set number
ISTAGE - BC stage number
IPØS - Matrix position offset

INPUT/OUTPUT

IØPT1 - Element convergence flag. Initialized to 1 by calling routine. Set to zero by DESCØNF if element convergence is successful.

COMMON: CØMMØN/USERCØM/BEFCYCL,ENDCYCL,CURCYCL,
CØNVERG(17), ISET, ISTAGE, IPØS
INTEGER DATAMIN

SUBROUTINES: None

LANGUAGE: FØRTRAN

ROUTINE: DESCØNS

AUTHOR: F. D. Nelson

DATE: April 1976

PURPOSE: Check for user specified convergence options. Any specified option(s) will be tested for completion.

USAGE: CALL DESCØNS

COMMON: CØMMØN/USERCØM/BEGCYCL,ENDCYCL,CURCYCL, CØNVERG(17), ISET, ISTAGE, IPØS

PARAMETERS: I N P U T

BEGCYCL - First design cycle in a do loop range

CØNVERG - Array containing the user convergence constraint options. Cells 1 thru 13 are for element types 1 thru 13. Cell 14 is the total weight change between consecutive cycles. Cell 15 is for ratios of weight change to total weight. Cell 16 is convergence summary print flag (=0 skip print; ≠0 - print summary). Cell 17 is the DESCØNS execution flag (=0 - test cells 1 thru 15; <0 - convergence terminated; =1 - convergence not required, cells 1 thru 15 are zero).

ISET - Data set number

ISTAGE - BC stage number

IPØS - Matrix position offset. Local position is set to IPØS + 1.

INPUT/OUTPUT

ENDCYCL - Last design cycle in a do loop range. Set to zero by DESCØNS if error detected.

CURCYCL - Current design cycle in a do loop series. Set to zero by DESCØNS if error detected. Set to ENDCYCL if convergence constraints satisfied.

SUBROUTINES: DESCØNF, FILEADD, IFRREC, SHIFT

LANGUAGE: SNARK, FØRTRAN

DISCUSSION: DESCØNS provides the user with the following convergence options:

1. Maximum allowable changes for each selected element type
2. Maximum change in total weight for two consecutive cycles
3. Maximum ratio of change in weight to total weight

Convergence constraint values must be stored by the user's control program in common block USERCØM, array CØNVERG. DESCØNS should be called for initilization after READ INPUT and at the end of each design execution cycle. Each call to DESCØNS must be coupled with a preceeding EXECUTE MASS (PRØCEDURE = n) statement.

ROUTINE: EXPFLEX

AUTHOR: F. P. Gray

DATE: June 1974

PURPOSE: This subroutine expands a symmetric matrix to a full N*N matrix.

USAGE: CALL EXPFLEX (NTMP,NR,NPØS,N)

PARAMETERS:

I N P U T

NTMP(NR) - A user matrix partition stored in lower triangular format.

NR - Row dimension of NTMP

N - Row dimension of NPØS

O U T P U T

NPØS(N,N) - Expanded matrix

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

ROUTINE: FLEXFIL

AUTHOR: F. P. Gray

DATE: May 1973

PURPOSE: This routine sets up the flexibility matrix data file of the ATLAS-FLEXSTAB interface.

USAGE: CALL FLEXFIL(INDEX)

PARAMETERS: I N P U T

INDEX - Flexibility matrix index name (H or L format)
FXXX000 - Flexibility matrix

O U T P U T

The flexibility matrix file (unblocked)
The flex matrix is written onto file SAVESSF as N records - one row per record.

COMMON: /KERRØR/
/KQBUFF
/KQRNDM/
/MATTAB/

SUBROUTINES: EXPFLEX, GMLKUP, INCRB

LANGUAGE: FØRTRAN, SNARK

FILE USAGE: MULTRNF
SAVESSF

DISCUSSION: WARNING - This subroutine accesses data in blank common by direct calls to SNARK support routines. Modifications may be required for new versions of SNARK.

SNARK subroutines called - LQ11

SNARK arrays used - KQDATA

ROUTINE: GCSGEØM

AUTHOR: K. K. Yagi

DATE: July 1976

PURPOSE: To convert the GCS surface external storage into matrices on DATARNF for the GCS-ATLAS interface.

USAGE: CALL GCSGEØM(LFIL,ID)

PARAMETERS: I N P U T

LFIL - File on which the GCS surface external storage is stored.
ID - id name of the geometry component to be defined.

COMMON: None

SUBROUTINES: FILEADD,GCSRD,IFRREC,INCR

LANGUAGE: FØRTRAN

FILE USAGE: DATARNF

DISCUSSION: This routine is entered using the SNARK language for matrix manipulation.

GCSGEØM checks DATARNF to see if this is the first geometry component to be defined or if there are other geometry definitions. It then creates or modifies matrices accordingly.

ROUTINE: KSATAT

AUTHOR: R. L. Dreisbach

DATE: February 1976

PURPOSE: Interface SAMECS-generated (see reference below) reduced stiffness matrices to ATLAS for subsequent ATLAS analyses. The CØEFIL 1 matrix data from a SØLPAC format file are read from an ATLAS restart file and output on file MULTRNF of ATLAS as a User Matrix.

USAGE: CALL KSATAT(NIF,NSTR,IRR)

PARAMETERS: I N P U T

NIF - Input file positioned properly at SØLPAC matrix. Must be one of the ATLAS restart files (e.g. SAVESSF).

NSTR - Substructure number, if the output matrix is to be used as such. Otherwise, a number in the range 0 thru 999 to be used to identify the output matrix KREDXXX.

O U T P U T

IRR - Error indicator.
0 = No errors
1 = Error encountered

KREDXXX - Stiffness matrix on MULTRNF file. XXX is input as NSTR.

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: The output matrix on the ATLAS file MULTRNF is identified as a SET/STAGE reduced stiffness matrix or as a lowest-level substructure reduced stiffness matrix. This matrix can be used in subsequent ATLAS analyses provided that the following ATLAS input data are also provided.

1. Nodal data containing at least those nodes associated with the stiffness matrix.

2. Stiffness elements providing dummy stiffnesses for at least the retained kinematic freedoms of the matrix. SCALAR elements are most convenient to effect these dummy stiffnesses.

3. Boundary condition data which identify the retained freedoms in the order associated with the matrix.

4. Other ATLAS data such as Loads, Mass, etc. to be associated with the retained freedoms.

Reference: S. L. Barter, et al., "SAMECS Structural Analysis System--User's Document, Version S172E," Boeing Computer Services, Doc. No. BCS-G0396, Aug., 1974.

ROUTINE: LRCGEØM

AUTHOR: G. N. Bates

DATE: December 1975

PURPOSE: To convert data intended for the NASA-LRC Airplane Configuration Program to ATLAS Geometry data.

