NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

THE JOSHUA SYSTEM

VOL. 2 USER'S GUIDE

Approved by

H. C. Honeck, Research Manager
Computer Applications Division

NOTICE

~~This document contains descriptions of the JOSHUA Operating System and Applications Modules. The document is prepared as a set of loose-leaf notebooks which will be periodically updated. All of the material is preliminary and should not be referenced.~~

September 1973

E. I. DU PONT DE NEMOURS & COMPANY
SAVANNAH RIVER LABORATORY
AIKEN, S. C. 29801

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

CONTRACT AT(07-2)-1 WITH THE

UNITED STATES ATOMIC ENERGY COMMISSION

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

JOSHUA Vol. 2 USER'S GUIDE

- Part A Design Concepts
- Part B Programming Facilities
- Part C Module Preparation and Execution
- Part D Utility Programs (Deleted)
- Part E Monitor System (JMS)

JOSHUA
DPSTM-500

Title	Part A - Design Concepts	Vol.	Page
		II	A.O.1
Rev.		Rev.	Date
			10/70

Contents

- A.1 - Design Concepts
- A.2 - An Example - Part I

JOSHUA DPSTM-500	Title Part A - Design Concepts	Vol.	Page
		II	A.0.2
		Rev.	Date
			10/70

(Blank)

Work Done By

J. E. Suich
H. C. Honeck
J. C. Jensen

Report Written By

J. E. Suich
H. C. Honeck
J. C. Jensen

Contents

	<u>Page</u>
1. Introduction	A.1.3
2. Review of JOSHUA System	A.1.4
3. Module Design Concepts	A.1.5
4. Data Base Design Concepts	A.1.7
4.1 Data Structures	A.1.7
4.2 Data Selection	A.1.10
4.3 Data Organization	A.1.11
4.4 Data Names	A.1.13
4.5 Data Sets	A.1.15
4.5.1 General	A.1.15
4.5.2 Standard Data Sets	A.1.15
4.5.3 User Data Sets	A.1.15
4.5.4 Job Data Sets	A.1.16
4.5.5 SCRATCH Data Set	A.1.16
4.5.6 Data Set Hierarchy	A.1.16
4.5.7 Data Set Maintenance	A.1.17

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.2
		Rev.	Date 10/70

(Blank)

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.3
		Rev. 1	Date 3/73

1. Introduction

This volume contains information pertaining to the design, programming, and use of JOSHUA modules, and the use of the JOSHUA Monitor System (JMS). The design concepts discussed in this section are an extension of the concepts discussed in Volume I, Part A, and should be helpful to analysts, programmers, and users of the system. Parts B and C are intended primarily for use by analysts and programmers, and presuppose a knowledge of both IBM 360 FORTRAN and OS/360 facilities.

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.4
		Rev. 1	Date 3/73

2. Review of JOSHUA System

The introductory material of Volume I has discussed the need for a modular computational system to enable use of very large computer programs as "subroutines" in an over-all procedure. The problem of manipulating these programs, and arranging for communication of data between them, is beyond the capability of "batch processing" operating systems currently in use.

The JOSHUA system addresses itself to this problem by providing for:

- "modules" that reflect the computational structure of the over-all problem
- "named-data" that reflect the information structure associated with the problem
- "user access" to the data, in convenient form for manipulation, at an on-line display terminal
- "job management" services to allow the user to conveniently set up for, and control, problem solution.

These four concepts constitute the new approaches taken with JOSHUA; use is also made of other features embodied in the standard batch processing operating system.

3. Module Design Concepts

A "computational module" is a logically complete, dynamically executable computer procedure, which receives input only from named data records, and places its results only in other named data records. Since named data records are "permanently" stored in the system, they represent information which is of lasting value (significance) to the user - they are not "temporary", or intermediate computed results. Thus, the notion of a logically complete module is equivalent to the notion of a procedure which produces new information of lasting value from old information also of lasting value.

A "computational path" is an automated sequence of modules which satisfies a particular application. In the simplest case, a "path" will be no more than the provision for executing a single module. In more complicated cases, however, a path may involve execution of several modules, perhaps even iteratively, to accomplish the total procedure required. Thus, a path may be thought of as a program written in a problem-oriented language.

Questions which must be asked concerning a proposed computational procedure in order to define the appropriate modular structure are:

- Which collections of the procedures involved constitute "modules"?
- How can these modules be written to achieve sufficient generality to suit various applications?
- What use can be made of previously defined modules?
- Do there exist, computer codes outside the JOSHUA system which can be adapted to the present purpose?
- If so, how should they be restructured to fit the system design philosophy?
- Which data generated by this module are of "lasting value", and how should they be named?

and, with respect to the paths involved:

- Which modules and previously defined paths are involved in constructing the present path?
- Would simple modifications to existing paths and modules permit their use in the present case, and reduce redundancy in the system?
- What options should be granted the user to allow for conversational control of the path? Does this indicate construction of several sub-paths?