USAGE: CALL LRCGEØM(IREAD,IWRITE,IERR)

PARAMETERS: I N P U T

IREAD - User defined file which contains the NASA-LRC data

IWRITE - User defined file which will receive the ATLAS Geometry data

IERR - Error counter
=0, No errors
≠0, Error contained on file IWRITE

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: NASA-LRC data is read from file IREAD, converted to ATLAS Geometry data and written to file IWRITE for further use by ATLAS.

ROUTINE: MASSFIL

AUTHOR: F. P. Gray

DATE: May 1973

PURPOSE: This routine sets up the mass/geometry data file of the ATLAS-FLEXSTAB interface.

USAGE: CALL MASSFIL(INDEX,NSET,NSTAGE)

PARAMETERS: I N P U T

INDEX - Mass matrix index name (H or L format)
NSET - Data set number
NSTAGE - Stage number
KNØALTX - Nodal data matrix
KNC100X - Nodal correspondence table
KRFV0XX - Retained freedom vector
MDCXXXX - Mass matrix

Card Input: Reference 1-1

O U T P U T

The FLEXSTAB mass/geometry file (unblocked)

SLENDER BØDY RECØRD

WORD		ITEM	FORMAT
1		YØ	Floating
2		ZØ	"
3		SYMM	Integer
4		ASYMM	"
5		CØDE	"
6		NN	"
7	XN1		Floating
8	XN2		"
"	"	node coordinate	"
"	"	(NN words)	"
"	"		"
7+NN	MN1		"
8+NN	MN2		"
"	"	mass at node	"
"	"	(NN words)	"
"	"		"
7+2*NN		NDØF	Integer
8+2*NN		X	1HX or blank
9+2*NN		Y	1HY or blank
10+2*NN		Z	1HZ or blank

THIN BØDY RECØRD

WORD		ITEM	FORMAT
1		XØ	floating
2		YØ	"
3		ZØ	"
4		T	"
5		SYMM	INTEGER
6		ASYMM	"
7		CØDE	"
8		NN	"
9		NP	"
10	XN1		floating
11	YN1		"
12	XN2		"
13	YN2	node coordinates	"
"	"	(2*NN words)	"
"	"		"
"	"		"
10+2*NN	MN1		"
11+2*NN	MN2		"
"	"	mass at node	"
"	"	(NN words)	"
"	"		"
10+3*NN	N1		Integer
"	B1	nodes and assoc	"
"	N2	body for each	"
"	B2	panel corner	"
"	"	point	"
"	"	(8*NP words)	"
10+3*NN+8*NP	"		"
11+3*NN+8*NP		NDØF	Integer
12+3*NN+8*NP		X	1HX or blank
13+3*NN+8*NP		Y	1HY or blank
14+3*NN+8*NP		Z	1HZ or blank

An end-of-file mark concludes the mass/geometry data.

COMMON: /KERRØR/
/CØNPARS/
/KQBUFP/
/KQRNDM/

SUBROUTINES: SETUP, LØDAREC, INCRB

LANGUAGE: FØRTRAN, SNARK

FILE USAGE: DATARNF
MASSRNF
SAVESSF

WARNING - This subroutine accesses data in blank common by direct calls to SNARK support routines. Modifications may be required for new versions of SNARK.

SNARK subroutines called - LQ11

SNARK arrays used - VQDATA
KQDATA

ROUTINE: SETUP

AUTHOR: F. P. Gray

DATE: May 1973

PURPOSE: This routine sets up the geometric data for the FLEXSTAB mass/geometry file.

USAGE: CALL SETUP(KNØD,KNCT,MASS,NØUT,KR,MR,MRØW,NNØDE, ID,NRFV)

PARAMETERS:

I N P U T

KNØD(X,X)	-	Nodal data matrix
KNCT(X)	-	Nodal correspondence table
MASS(X,X)	-	Mass matrix
KR	-	Row dimension of KNØD
MR	-	Row dimension of MASS
MRØW	-	Pointer to mass matrix terms for the body
NNØDE	-	Number of retained nodes for the body
ID	-	Body type indicator
NRFV	-	Retained freedom vector

O U T P U T

NØUT(X)	-	Geometric data

COMMON /KERRØR/

SUBROUTINES: NCUIR

LANGUAGE: FØRTRAN

ROUTINE: STREFIL

AUTHOR: F. P. Gray

DATE: June 1973

PURPOSE: This routine reads a FLEXSTAB net loads file and sets up the ATLAS nodal loads matrices.

USAGE: CALL STREFIL (NCASES)

PARAMETERS: I N P U T

The FLEXSTAB net loads file (unblocked)

NCASES - The number of load cases (integer)

NOTE: The number of load cases represents the number of input files, the number of symmetric cases, or the number of antisymmetric cases, whichever is largest.

O U T P U T

QSYMMXX - Symmetric nodal loads matrix
QASYMXX - Antisymmetric nodal loads matrix

NOTE: The load case numbers will be assigned sequentially beginning with the first input file.

COMMON: /KERRØR/
/KQBUFP/
/KQRNDM/

SUBROUTINES: None

LANGUAGE: FØRTRAN, SNARK

FILE USAGE: DATARNF
SAVESSF

DISCUSSION: WARNING - This subroutine accesses data in blank common by direct calls to SNARK support routines. Modifications may be required for new versions of SNARK.

SNARK subroutines called - LQ11

SNARK arrays used - VQDATA
KQDATA

ROUTINE: VAMAT

AUTHOR: K. R. Helmut
A. Tornallyay
F. P. Gray

DATE: December 1976

PURPOSE: Calculate shear, moment, and torsion along a wing, body, and horizontal tail from a given set of panel airloads and panel weight (symmetric conditions only)

USAGE: Reference 1-1

PARAMETERS: I N P U T

The SDSS tape from FLEXSTAB - file SA F
(Ref. D6-41064-3 FLEXSTAB 1.02.00 Pr<
Description)

The panel weight matrix MDCXXXX

Card Input: Reference Boeing document D6-42315TN

O U T P U T

File SAVESS1 - Unformated Binary (1file per case)

Record	Contents
1	x coordinates of VAMAT cuts
2	y coordinates of VAMAT cuts
3	Shear at cut K
4	Net shear at cut K
5	Bending moment at cut K
6	Net bending moment at cut K
7	Torsion at cut K
8	Net torsion at cut K
9	Panel airload
10	Panel areas

COMMON: /KERRØR/
/KQBUFP/
/KQRNDM/
/CARDS/
/MISC/IMAX,IVAM,NPRINT,LBCUT,NSCUT,NPS1,NPS2,NCS1,
NCS2,KØND

SUBROUTINES: VAMGEØ
VAMIT
VAMØUT

LANGUAGE: FØRTRAN, SNARK

FILE USAGE: DATARNF
MASSRNF
SAVESSF
SAVESS1
SC00SSF

DISCUSSION: This routine represents the VAMAT program and is used in conjunction with the FLEXSTAB/ATLAS interface.

1003. NASTRAN TO ATLAS DATA CONVERSION ROUTINES

ROUTINE: BC

AUTHOR: Kandace K. Yagi

DATE: February 1975

PURPOSE: To convert NASTRAN data (SPC,SPC1,ASET, and ASET1 cards) into ATLAS BC data.