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.6
		Rev.	Date 10/70

In answering these questions, the total system of applications should be kept in mind, because of the interactions between module and data requirements for the several applications in which they may be used.

Thus, the notion of "logically complete" modules is equivalent to determining which pieces of a complete calculational procedure are useful in themselves, as evidenced by their appearance in several such procedures. In this sense, a module is equivalent to a FORTRAN "subroutine" of reasonably wide utility, that solves a physically meaningful piece of a problem. In place of subroutine "arguments", named data records are utilized to define the interfaces between modules. The "path" may then be thought of as a "main program" consisting of a series of subroutine calls plus control logic.

Modules should be small enough pieces of a complete procedure that:

- modifications to a single module may be easily made;
- a great deal of coding is not duplicated in several modules in the system;
- commonly used options can be easily accommodated by interchange of modules.

On the other hand, modules should be large enough pieces of a complete procedure that:

- a modest number of modules constitutes a path;
- the standard module interfaces are not temporary data of no interest to the user;
- the module logic specifications are independent of the logic of other modules in the system.

These two requirements on module size may not always be compatible: to resolve difficulties the designer may have recourse to common subroutines in a library, overlay structures, or "paths" which are used precisely as modules, containing modular procedures with temporary data interfaces.

4. Data Base Design Concepts

4.1 Data Structures

A large computing system with many modules (all interfacing through direct access storage) creates, and requires the management of, a very large amount of data. Both the system user and the program developer are faced with the formidable problem of keeping track of all of this data and referring to it repeatedly. The first activity demands that the data be referred to by the user in his natural language, and the second activity demands that the data be handled efficiently by the computing system.

Figures 1 and 2 address themselves to the problem of handling large quantities of data in a natural way. Figure 1 is a diagram of a typical multigroup cross section library, showing that the data are structured in the user's mind in a hierarchical fashion, that is, as a tree structure. For example, it is most natural for the user to indicate that he is interested in the group 1 value of the elastic P_c matrix at 20°C of ^{238}U in the deluxe version of the multigroup library. (Note that the logical hierarchy of qualifiers is usually the reverse of English language usage.). He cannot, without a great deal of research, refer to this data as, for example, FORTRAN record 1057. Similarly, the library of lattice cell parameters contains data which the user thinks of in a hierarchical structure. That is, he would, for example, like to know the annular geometry of standard type Mark 14 fuel assembly, (Figure 2). This type of data structure is provided for in the JOSHUA system through the use of "named" data records. The record names consist of a series of meaningful qualifiers that are stored in a tree structured catalog.

Figure 1. Typical Multigroup Cross Section Library.

Figure 2. Typical Cell Data Library

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.10
		Rev. 1	Date 3/73

In developing the data structures for use in the JOSHUA system, the following considerations are important:

- Which data should be catalogued and saved?
- What are the logical relationships between the data elements involved?
- How should the data be grouped to best suit the module(s) which will use it?
- Which data are already in the system, and how should they be used?
- Which elements of the data are to be input by the user, which are to be computed, and which either input or computed depending on the application?
- How should the data be named in order to best express the logical relations between records in the natural terminology of the user's discipline?

4.2 Data Selection

Selecting the data to be included is the most difficult part of designing a data set. Several factors must be considered:

- Lasting value of the data.
- Utility of the data.
- Frequency of use of the data.
- Regeneration of data.

Since the data base resides on on-line direct access storage, it is also necessary to design the data sets so that they contain only the essential data.

If data is to be saved, there must be some lasting value to the data either because

- the user will need to look at this data at some future time, or,
- the data is required in later calculational steps.

The time element is important here. If the user wants to look at the data at the end of a job to verify that the job was done properly, and there is little need to look at the data at a later time, the data should be placed on a temporary data set. This data set can be displayed at the end of job execution but is not permanently saved. If the data

is needed for a later calculational step, will that step always be executed following this step? If that data has no lasting value, it should be transmitted to later calculational steps using a "scratch" data set. A scratch data set is active until a job is completed, but is deleted at the end of the job.

If it is determined that there is some lasting value to the data, is it better to

- permanently store the data, or,
- recalculate it?

As an example, consider a lattice physics module which computes the flux $\phi(r,E)$ at 50 space points and 80 energy points. Since other results can be deduced from this flux, it has a lasting value and should be stored.

In the case of few-group cross sections it is clearly better to calculate and store them at the point in time when the flux and basic cross section data are core resident. A later module would have to make many accesses to both the flux and basic cross section data to enable the calculation of this data.

A third example is a neutron balance sheet. Like few-group cross sections, many accesses to the flux data and cross section data are required to compute a neutron balance sheet. In this case, however, it is probably not justified to devote a few thousand words of disk space for each lattice to a neutron balance sheet since

- it is not used in later calculation steps
- it is not frequently examined by the user
- the total time required to generate it is small (probably less than a minute).

It would be better to have a special module to prepare a neutron balance sheet and other tables for human consumption and display them as needed.

4.3 Data Organization

Having decided what data is to be stored, the next step is to organize the data into named data records. The following things must be considered:

- logical association
- module requirements

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.12
		Rev.	Date 10/70

- record size
- future expansion
- display requirements
- list complexity

Organizing the data into groups by logical association provides more than just esthetic appeal. People (programmers and users) must use the data, and the easier it is to remember where the data is and what it means, the easier it will be to develop and use the system. If the data is used by other modules, their requirements must be considered. For example, nearly all modules using the data set named 'MULTIGRP' will need cross sections and transfer matrices blocked by final energy group. Since this data set is primarily for use by modules and not users, it is structured in this way. When several modules with conflicting requirements are involved, a compromise slanted toward the module with the greatest expected use must be made.

If a record is intended primarily for display purposes, the size of the record should be small, perhaps 10 - 100 numbers. If a record is intended primarily for use by other modules, it should be larger, perhaps 50 - 500 numbers. It takes an average of about 10 words (40 bytes) to store the record name and associated pointers. If the record contains only 10 words, then 50% of the disk space is allocated to the name and pointers, and 50% to the data (50% efficiency). If the data record contains 100 words, the storage efficiency is about 90%. It would be desirable to achieve an over-all storage efficiency of at least 80%.

A major advantage of using named data records is the convenience for future expansion. Named data records defined now (if reasonable foresight is used) will not have to be modified when future modules require additional data. New data records can be defined as needed. The presence of these new records will not affect the operation of older modules. It is desirable, however, to anticipate the needs of future modules and include sufficient data (where possible) in the data sets so that the new data needed can be deduced from the data already stored.

It is a programming convenience to have simple I/O lists. A simple list is also easier to display and provides a more compact and readable display. If the record is primarily for display purposes, the list should be tailored to the display. Simple lists will greatly facilitate the future development of information retrieval capability. It is recommended that the I/O list be no more complex than a single two-dimensional array or several one-dimensional arrays.

Title	Design Concepts	Vol.	Page
		II	A.1.13
Rev.		Date	
		10/70	

4.4 Data Names

Let us consider an example of a named data record. Suppose that we have decided that a data record is needed to contain the multigroup cross sections for U235. We might then pick the qualifiers to be MULTIGRP, XSECTION, and U235. Since the cross sections are temperature dependent and we would like to have several versions of the library, we might pick 20 and VERSN1 as additional qualifiers. These qualifiers separated by periods are the record name.

Record Name: MULTIGRP.VERSN1.U235.20.XSECTION

With the exception of the first qualifier (the data set name) we can order the qualifiers in any way we like (guidelines for this are given later).

Suppose the cross sections are given in an array XS(I,J). The corresponding list is

List: NTYPE,NEG,((XS(I,J),I=1,NEG),J=1,NTYPE).

The combination of the record name and list defines a specific named data record. It is clear that the same name and list structure can be used for other isotopes, other temperatures, and other versions of the library. Thus, we might consider general names of DISØ, TDC, and VERSN for these parameters define a record name type.

Record Name Type: MULTIGRP.?VERSN.?DISØ.?TDC.XSECTION

Qualifiers starting with a question mark are called variable qualifiers. The combination of a record name type and an I/O list defines a named data record type. The example above indicates that the same name and list structure is used for all isotopes, temperatures, and versions. At module execution time the variable qualifiers must be replaced with a literal string, a Fortran variable, or an integer constant. For example, a read statement in a module might be written as -

```

READ('MULTIGRP'. 'V1'. DISØ(N). 20. 'XSECTION')
C      NTYPE,NEG,((XS(I,J),I=1,NEG),J=1,NTYPE)

```

The name of the isotope requested is in DISØ(N). A precompiler converts this statement to a set of ordinary Fortran IV statements. At execution time these statements transmit the record name, replacing the variable DISØ(N) with its current value, to the data management routine which searches its catalogue for the name. If the name and associated pointers are found, the data is read from the disk into the core storage locations

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.14
		Rev.	Date 10/76

specified in the list.