USAGE: CALL BC(NEND)

PARAMETERS: I N P U T

None

O U T P U T

NEND - Equals 1 if an EØF is reached, otherwise it equals zero.

COMMON: CØMMØN/NA/CARD(10),NCARD,NTAPE,LTAPE,NATAPE

Common block to communicate with the calling program.

CARD - Array which holds the current NASTRAN card image. (Hollerith)

NCARD - Number of cards read in to date.

NTAPE - File where the ATLAS input goes.

LTAPE - File where the ATLAS output goes.

NATAPE - File containing an echo of the NASTRAN card images.

CØMMØN/PSCØNST/PS(2048),IP

Common block to receive from NØDSTIF the non-zero constraints

PS - Vector containing nodes and their constraints.

IP - Number of words in PS filled with data.

SUBROUTINES: INTNUM,REALNUM,SHIFT,ISCAN,PAC,IRECØG

LANGUAGE: FØRTRAN

DISCUSSION:

1. BC reads in cards from NTAPE one at a time. It assumes the first card image has already been read into the array CARD before the subroutine was called. All card images read in are immediately echoed out to the NASTRAN output file.

2. BC writes out the ATLAS card image onto LTAPE as soon as the NASTRAN code is translated.

3. BC returns when an EØF is reached or when a card other than SPC,SPC1,ASET, OR ASET1 is encountered.

4. A warning message is issued when an EØF is reached and a continuation card was expected. The card is ignored.

5. The maximum number of stages allowed in the ATLAS BC data is 10. When more than this is encountered in the conversion, an error message is issued and only those SPC or SPC1 cards with SID's already read in will be converted.

6. At the end of the BC conversion a table is printed on NATAPE showing the correspondence between the ATLAS stage numbers and NASTRAN set ID's.

ROUTINE: CMASS

AUTHOR: Kandace K. Yagi

DATE: February 1975

PURPOSE: To convert NASTRAN data (CØNM2 cards) into ATLAS concentrated mass data.

USAGE: CALL CMASS(NEND)

PARAMETERS: I N P U T

None

O U T P U T

NEND - Equals 1 if EØF is reached, it otherwise equals zero.

COMMON: CØMMØN/NA/CARD(10),NCARD,NTAPE,LTAPE,NATAPE

Common block to communicate with the calling program.

CARD - Array which holds the current NASTRAN card image. (Hollerith)

NCARD - Number of cards read in to date.

NTAPE - File where the ATLAS input goes.

LTAPE - File where the ATLAS output goes.

NATAPE - File containing an echo of the NASTRAN card images.

SUBROUTINES: INTNUM,REALNUM,SHIFT,IRECØG

LANGUAGE: FØRTRAN

DISCUSSION:

1. CMASS reads in cards from the input file one at a time. It assumes the first card image has already been read in before the subroutine was called. All card images read in are immediately echoed out onto the NASTRAN output file.

2. It writes out the ATLAS card image onto the ATLAS output file as soon as the NASTRAN card is translated.

3. CMASS returns when an EØF is reached or when a card other than CØNM1 is encountered.

4. An error message is issued when a continuation card was indicated for a NASTRAN CØNM2 card and an EØF was reached instead.

ROUTINE: ELESTIF

AUTHOR: Kandace K. Yagi

DATE: February 1975

PURPOSE: To convert NASTRAN data (BARØR,CBAR,PBAR,CHEXA1, CHEXA2,CØNRØD,CRØD,PRØD,CQDMEN,PQDMEN,CTRMEN, PTRMEN,CQUAD1,PQUAD1,CTRIA1,PTRIA1,CQUAD2, PQUAD2,CQDPLT,PQDPLT,CTRPLT,PTRPLT,CIRBSC, PTRBSC,CSHEAR,PSHEAR,CTUBE,PTUBE cards) into ATLAS element data.

USAGE: CALL ELESTIF(NEND)

PARAMETERS: I N P U T

None

O U T P U T

NEND - Equals 1 if an EØF is reached, otherwise it equals zero.

COMMON: CØMMØN/NA/CARD(10),NCARD,NTAPE,LTAPE,NATAPE

Common block to communicate with the calling program.

CARD - Array which holds the current NASTRAN card image. (Hollerith)

NCARD - Number of cards read in to date.

NTAPE - File where the ATLAS input goes.

LTAPE - File where the ATLAS output goes.

NATAPE - File containing an echo of the NASTRAN card images.

CØMMØN/MC/MCØDE(49,2)

Common block to receive from MATERIAL the material codes and their corresponding temperatures and NASTRAN MID's.

MCØDE - Matrix which holds the information.

SUBROUTINES: INTNUM,REALNUM,SHIFT,ISCAN,IRECØG

LANGUAGE: FØRTRAN

DISCUSSION: 1. ELESTIF is divided into two sections.

The first section reads in the element cards putting the P type cards on file JUNKP and the C type cards on file JUNKC. The CØNRØD, CHEKA1, and CHEKA2 cards are put on JUNCKC. BARØR is not put on a file but decoded as it is read in.

The second section takes one card at a time from JUNKC then looks through JUNKP for a matching card. If none is found, an error message is issued and the next C type card is read in. If a match is found, the program goes to the proper place and decodes the cards, writing the ATLAS card images on the output file.

Transfer from the first section to the second occurs when a card is encountered that is not one of the NASTRAN cards mentioned in the purpose section above, or when an EØF is reached.

2. The routine returns when an EØF is reached on file JUNKC.

3. The first section checks to make sure all cards that indicate a continuation cards do have them. If one is not found, a warning message is issued and that card is ignored.

4. Whenever trouble is encountered in trying to convert an element in the second section, a warning message is issued indicating the trouble.

5. The following shows the correspondence between the NASTRAN cards and the ATLAS cards they convert to:

NASTRAN	ATLAS
CHEXA1, CHEXA2	BRICK
CØNRØD, CRØD and PRØD	RØD or BEAM
CTUBE	BEAM
CBAR and PBAR	

CQDMEN and PQDMEN, CTRMEN and PTRMEN	PLATE
CQUAD1 and PQUAD1, CTRIA1 and PTRIA1, CQUAD2 and PQUAD2, CTRIA2 and PTRIA2, CQDPLT and PQDPLT, CTRPLT and PTRPLT, CTRBSC and PTRBSC	GPLATE
CSHEAR and PSHEAR	SPLATE

6. ELESTIF assumes the first card image has already been read into the array CARD before the subroutine is called.

ROUTINE: INTNUM

AUTHOR: Kandace K. Yagi

DATE: February 1975

PURPOSE: To convert a word containing eight Hollerith characters representing an integer into that integer number.

USAGE: I = INTNUM(X)

PARAMETERS: X - The input Hollerith word to be converted.

I - The output integer form of X.

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION:

1. If X is blank, I is returned with a zero value.
2. The number in X must be stored in contiguous bit locations (no separating blank characters), but can otherwise be located anywhere within X.
3. No warning messages are printed.