The reason for identifying data by name is to make it convenient for the user at a terminal to interrogate the data. If the names are not a part of the user's natural language, or do not have a high mnemonic content, the usefulness of the names is lost. There are several other considerations in choosing names:

- hierarchy
- disk ordering
- uniqueness

Qualifiers in a name should follow a hierarchical structure starting with the highest order name first. Unfortunately, in the English language, one generally starts with the lowest order name first. One would state, "I want the cross sections at 20°C for U235 from the standard version of the multigroup library". Clearly, the proper order is

MULTIGRP.STANDARD.U235.20.XSECTION

When other cross section types (secondary cross sections), other data types (transfer matrices, resonance parameters, etc.), and other isotopes are considered, the entire collection of names fall into a well-ordered tree structure. (See Figure 1).

The data manager writes records on the disk sequentially within a data set. All records of the MULTIGRP data set are in one area of the disk but they appear in the order written. Maintenance programs will reorder the records according to the hierarchy of their names. This means that, in the example above, all cross sections for ²³⁵U are in contiguous areas of the disk. This grouping provides the data manager with the possibility for "looking ahead", since, if a request has been made for data from one part of the tree, there is a good chance that the next request will be for data in the same part of the tree and hence the same area of the disk.

Care must be taken that the names are unique. If the above example were a valid name, the names

MULTIGRP.STANDARD.U235
 MULTIGRP.STANDARD
 MULTIGRP

are not valid record names since they refer to groups of records, i.e., a subtree of the data set. The name MULTIGRP.STANDARD.U235.20.XSECTION.FISSION is also invalid since the first five qualifiers already define a single record - not a group of records. A record name must

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.15
		Rev. 1	Date 3/73

uniquely point to a single record.

4.5 Data Sets

4.5.1 General

The first qualifier of a record is referred to as the JOSHUA data set name, e.g., the record MULTIGRP.STANDARD.U235.20.XSECTION is a member of the MULTIGRP data set. Each data set and an associated tree catalog exist on direct access storage. A data set name catalog is used by the data management routine to locate the proper tree catalog for each JOSHUA data set.

Four types of data exist in the system; Standard, User, Job, and Scratch. The data are divided into different types to preserve the integrity of the data base, avoid concurrent updating of data sets in a multiprogramming environment, and to ease the burden of data set maintenance.

4.5.2 Standard Data Sets

Standard data sets contain data of a "permanent" nature that will be used by many different modules within the system. This data is read-only for all users. These data sets are updated only by designated personnel.

4.5.3 User Data Sets

User data sets contain data of a "semi-permanent" nature. "Semi-permanent" alludes to the fact that the storage space available for each user is finite; all data cannot be saved. This data may be updated by authorized users by means of JOSHUA monitor functions, but may not be updated by a computational module. The data may also be accessed but not updated by other users.

Data names appearing in a User data set may be identical to those in a Standard data set. The data management routine will locate the user record if identical names exist*, thus permitting the user to substitute data values for those contained in the Standard data sets. Data set names that have no corresponding Standard data set may also be used.

* See Section 4.5.6, Data Set Hierarchy.

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.16
		Rev. 1	Date 3/73

4.5.4 Job Data Sets

Job data sets contain the results of computational modules. This type data may also be generated and updated by authorized users through the use of JOSHUA monitor functions, and may be accessed by any user. This data may or may not be of a lasting or permanent nature.

Data names appearing in a Job data set may be identical to those in a User or Standard data set. The data management routine will locate the job record if identical names exist*, thus permitting a module to substitute data values for those contained in either Standard or User data sets. Data set names that have no corresponding Standard data set may also be used.

A Job data set is provided solely for temporary storage of module-generated data. The user is expected to examine the data as soon as practical after the completion of a job. He may then copy data of lasting value to his User data set and delete the Job data set. Job data sets that remain in the system for an unnecessary length of time will be automatically deleted. A "reasonable" length of time should be a few days but will depend upon the amount of direct access space that is available for allocation.

4.5.5 Scratch Data Set

A special data set named "SCRATCH" is available to computational modules for intermediate results within a job step, i.e., execution of a module or path. The entire data set is deleted at the beginning of each job step.

4.5.6 Data Set Hierarchy

Due to the existence of several types of data sets, each possibly containing identical record names, the data management routine must be concerned with the hierarchy of their use. Also, to meet the requirements of each data set type, the hierarchy is necessarily different for computational modules and the terminal monitor.

The following conventions are used:

A. Computational Modules

- 1) All I/O requests, (READ, WRITE, DELETE), for SCRATCH data are unaltered.

* See Section 4.5.6, Data Set Hierarchy

- 2) All other READ requests are altered to cause a search of the Job catalog first, then the User catalog, and finally the Standard catalog. This sequence may be changed by the user at run time or by a statement in a computational module.
- 3) All other WRITE or DELETE requests are altered to cause updating of the Job catalog only.

B. JOSHUA Monitor Functions

- 1) SCRATCH data may not be accessed from a terminal.
- 2) READ requests are altered to cause a search of the Job catalog first, (if the user has specified a particular job), then the User catalog, and finally the Standard catalog. However, the user may direct a request to a specific catalog if desired.
- 3) WRITE or DELETE requests are normally altered to cause updating of the User catalog only. This option may be changed by a user if he is authorized to update another data set.

4.5.7 Data Set Maintenance

Maintenance of JOSHUA data sets is necessary for the following reasons:

- 1) When the name of a record or group of records is deleted from a catalog, the space that is occupied by both the name in the catalog and the data itself remains unavailable for other uses.
- 2) If data records are created in an order that is not logically consistent with the logical order of the name qualifiers in the tree catalog, the retrieval of the data will be inefficient.

Data sets therefore need to be periodically "cleaned up" to recover the wasted space and to reorder the catalog and the data for efficient retrieval.

JOSHUA DPSTM-500	Title Design Concepts	Vol. II	Page A.1.18
		Rev. 1	Date 3/73

The maintenance of the data sets is accomplished by copying the data to a temporary data set. This procedure ignores deleted names and reorders the data. The data and catalog are then written to backing storage, (usually tape), the original data set is deleted, and the temporary data set is copied back to the original data set. This procedure is accomplished with a JOSHUA monitor function called MAINTAIN. Other monitor functions are also provided that assist the user in this and other types of data set maintenance.

JOSHUA DPSTM-500	Title An Example - Part I	Vol. II	Page A.2.1
		Rev.	Date 10/70

Work Done By

H. C. Honeck

Report Written By

H. C. Honeck

Contents

	<u>Page</u>
1. Introduction	A.2.3
2. The Problem	A.2.3
3. Data Set Definition	A.2.5

JOSHUA DPSTM-500	Title An Example - Part I	Vol. II	Page A.2.2
		Rev.	Date 10/7

(Blank)

1. Introduction

It is always helpful to illustrate the formal specifications of a system with an example. This manuscript along with B.2, C.2, and E.2 describe the construction of a data set and the construction and execution of a sample module. Since most existing modules are far too complex to serve as examples, a simple illustrative module and data set were designed solely for this purpose.

2. The Problem

The problem is to define and create a data set containing one-group cross sections. A data set is defined because there is a need for the data by either calculational modules or reactor designers. The need for such a data set might be twofold. Three-dimensional reactor calculations might be done using only one energy group. Following the flux calculation a series of sensitivity studies might be done which require one-group cross sections by reaction type, isotope, temperature, energy spectrum (characterized by the lattice type), and possibly, buckling. The second need is for reactor designers to see how cross sections vary with spectrum and buckling. The requirements are then for a data set which contains one-group averaged microscopic cross sections catalogued by:

1. Isotope
2. Temperature
3. Spectrum characterized by the lattice type.
4. Spectrum changes caused by leakage as characterized by buckling.
5. Reaction type.

We will call this the ØNEGRP data set and define it in the next section.

If we examine the standard data sets in the JOSHUA system (See Volume III), we find that all data needed to compute one-group cross sections are contained in the MULTIGRP and LATTICE data sets. The MULTIGRP data set contains microscopic cross sections by:

1. Isotope
2. Temperature
3. Reaction type.
4. Energy group.

NAME - 'MULTIGRP'.?VERSN?.?DISC?.?ITDC?. 'XSECTION'

LIST - NEG,(XTOT(J),XCAP(J),XFIS(J),XNU(J),XTR(J),XN2N(J),
CHI(J),J=1,NEG)

SYMBOL	DEFINITION	TYPE
CHI	- FISSION FRACTION IN GROUP J VALUE IS ZERO IF ICHI.NE.1. SEE Q.CONTENTS RECORD.	R4
DISC	- EIGHT-CHARACTER ISOTOPE NAME	C8
ITDC	- INTEGER VALUE OF TEMPERATURE (DEG C)	I4
NEG	- NO. OF ENERGY GROUPS	I4
XNU	- NEUTRONS/FISSION IN GROUP J	R4
XTR	- TRANSPORT CROSS-SECTION (BARNs) IN GROUP J	R4
XCAP	- CAPTURE CROSS-SECTION (BARNs) IN GROUP J	R4
XFIS	- FISSION CROSS-SECTION (BARNs) IN GROUP J	R4
XN2N	- (N,2N) CROSS-SECTION (BARNs) IN GROUP J	R4
XTOT	- TOTAL CROSS-SECTION (BARNs) IN GROUP J	R4
VERSN	- VERSION NAMES	C8

Figure 1. Smooth Cross-Sections.