ROUTINE: IRECØG

AUTHOR: Kandace K. Yagi

DATE: April 1975

PURPOSE: To recognize the cards that are converted to ATLAS by any of the programs called by NASTATL.

USAGE: CALL IRECØG(I)

PARAMETERS: O U T P U T

I - Equals 1 if the card image in array CARD is recognized. Equals 0 otherwise.

COMMON: CØMMØN/NA/CARD(10),NCARD,NTAPE,LTAPE,NATAPE

Common block to communicate with the calling program.

CARD - Array which holds the current NASTRAN card image. (Hollerith)

NCARD - Number of cards read in to date.

NTAPE - File where the ATLAS input goes.

LTAPE - File where the ATLAS output goes.

NATAPE - File containing an echo of the NASTRAN card images.

ROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: No warning or error messages are issued.

ROUTINE: LØADS

AUTHOR: Kandace K. Yagi

DATE: February 1975

PURPOSE: To convert NASTRAN LØADS cards into ATLAS LØADS data, NASTRAN FØRCE and MØMENT cards into ATLAS NØDAL LØAD data, and NASTRAN PLØAD2 cards into ATLAS ELEMENT LØAD DATA.

USAGE: CALL LØADS (NEND)

PARAMETERS: I N P U T

None

O U T P U T

NEND - Equals 1 if an EØF is reached, otherwise it equals zero.

COMMON: CØMMØN/NA/CARD(10),NCARD,NTAPE,LTAPE,NATAPE

Common block to communicate with the calling program.

CARD - Array which holds the current NASTRAN card image. (Hollerith)

NCARD - Number of cards read in to date.

NTAPE - File where the ATLAS input goes.

LTAPE - File where the ATLAS output goes.

NATAPE - File containing an echo of the NASTRAN card images.

SUBROUTINES: INTNUM,REALNUM,SHIFT,SQRT,IRECØG

LANGUAGE: FØRTRAN

DISCUSSION:

1. LØADS reads in cards from NTAPE one at a time. It assumes the first card image has already been read into array CARD before the subroutine was called. All cards read in are immediately echoed out to the NASTRAN output file.

2. LØADS writes the ATLAS card image onto LTAPE as soon as the NASTRAN code is translated.

3. If all the scale factors on a NASTRAN LØAD card are zero, a warning is issued and that card is ignored.

4. If an EØF is encountered where a continuation card is expected, a warning message is issued, the card is ignored and LØADS returns.

5. LØADS returns when an EØF is reached or when a card is encountered that is converted in another routine.

ROUTINE: MATRIAL

AUTHOR: Kandace K. Yagi

DATE: February 1975

PURPOSE: To convert NASTRAN data (MAT1 and MAT3 cards) into ATLAS material data.

USAGE: CALL MATRIAL (NEND)

PARAMETERS: I N P U T

None

O U T P U T

NEND - Equals 1 if an EØF is reached, otherwise it equals zero.

COMMON: /NA/

Common block to communicate with the calling program.

CARD - Array which holds the current NASTRAN card image. (Hollerith)

NCARD - Number of cards read in to date.

NTAPE - File where the ATLAS input goes.

LTAPE - File where the ATLAS output goes.

NATAPE - File containing an echo of the NASTRAN card images.

COMMON: CØMMØN/MC/MCØDE(49,2)

Common block to pass to ELESTIF the material codes and corresponding temperatures.

MCØDE - Matrix holding the information.

SUBROUTINES: INTNUM,REALNUM,IRECØG

LANGUAGE: FØRTRAN

DISCUSSION:

1. MATRIAL reads in cards from the input file one at a time. It assumes the first card image has already been read into array CARD before the subroutine is called. All card images read in are immediately echoed out to the NASTRAN output file.

2. It writes the ATLAS card image onto the ATLAS output file as soon as the NASTRAN card is translated.

3. MATRIAL returns when an EØF is reached or when a card other than MAT1 or MAT3 is encountered. A warning message is issued when a continuation card is indicated and an EØF is reached.

4. A check is made to make sure MAT3 cards have a continuation card. If not, a warning message is issued and that card is ignored.

5. A check is made on the number of materials defined. If the number exceeds 49, an error message is issued and the rest of the MAT1 and MAT3 cards are ignored.

6. Poisson's ratios in the xy and yz directions are checked for absolute values greater than 1.0, a warning is issued for any such cards and they are ignored.

ROUTINE: NASTATL

AUTHOR: Kandace K. Yagi

DATE: February 1975

PURPOSE: To drive the subroutines which convert NASTRAN data into ATLAS data.

USAGE: CALL NASTATL(LTAPE)

PARAMETERS: LTAPE - file where the ATLAS output goes.

COMMON: COMMON/NA/CARD(10),NCARD,NTAPE,LTAPE,NATAPE

Common block to communicate with the calling program.

CARD	-	Array which holds the current NASTRAN card image. (Hollerith)
NCARD	-	Number of cards read in to date.
NTAPE	-	File where the ATLAS input goes.
LTAPE	-	File where the ATLAS output goes.
NATAPE	-	File where the NASTRAN card images read in are echoed out to.

COMMON/PSCONST/PS(2048), IP

Common block to pass the non-zero constraints from NODSTIF to BC.

PS	-	Vector containing nodes and their constraints.
IP	-	Number of words in PS filled with data.

COMMON/MC/MCODE(49,2)

Common block to pass the material codes and their temperatures from MATRIAL to ELESTIF.

MCODE	-	Matrix which holds the data.

SUBROUTINES: BC,CMASS,ELESTIF,LOADS,MATRIAL,NODSTIF

LANGUAGE: FORTRAN

DISCUSSION:

1. NASTATL reads the first card from the input file into the array CARD.

2. It then interrogates the card image finding which subroutines it should be sent to.

3. Those subroutines return to NASTATL when they come across a card they do not convert. They pass that card image back via the array CARD.

4. NASTATL either finds the proper subroutine to send it to or writes an error message when the card can not be recognized. It then reads in the next card.

5. When an EØF is reached by either NASTATL or one of the subroutines, NASTATL returns.

No warning messages are issued.

ROUTINE: NØDSTIF

AUTHOR: Kandace K. [illegible]

DATE: February 1975

PURPOSE: To convert NASTRAN data (GRID and GRDSET cards) into ATLAS stiffness NØDAL data.

USAGE: CALL NØDSTIF(NEND)

PARAMETERS: I N P U T

None

O U T P U T

NEND - Equals 1 if an EØF is reached, it otherwise equals zero.

COMMON: CØMMØN/NA/CARD(10),NCARD,NTAPE,LTAPE,NATAPE

Common block to communicate with the calling program.

CARD - Array which holds the current NASTRAN card image. (Hollerith)

NCARD - Number of cards read in to date.

NTAPE - File where the ATLAS input goes.

LTAPE - File where the ATLAS output goes.

NATAPE - File containing an echo of the NASTRAN card images.