These data are contained in the XSECTION type record described in manuscript III.B.3 and reproduced in Figure 1. We will use only the total, capture, fission, ν , transport, and (n,2n) reaction types. We also need the ISOTOPES record shown in Figure 2.

The multigroup cross sections must be weighted by a neutron spectrum to obtain a one-group value. The LATTICE data set contains total neutron fluxes by:

1. Lattice name and version.
2. Search option (related to a Buckling).
3. Legendre order.
4. Energy group.

These data are contained in the KEFF and LFLUX type records described in manuscript III.D.2 and reproduced in Figures 3 and 4.

Modules are defined to create data sets from other data sets. The above three data sets plus the formulas

$$\bar{\sigma}^x = \frac{\sum_{i=1}^{NEG} \sigma_i^x \phi_i}{\sum_{i=1}^{NEG} \phi_i}$$

$$\bar{\nu} = \frac{\sum_{i=1}^{NEG} \nu_i \sigma_i^f \phi_i}{\sum_{i=1}^{NEG} \sigma_i^f \phi_i}$$

define module CØG to Create One-Group data. This is illustrated in Figure 5 where an INPUT data set has also been shown. This data set supplies CØG with the few additional control parameters required.

3. Data Set Definition

The ØNEGRP data set is to contain one-group microscopic cross sections catalogued by:

1. Isotope
2. Temperature
3. Lattice
4. Buckling
5. Reaction type

JOSHUA DPSTM-500	Title An Example - Part I	Vol.	Page
		II	A.2.6
		Rev.	Date
			10/7

```

-----
| NAME - 'MULTIGRP'.?VERSN?. 'ISOTOPES'
| LIST - NEG, ISOX, (DISO(N), ITDC(N), N=1, ISOX)
|
-----

```

SYMBOL	DEFINITION	TYPE
DISO	- ISOTOPE NAME	C8
ISOX	- NUMBER OF CATALOGUED ISOTOPE/TEMPERATURE PAIRS	I4
ITDC	- INTEGER VALUE OF TEMPERATURE (DEG C)	I4
N	- ISOTOPE INDEX	I4
NEG	- NUMBER OF ENERGY GROUPS	I4
VERSN	- NAME OF A VERSION OF THE DATA SET	C8

Figure 2. List of Isotopes in Version of Data Set.

NAME - 'LATTICE'.?DLAT?.?VLAT?.'KEFF'

LIST - NSCX,(LSOPT(NSC),KEFF(NSC),BSQD(NSC),MSQD(NSC),ALPHA(NSC),
NSO=1,NSOX)

SYMBOL	DEFINITION	TYPE
ALPHA	TIME EIGENVALUE (1/SEC)	R4
BSQD	BUCKLING (1/CM**2)	R4
DLAT	LATTICE NAME	C8
KEFF	K EFFECTIVE	R4
LSOPT	SEARCH OPTION FLAG FOR BL CALCULATION 1 - KEFF CALCULATION 2 - BUCKLING SEARCH 3 - TIME EIGENVALUE (ALPHA) SEARCH 4 - CONCENTRATION SEARCH	I4
MSQD	MIGRATION AREA (CM**2)	R4
NSO	SEARCH INDEX (SEE LSOPT)	I4
NSOX	NO. OF SEARCHES DONE (SEE LSOPT)	I4
VLAT	LATTICE VERSION NAME	C8

Figure 3. Table of k_{eff} Versus Buckling.

JOSHUA DPSTM-500	Title An Example - Part I	Vol.	Page
		II	A.2.8
		Rev.	Date
			10/7

```

NAME - 'LATTICE'.?DLAT?.?VLAT?. 'LFLUX'.?NSO?
LIST - LPX,NEG,(Q(I),(LFLUX(LP,I),LP=1,LPX),I=1,NEG)

```

SYMBOL	DEFINITION	TYPE
DLAT	LATTICE NAME	C8
I	ENERGY GROUP INDEX	I4
LP	LEGENDRE INDEX = 1+LEGENDRE NUMBER	I4
LPX	MAX. LEGENDRE INDEX	I4
LFLUX	SUPERCCELL LEGENDRE FLUX (NEUTRON/SEC-CM**2)	R4
NEG	NUMBER OF ENERGY GROUPS	I4
NSO	CASE NUMBER	I4
Q	SUPERCCELL SLOWING DOWN DENSITY (NEUTRONS/SEC)	R4
VLAT	LATTICE VERSION NAME	C8

Figure 4. Legendre Flux and Q by Group for Case ?NSO?
in k_{eff} Record.

Figure 5. CØG Module Interfaces

JOSHUA DPSTM-500	Title An Example - Part I	Vol. II	Page A.2.10
		Rev.	Date 10/7

We must decide on how to structure these data and define suitable record types. By structure we mean how the above five independent qualifiers are to be ordered in a hierarchy.