CØMMØN/PSCØNST/PS(2048),IP

Common block to receive from NØDSTIF the non-zero constraints

PS - Vector containing nodes and their constraints.

IP - Number of words in PS filled with data.

SUBROUTINES: INTNUM,REALNUM,ISCAN,SHIFT,PAC, IRECØG

LANGUAGE: FØRTRAN

DISCUSSION:

1. NØDSTIF reads in cards from the input file one at a time. It assumes the first card image has already been read into CARD before the subroutine was called. All card images read in are immediately echoed out to the NASTRAN output file.

2. It writes the ATLAS card image onto the ATLAS output file as soon as the NASTRAN card is translated.

3. NØDSTIF returns when an EØF is reached or when a card other than GRID or GRDSET is encountered.

   No error or warning messages are issued.

ROUTINE: REALNUM

AUTHOR: Kandace K. Yagi

DATE: February 1975

PURPOSE: To convert a word containing eight Hollerith characters representing a real number into that real number.

USAGE: R = REALNUM(X)

PARAMETERS: X - The input Hollerith word to be converted.

R - The output real form of X.

COMMON: None

SUBROUTINES: None

LANGUAGE: FORTRAN

DISCUSSION:

1. If X is blank, R is returned with a zero value.

2. The number in X must be stored in contiguous bit locations (no separating blank characters), but can otherwise be located anywhere within X.

3. No warning messages are printed.

1004. ATLAS TO NASTRAN DATA CONVERSION ROUTINES

ROUTINE: ATNA/ATEXP

AUTHOR: Kandace K. Yagi

DATE: April 1975

PURPOSE: To convert ATLAS internal data into NASTRAN data or to convert ATLAS internal data into expanded ATLAS data.

USAGE: CALL ATNA(ISET,ISTAGE,NCØND,L11,L21,L31,D31, NELE,NØD,LDBC,MASS,LISTN)

CALL ATEXP(ISET,ISTAGE,BL,L11,L21,L31,D31,NELE, NØD,LDBC,LISTA,LISTEXP)

PARAMETERS: I N P U T (ATNA)

ISET - Set number

ISTAGE - Stage number

NCØND - Mass distribution condition number

L11 - Name of the free loads matrix

L21 - Name of the retain loads matrix

L31 - Name of the support loads matrix

D31 - Name of the support displacement matrix

NELE - 1 if element data are to be converted, otherwise 0

NØD - 1 if node data are to be converted, otherwise 0

LDBC - 1 if loads and BC data are to be converted, otherwise 0

MASS - 1 if mass data are to be converted, otherwise 0

LISTN - Output file where the NASTRAN data goes

I N P U T (ATEXP)

ISET - Set number

	ISTG	-	Stage number
	BL	-	Always set 0
	L11	-	Name of free loads matrix
	L21	-	Name of retain loads matrix
	L31	-	Name of support loads matrix
	D31	-	Name of the support displacement matrix
	NELE	-	1 if element data are to be converted, otherwise 0
	NØD	-	1 if nodal data are to be converted, otherwise 0
	LDBC	-	1 if loads and BC data are to be converted, otherwise 0
	LISTA	-	ATLAS input card images, if none set 0
	LISTEXP	-	Output file where the expanded ATLAS is put.
COMMON:	CØMMØN/KQRNDM/	-	system common block
	CØMMØN/KQBUFP/	-	system common block
	CØMMØN/KERROR/	-	system common block
	CØMMØN/LRCØMM/	-	LØDAREC common block
	CØMMØN/INCREMT/	-	common block of increments used for creating new user id's when more than one NASTRAN card is converted from one ATLAS card.

SUBROUTINES: SHIFT,IFFREC,NØDEXP,NØDCNV,SMASK,ESØRT1, SHELL,INCR,KSFREØR,ELEMEXP,ELEMCNV,BCLDEXP, BCCNV,LDSCNV,MASSCNV,EXPFILE

LANGUAGE: FØRTRAN

DISCUSSION: Subroutine ATNA is used for two purposes. One, to convert ATLAS internal data to NASTRAN. Two, to convert ATLAS internal data into expanded ATLAS data.

Entry point ATNA is used for the ATLAS to NASTRAN conversion and entry point ATEXP is used for expanded ATLAS.

The ATLAS to NASTRAN conversion for DC, LØADS and ELEMENT data takes place in two steps. The first is conversion of ATLAS internal to ATLAS expanded. The second is from ATLAS expanded to NASTRAN.

The ATLAS to NASTRAN conversion for Mass data is directly from ATLAS internal to NASTRAN.

The ATLAS to NASTRAN conversion for Nodal data is a combination of the above two. For the creation of the GRID cards the conversion is done in the two step way and for creation of SEQGP cards the conversion is done directly.

Before calling the subroutine, DATARNF,MASSRNF, MERGRNF, and SC00RNF must be opened. Also SC00SSF,SC01SSF,SC02SSF, and SC03SSF and the output file must be assigned.

Error messages are issued and the subroutine returns when any of the internal ATLAS matrices needed for the conversion are unavailable.

For the expanded ATLAS, if LISTA equals 0 LISTEXP will just hold the data sets created. If LISTA is not zero, LISTEXP will hold the data from LISTA with the expanded ATLAS data sets replacing their corresponding data.

This routine is coded using the SNARK language for matrix manipulation.

ROUTINE: BCCNV

AUTHOR: Kandace R. Yagi

DATE: December 1974

PURPOSE: To convert expanded ATLAS BC data into NASTRAN data (SPC and ASET cards).

USAGE: CALL BCCNV(NTAPE)

PARAMETERS: I N P U T

NTAPE - The file where the NASTRAN cards are written.

COMMON: CØMMØN/CARDS/ - A LØDAREC common block

CØMMØN/LRCØMM/ - A LØDAREC common block

SUBROUTINES: LØDAREC,ØPTFØR

LANGUAGE: FØRTRAN

DISCUSSION:

1. The ATLAS input cards are assumed to be on the input file. If a different file is wanted, NINT in the common block LRCØMM will have to be changed.

2. Only one card image is kept at a time.

3. The ASET card is made from the data of four RETAIN cards. After every fourth RETAIN card is read in, the ASET card is written to NTAPE.

4. The SPC card is written from the data of one SUPPØRT card. The SPC card is written as soon as the SUPPØRT card is deciphered.

5. If the first item on a card is not BEGIN,END,STAGE, SET,RETAIN, or SUPPØRT, the card is not recognized by this program. A warning message is issued and the card is ignored.

6. BCCNV returns when an END card is reached, or when a record is encountered which takes up more than one card. The latter condition is included so that a comment card can be the delimiter between the support of inactive freedoms and the support of active freedoms. The latter are converted with the "support displacement" data and therefore should not be converted here. The SUPPØRT DISPLACEMENT cards are converted by the LDSCNV routine.

ROUTINE: BCLDEXP

AUTHOR: Kandace K. Yagi

DATE: April 1975

PURPOSE: To prepare data for and drive the subroutines which take internal ATLAS data and convert it into expanded ATLAS input data for LØADS and BC.

USAGE: CALL BCLDEXP (ISIT,ISTG,L11,L21,L31,D31,KRFV, NDIM,NN)

PARAMETERS: I N P U T

ISIT	-	Set number
ISTG	-	Stage number
L11	-	Loads matrix for the free partition
L21	-	Loads matrix for the retain partition
L31	-	Loads matrix for the support partition
D31	-	Support displacement matrix
KRFV	-	Flag to indicate if retained freedom vector is available or not.
NAIM	-	Row dimension of KRFV
NN	-	Number of nodes.

COMMON:

CØMMØN/KERRØR/	A system common block
CØMMØN/KQRNDM/	A system common block
CØMMØN/KQBUFP/	A system common block
CØMMØN/LRCØMM/	A LØDAREC common block

SUBROUTINES: SHIFT,RETEXP,IFRREC,SDEXP,GMLKUP,INCRB, LDPRT,DISPRT

LANGUAGE: FØRTRAN

DISCUSSION: BCLDEXP contains SNARK statements.

The expanded ATLAS card image data for LØADS is output on SC03SSF, for BC it is on SC02SSF.

BCLDEXP expects the nodal correspondence table to be read in at position 1 and the assembly control vector to be read in at position 3 before it is called. It also expects the retained freedom vector to be read in at position 2 if it is available.

ROUTINE: CNVMASS

AUTHOR: M. Tamekuni

DATE: March 1975

PURPOSE: To produce NASTRAN card images from ATLAS internal mass data.

USAGE: CALL CNVMASS (NCT,KRFV,MASS,NRØWS,NØUT,NRETHØD,ST, NTYPE,NRET)

PARAMETERS: I N P U T

NCT	-	Nodal correspondence table
KRFV	-	Retained freedom vector
MASS	-	Mass matrix
NRØWS	-	Row dimension of KRFV
NRETNØD	-	Number of retained mass nodes
ST	-	Matrix to store mass matrix for nodes
NØUT	-	Output file
NTYPE	-	Matrix type, diagonal or full
NRET	-	Vector of retained nodes (internal ID's).

COMMON: None

SUBROUTINES: ZERØUT,SHIFT,NCUIR,ØPTFØR

LANGUAGE: FØRTRAN

DISCUSSION: NASTRAN CØNM1 cards are created from the information in the internal ATLAS data.

No error/warning messages are issued.

ROUTINE: DISPRT

AUTHOR: Kandace K. Yagi

DATE: March 1975

PURPOSE: To print expanded BC support cards of active freedoms and the loads support displacement cards.

USAGE: CALL DISPRT(D,NRØW,KACV,KFAV,NCT,NN,LISTLD,LISTBC, NUL)

PARAMETERS: I N P U T

D	-	Displacement matrix.
NRØW	-	Number of rows in D.
KACV	-	Assembly control vector.
KFAV	-	Freedom activity vector.
NCT	-	Nodal correspondence matrix.
NN	-	Number of nodes in set.
NUL	-	0 if D is a null matrice, otherwise 1.
LISTLD	-	File where the SUPPORT DISPLACEMENT card images are written.
LISTBC	-	File where the BC card images are written.

COMMON: CØMMØN/LOADS/NFØRC,NF,LC,LCR

Common block to communicate with the calling program.

NFØRC	-	Array which holds the active freedom numbers.
NF	-	Number of active freedoms.
LC	-	Number of load cases the matrix will hold.
LCR	-	Number of load cases processed to date.

SUBROUTINES: SHIFT,NCUIR

LANGUAGE: FØRTRAN

DISCUSSION:

(1) If D is null, only the support cards are generated. If D is not null, both the support cards and the support displacement cards are generated.

(2) DISPRT looks at both KACV and KFAV to find which nodes and freedoms have displacements. It then prints out the support cards. If D is not null, it also goes through D printing both the zero and non-zero displacements for the support displacement data.

No error or warning messages are issued.

ROUTINE: ELEMCNV

AUTHOR: KANDACE K. Yagi

DATE: December, 1975

PURPOSE: To convert expanded ATLAS element data into NASTRAN data (CØNRØD,PBAR,CBAR,PSHEAR,CSHEAR,PTRIA1,CTRIA1, PQUAD1,CQUAD1,CHEXA1,PTRMEM,CTRMEM,PØDMEM,CQDMEM cards).

USAGE: CALL ELEMCNV (NTAPE)

PARAMETERS: NTAPE - the file where the NASTRAN card images are written.

COMMON: CØMMØN/CARDS/ A LØDAREC common block.

CØMMØN/LRCØMM/ A LØDAREC common block

CØMMØN/INCREMT/ Common block which holds the increments used when one ATLAS card creates two or more NASTRAN cards. The increments are added to the ATLAS user number to create new NASTRAN user numbers.

NL,NU - SPAR and CØVER midnodes (default 10000,20000)

IPL1,IPL2 - PLATE stiffening (default - 10000,20000)

IRØDU,IRØDL - Upper and lower RØDS of SPAR (default - 10000,20000)

ICVU,LCVL - Basic CØVER (default - 10000, 20000).

ICVU1,ICVU2 - Upper CØVER stiffening (default - 12000,14000)

ICVL1,ICVL2 - Lower CØVER stiffening (default - 22000,24000)

SUBROUTINES: LØDAREC,ØPTFØR,SHIFT

LANGUAGE: FØRTRAN

DISCUSSION:

1. The ATLAS input card images are assumed to be on the input file. If a different file is wanted, NINT in the common block LRCØMM has to be changed.

2. Only one ATLAS card image is kept at a time.

3. RØD and SRØD are converted to CØNRØD.

4. SPAR is converted to CØNRØD,CSHEAR and PSHEAR.

5. BEAM is converted to PBAR and CBAR.

6. CØVER and PLATE are converted to CTRMEM and PTRMEM, or CQDMEM and PQDMEM.

7. GPLATE is converted to CTRIA1 and PTRIA1, or CQUAD1 and PQUAD1.

8. BRICK is converted to CHEXA1.

9. SPLATE is converted to CSHEAR and PSHEAR.

10. Off-set PLATES, BRICKs with more than 8 nodes, and SCALAR elements are not converted. If any of these are encountered, a warning message is issued and the element is ignored.

11. A check is made on the lumping factors of SPARS. If they are not zero a warning message is issued and they are assumed zero.

12. Z1A,Z2A,Z3A,Z1B,Z2B, and Z3B, and Z3B in CBAR cards are not set. Comment cards are issued to indicate what they should be.

13. No material property definition cards are created by this program. MAT1 and MAT3 comment cards are issued when an element is converted that needs one.

14. If a card other than BEGIN,END,RØD,BEAM,SPAR, CØVER,PLATE,GPLATE,BRICK,SPØD,SPLATE, or SCALAR is encountered, a warning is issued and the card is ignored.

15. ELEMCNV returns when an END card is reached. There is no check for EØF.

No error messages are issued.