We first ask if we can combine any qualifiers to reduce the number of independent qualifiers. Since temperature is actually used in MULTIGRP as a qualifier to the isotope name we can combine them into a single isotope/temperature qualifier. The number and value of Bucklings are specific to a lattice so that Buckling is a qualifier to lattice. Our list of independent qualifiers can then be reduced to three.

1. Isotope/temperature.
2. Lattice/Buckling.
3. Reaction type.

We next ask if the modules which will use ONEGRP require any particular structure. The best guess is that the favored hierarchy is

1. Isotope/temperature
2. Reaction type
3. Lattice/Buckling

since several lattices (at a variety of Bucklings) exist in a reactor. However, there does not appear to be any clear choice between 1 and 2. The above hierarchy also appears to be best for the reactor designer since he probably is primarily interested in the variation of a given cross section with lattice and Buckling.

Next consider how large the data set might be. A typical MULTIGRP data set might have 50 isotope/temperature combinations. While there may be hundreds of lattices each with several Bucklings on the LATTICE data set, a typical situation might require some 20 lattices with 5 Bucklings for a total of 100 lattice/Buckling combinations. The number of reaction types varies between 1 and 6, the actual number depending strongly on the isotope. A typical average is three (total, capture, and transport). The expected number of cross section values is then $50 \times 100 \times 3 = 15,000$.

If we were to place all of the qualifiers in the record name, we would certainly have the ultimate retrieval capability. However, we would have only one data word per record. Since data records are currently blocked on the disk at 100 words (400 bytes), we would have 1% disk packing. Further, since about 10 catalogue words are required for each record, the ratio of data words to catalogue words is 1/10. Obviously, we need larger record sizes.

We must then divide our qualifiers into those which name the record, and those which locate the cross section with the record. Suppose we choose reaction type as the variable within the record. The isotope/temperature and lattice/Buckling become qualifiers in the record name, and the record contains cross sections by reaction type. Since some reaction types may be missing, we have the choice of always entering all types (fixed length record with zeroes for missing types) or entering only the non-zero types (variable length record with additional data to relate cross section value to reaction type). In either case, we would be adding extra unnecessary data and still only obtain a record of about 6 words in size. In general, qualifiers for which data is often missing are best placed in the record name.

We now have our choice of isotope/temperature or lattice/Buckling as the locational qualifier in the record. Since, in all expected applications, the lattice/Buckling qualifier appears lowest in the hierarchy, it is the best choice. This will lead to about 100 words/record which is a convenient size both for a module to read and for a user to view at a terminal (about 25/page, 4 pages).

We next assign symbols to the qualifiers and construct a record type. Define:

DISØ	-	8 character isotope name
ITDC	-	4 byte integer temperature (°C)
REAC	-	8 character reaction type from the list TOTAL, CAPTURE, FISSION, NU, TRANSPRT, N2N
DLAT	-	8 character lattice name
VLAT	-	8 character lattice version
BSQD	-	4 byte floating point Buckling (cm ⁻²)
SIG	-	Cross section.
NCS	-	Case index. A case is a particular combination of lattice/Buckling.
NCSX	-	Number of cases.
NIR	-	Isotope/temperature/reaction index.
NIRX	-	Number of isotope/temperature/reaction combinations.

Having decided that lattice/Buckling was to be the locational qualifier, can it be positional or must it be a pointer to a particular lattice/Buckling combination; that is, should the list be

NCSX, (SIG(NCS), NCS=1, NCSX)

JOSHUA DPSTM-500	Title An Example - Part I	Vol. II	Page A.2.12
		Rev.	Date 10/70

or

IX,(NCS(I),SIG(I),I=1,IX)

If SIG(NCS) is sparse (contains many zeros), the second list is better. In our case, for a given isotope/temperature/reaction, if the record is present, we expect no zeros and the first list is preferable. Several choices exist for the record name.

'ØNEGRP'.DISØ.ITDC.REAC
'ØNEGRP'.REAC.DISØ.ITDC
'ØNEGRP'.ITDC.REAC.DISØ
etc.

The expected use of the record gives little guidance as to the preferred order. As long as we read records in the same hierarchal order as implied by the record name, the efficiency is independent of the order of the qualifiers. If the records are accessed randomly, there is some advantage to ranking the qualifiers in order of increasing variability (how many values a qualifier can assume). Thus, from a data retrieval viewpoint, the best name might be the third in the above list.