ROUTINE: ELEMEXP

AUTHOR: S. Wahlstrom

DATE: June 1972

PURPOSE: To create expanded ATLAS input cards from the KSF matrices.

USAGE: CALL ELEMEXP(KSF,NCT,FIL,PRØPFW)

PARAMETERS: I N P U T

KSF - Matrix containing element data.

NCT - Nodal correspondence table.

FIL - File where the expanded ATLAS data goes.

PRØPFW - Field width used for section properties.

O U T P U T

None

SUBROUTINES: SHIFT,NCUIR,PUNØNEL

LANGUAGE: FØRTRAN

DISCUSSION: ELEMEXP does the setting up of the information for the card images. It pases this information to PUNØNEL which does the actual writing of the card.

No error or warning messages are issued.

ROUTINE: EXPFILE

AUTHOR: Randace K. Yagi

DATE: April 1975

PURPOSE: To create an ATLAS input deck from a given ATLAS deck and one to four replacement input data sets.

USAGE: CALL EXPFILE (ISET,LISTA,LISTEXP,LNOD,LELE,LBC, LLD,NØD,NELE,LDBC)

PARAMETERS: I N P U T

ISET	-	Set number.
LISTA	-	If 1, no ATLAS input data to be changed, otherwise the input data will be changed.
LISTEXP	-	Output file containing the modified input data.
LNØD	-	File of new nodal data.
LELE	-	File of new element data.
LBC	-	File of new BC data.
LLD	-	File of new LØADS data.
NØD	-	If 1 then replace nodal data, otherwise NØD = 0.
NELE	-	If 1 then replace element data, otherwise NELE = 0.
LDBC	-	If 1 then replace loads and BC, otherwise LDBC = 0.

COMMON: None

SUBROUTINES: ISCAN,KØMSTR,NSCAN,LSTRNG

LANGUAGE: FØRTRAN

DISCUSSION: It LISTA is 0, only the replacement sets indicated are placed on LISTEXP. LNØD and/or LELE make up the stiffness data set.

It LISTA is a file name, it is searched for the stiffness, loads, and BC data sets. If any of these data sets are found on LISTA, and the corresponding indicator (NØD,NELE,LDBC) is turned on, that data set will be substituted in the input deck.

If there is more than one stiffness, loads, or BC data sets in the input file, just the first one is replaced.

ROUTINE: LDSCNV

AUTHOR: Kandace K. Yagi

DATE: December 1974

PURPOSE: To convert expanded ATLAS LØADS data into NASTRAN data (FØRCE,MØMENT, and SPC cards).

USAGE: CALL LDSCNV(NTAPE)

PARAMETERS: I N P U T

NTAPE - File where the NASTRAN cards are written.

COMMON: CØMMØN/CARDS/ A LØDAREC common block.

CØMMØN/LRCØMM/ A LØDAREC common block.

SUBROUTINES: LØDAREC,ØPTFØR,SQRT

LANGUAGE: FØRTRAN

DISCUSSION:

1. The ATLAS input card images are assumed to be on the input file. If a different file is wanted, NINT in the common block LRCØMM has to be changed.
2. Only one card image is kept at a time.
3. Nodal loads data is converted into FØRCE and MØMENT cards.
4. SUPPØRT DISPLACEMENT cards are converted into SPC cards.
5. Cards not recognized by this program trigger a warning message and are then ignored.
6. LDSCNV returns when an END LØADS card is reached.

ROUTINE: LDPRT

AUTHOR: Kandace K. Yagi

DATE: March 1975

PURPOSE: To store the loads in a dummy matrix and then use that matrix to print ATLAS cards in expanded form.

USAGE: CALL LDPRT(L,LRØW,KACV,KFAV,IDUMM,NN,NCT,MATR, LDWP,LIST)

PARAMETERS: I N P U T

L	-	Loads matrix.
LRØW	-	Number of rows in L.
KACV	-	The assembly control vector.
KFAV	-	The freedom activity vector.
IDUMM	-	A dummy matrix.
NN	-	Number of nodes in set.
NCT	-	Nodal correspondence matrix.
MATR	-	15 if free loads matrix 30 if retain loads matrix 45 if support loads matrix.
LDWR	-	1 if this is the last matrix to be read in and a print is wanted. 0 if no print is wanted.
LIST	-	File where the ATLAS card images are to be printed.

O U T P U T

None

COMMON: CØMMØN/LØADS/NFØRC,NF,LC,LCR

Common block to communicate with the calling program.

NFØRC	-	Array which holds the active freedom numbers.

NF - Number of active freedoms.

LC - Number of load cases the dummy matrix will hold.

LCR - Number of load cases processed to date.

SUBROUTINES: SHIFT,NCUIR

LANGUAGE: FØRTRAN

DISCUSSION:

1. In the calling program IDUMM should be set up such that the number of rows equals NN and the number of columns equals NF times LC.

2. LDPRT goes through L only looking at load cases LCR + 1 to LCR + LC. It reads each row of L finding out which node and freedom it represents by examining KACV and KFAV. It then puts the information from L into its proper place in IDUMM.

3. If LDWR equals 1, LDPRT will print the expanded ATLAS load cards for load cases LCR + 1 through LCR + LC using only the non-zero load values in IDUMM.

No error or warning messages are issued.

ROUTINE: MASSCNV

AUTHOR: M. Tamekuni

DATE: March 1975

PURPOSE: To create NASTRAN card images from the internal form of ATLAS data.

USAGE: CALL MASSCNV(NSET,NCØND,NØUT,NRØWS,NR)

PARAMETERS:

I N P U T

NSET - Set number.

NCØND - Stage number.

NRØWS - Number of rows in KRFV matrix.

NR - Number of retained nodes.

O U T P U T

NØUT - File where the NASTRAN output goes.

COMMON: CØMMØN/KQRNDM/ A system common block.

CØMMØN/KQBUFF/ A system common block.

CØMMØN/KERRØR/ A system common block.

SUBROUTINES: SHIFT,INCR,IFRREC,INCRB,RETNØDE,CNVMASS

LANGUAGE: FØRTRAN

DISCUSSION: MASSCNV contains SNARK statements.

MASSCNV reads the appropriate matrices and then calls RETNØDE and CNVMASS to do the actual conversion to card images.

Error messages are issued when matrices needed for the conversion are not found.

ROUTINE: NØDCNV

AUTHOR: Kandace K. Yagi

DATE: December 1974

PURPOSE: To convert expanded ATLAS nodal data into NASTRAN data (GRID,MPC, and SEQGP cards).

USAGE: CALL NØDCNV(NTAPE)

PARAMETERS:

NTAPE	-	The file where the NASTRAN card images are written.

COMMON:

CØMMØN/CARDS/		A LØDAREC common block.
CØMMØN/LRCØMM/		A LØDAREC common block.
CØMMØN/INCREMT/		Common block which holds the increments used when one ATLAS card creates two or more NASTRAN cards. The increments are added to the ATLAS user number to create new NASTRAN user numbers.
NL,NU	-	SPAR and COVER mid-nodes (default - 10000,20000)
IPL1,IPL2	-	PLATE stiffening (default - 10000,20000)
IRØDU,IRØAL	-	Upper and lower RODS of SPAR (default - 10000,20000)
ICVU,ICVL	-	Basic COVER (default - 10000, 20000)
ICVU1,ICVU2	-	Upper COVER stiffening (default - 12000,14000)
ICVL1,LCVL2	-	Lower COVER stiffening (default - 22000,27000)

SUBROUTINES: LØDAREC,ØPTFØR

LANGUAGE: FØRTRAN

DISCUSSION:

(1) The ATLAS input card images are assumed to be on the input file. If a different file is wanted, NINT in the common block LRCØMM has to be changed.

(2) Only one card image is kept at a time.

(3) Node cards specifying three coordinates are converted into one GRID card. Node cards specifying four coordinates are converted into two GRID cards and one MPC card.

(4) A SEQGP card image is made after every fourth GRID card is created.

(5) NØDCNV does not check for unrecognized cards or for EØF's before the END NØDAL DATA card.

(6) NØDCNV returns when an END NØDAL DATA card is reached.

No error or warning messages are issued from NØDCNV.

ROUTINE: NØDEXP

AUTHOR: S. Wahlstrom

DATE: June 1972

PURPOSE: To create expanded ATLAS input cards from the nodal data matrix.

USAGE: CALL NØDEXP(NØDMAT,RN,NN,NCT,FILE,FØRMAT)

PARAMETERS: I N P U T

NØDMAT	-	Nodal data matrix.
RN	-	Row dimension of NØDMAT.
NN	-	Number of nodes in set.
NCT	-	Nodal correspondence table.
FILE	-	File on which the expanded ATLAS input data are written.
FØRMAT	-	The punch format GE 0 desired field width (7 ≤ FORMAT ≤ 15) LT 0 desired field width but decimal point line up.

COMMON: None

SUBROUTINES: ØPTFØR,NCUIR

LANGUAGE: FØRTRAN

DISCUSSION: The nodes are printed out in user order.

No error messages are issued.

ROUTINE: PUNØNEL

AUTHOR: S. Wahlstrom

DATE: June 1972

PURPOSE: To create expanded ATLAS input card images of ELEMENT data.

USAGE: CALL PUNØNEL(ID,MAT,TEMP,USERID,NØDES,NNØDES,
PRØP,PRØPFW,NPRØP,FIL)

PARAMETERS: I N P U T

ID	-	Element type (INTEGER)
MAT	-	Material code.
TEMP	-	Temperature.
USERID	-	ELement user id (integer).
NØDES	-	Vector of user node numbers.
NNØDES	-	Number of nodes.
PRØP	-	Vector of element properties.
PRØPFW	-	Field width used for section properties. (integer)
NPRØP	-	Number of properties.
FIL	-	Output file.

COMMON: None

SUBROUTINES: ØPTFØR

LANGUAGE: FØRTRAN

DISCUSSION: PUNØNEL takes the input information and creates one ATLAS element card image.

No error or warning message is printed.

ROUTINE: RETEXP

AUTHOR: M. Tamekuni

DATE: March 1975

PURPOSE: To create ATLAS RETAIN card images.

USAGE: CALL RETEXP(NRET,NCT,NØUT,NDIM)

PARAMETERS: I N P U T

NRET	-	Retained freedom vector (KRFV).
NCT	-	Nodal correspondence table.
NØUT	-	Output file.
NDIM	-	Dimension of NRET

COMMON: None

SUBROUTINES: SHIFT,NCUIK

LANGUAGE: FØRTRAN

DISCUSSION: RETEXP takes the information from NRET and NCT and creates expanded ATLAS input data.

No error or warning messages are issued.

ROUTINE: RETNØDE

AUTHOR: M. Tamekuni

DATE: March 1975

PURPOSE: To compute the number of retained nodes and create a vector containing those nodes.

USAGE: CALL RETNØDE(KACV,NR,NRETDØDE,NRET)

PARAMETERS: I N P U T

KACV - Assembly control vector.

NR - Number of nodes.

O U T P U T

NRETNØD - Number of retained nodes.

NRET - Vector of retained nodes.

COMMON: None

SUBROUTINES: None

LANGUAGE: FØRTRAN

DISCUSSION: A pass is made through KACV and a check is made for non-zero freedoms. These non-zero freedoms are counted and put into the vector.

No error or warning messages are issued.

ROUTINE: SDEXP

AUTHOR: M. Tamekuni

DATE: March 1975

PURPOSE: Create ATLAS data SUPPØRT card images for inactive freedoms.

USAGE: CALL SDEXP(KFAV,NCT,NØUT,NNØDES)

PARAMETERS: I N P U T

KFAV	-	Freedom activity vector.
NCT	-	Nodal correspondence table.
NØUT	-	Output file.
NNØDES	-	Number of nodes.

COMMON: None

SUBROUTINES: SHIFT,NCUIK

LANGUAGE: FØRTRAN

DISCUSSION: SDEXP takes the information from KFAV and NCT and writes it to NØUT in the form of expanded ATLAS input data.

No error or warning messages are printed.

REFERENCES

1-1 K. L. Dreisbach, ed., ATLAS--An Integrated Structural Analysis and Design System, "Users Manual - Input and Execution Data," NASA CR-159043, July 1979.

1-2 F. P. Gray, ed., ATLAS--An Integrated Structural Analysis and Design System, "Random Access File Catalog," NASA CR-159044, July 1979.

10-1 W. J. Erickson, "SNARK User's Manual," BCS-G0686, Boeing Computer Services, 1975.

702-1 L. D. Richmond, "A Rational Method of Obtaining Three-Dimensional Unsteady Aerodynamic Derivatives of Intersecting Airfoils in Subsonic Flow." D6-7401, Boeing Commercial Airplane Company, April 1962.

704-1 J. M. Li and D. F. Tankersley, "Unsteady Aerodynamic Interaction for Advanced Aircraft Configurations in Subsonic Flow (Improved Doublet-Lattice TEV155)," D6-41258, Dec. 1973.

706-1 R. N. Desmarais and R. M. Bennett, "An Automated Procedure for Computing Flutter Eigenvalues," AIAA Paper No. 73-393, presented at the AIAA/ASME/SAE 14th Structures, Structural Dynamics and Materials Conference, Williamsburg, Virginia, March 20-22, 1973.

709-1 J. M. Li, C. J. Borland and J. R. Hogley, "Prediction of Unsteady Aerodynamic Loadings of Non-Planar Wings and Wing-Tail Configurations in Supersonic Flow," Part I--Theoretical Development, Program Usage and Application, AFFDL-TR-71-108, Wright-Patterson Air Force Base, Ohio, March 1972.

713-1 W. S. Rowe, B. A. Winther and M. C. Redman, "Prediction of Unsteady Aerodynamic Loadings Caused by Trailing Edge Control Surface Motions in Subsonic Compressible Flow," NASA CR-2003, June 1972.