Lacking strong guidance from expected uses and retrieval efficiency, we next examine the creation of the data set. Examination of the MULTIGRP data set shows that reaction type is subordinate to isotope/temperature. If we chose the first name in the above list, and processed the MULTIGRP data in the easiest and most natural way, we would then write the ØNEGRP file in the order indicated by the qualifiers in the name. The file on disk would then be in its natural hierarchal order. If the file were searched in this order, system overhead would be minimized. Our tentative record then is

Name: 'ØNEGRP'.DISØ.ITDC.REAC

List: NCSX,(SIG(NCS),NCS=1,NCSX)

We must next relate the case number (NCS) with a particular lattice/buckling (DLAT,VLAT,BSQD). Since these parameters characterize the neutron spectrum used to weight the cross sections, we can create a SPECTRA record which contains the required correspondence.

Name: 'ØNEGRP'. 'SPECTRA'

List: NCSX,(DLAT(NCS),VLAT(NCS),BSQD(NCS),NCS=1,NCSX)

A module using the ØNEGRP data set would attempt to read the cross section records for all combinations of DISØ, ITDC, and REAC that the module required. Some of these records would be missing (e.g., fission for Hydrogen) and unnecessary disk accesses (for catalogue pages) wasted. A useful additional record would

JOSHUA DPSTM-500	Title An Example - Part I	Vol. II	Page A.2.13
		Rev.	Date 10/70

list the name combinations that were actually present on the data set. We might call this the ISOTOPES record.

Name: 'ØNEGRP'. 'ISOTOPES'

List: NIRX, (DISØ(NIR), ITDC(NIR), REAC(NIR), NIR=1, NIRX)

The user module can read this record and avoid unnecessary data requests.

Having defined all of the records we need, we must re-examine the names. To the system these names look like

```

'ØNEGRP'.   ?   .   ?   .   ?
'ØNEGRP'. 'SPECTRA'
'ØNEGRP'. 'ISOTOPES'

```

What would the system do if an isotope named SPECTRA were encountered? We need to uniquely distinguish the first record from the other two. Since it contains cross sections, call it the XSECTION record and change its name to

```

'ØNEGRP'. 'XSECTION'. DISØ. ITDC. REAC

```

We note that there are no variable qualifiers in the SPECTRA and ISOTOPES record. Hence, there is only one such record on the data set. Is there a need to further qualify these names? Examination of the MULTIGRP and LATTICE data sets shows that they both depend on the version of the MULTIGRP data set being used. Basically, the MULTIGRP version (VERSN) distinguishes between alternate group structures and evaluations of the data. It is then possible to create a different ØNEGRP data set for each different version of the MULTIGRP data set. We need to insert VERSN as a high order qualifier in the names.

```

'ØNEGRP'.VERSN. 'XSECTION'. DISØ. ITDC. REAC
'ØNEGRP'.VERSN. 'SPECTRA'
'ØNEGRP'.VERSN. 'ISOTOPES'

```

The tree associated with these names is illustrated in Figure 6, and the formal record definitions (as they would appear in Volume III) are given in Figures 7-9.

Figure 6. ØNEGRP Tree Structure.

NAME - 'ONEGRP'.?VERSN.'SPECTRA'

LIST - NCSX,(DLAT(N),VLAT(N),BSQD(N),N=1,NCSX)

SYMBOL	DEFINITION	TYPE
VERSN	- NAME OF THIS VERSION OF THE DATA SET	C8
NCSX	- NUMBER OF LATTICE/BUCKLING COMBINATIONS	I4
DLAT	- LATTICE NAME	C8
VLAT	- LATTICE VERSION	C8
BSQD	- BUCKLING VALUE (CM-2)	R4

Figure 7. List of Lattices and Bucklings.

JOSHUA DPSTM-500	Title An Example - Part I	Vol.	Page
		II	A.2.16
		Rev.	Date
			10/70

```

NAME - 'CNEGRP'.?VERSN.'ISOTOPES'
LIST - N1RX,(DISC(N),ITDC(N),REAC(N),N=1,N1RX)

```

SYMBOL	DEFINITION	TYPE
VERSN	- NAME OF THIS VERSION OF THE DATA SET	C8
N1RX	- NUMBER OF ISOTOPE/TEMPERATURE/REACTION COMBINATIONS	I4
DISC	- ISOTOPE NAME	C8
ITDC	- INTEGER TEMPERATURE (C)	I4
REAC	- REACTION TYPE = 'TOTAL', 'CAPTURE', 'FISSION', 'NU', 'TRANSFRT', OR 'N2N'	C8

Figure 8. List of Isotope/Temperatures and Reaction Types in File.

NAME - 'ONEGPP'.?VERSN.'XSECTION'.?DISO.?ITDC.?REAC

LIST - NCSX,(SIG(N),N=1,NCSX)

SYMBOL	DEFINITION	TYPE
VERSN	- NAME OF THIS VERSION OF THE DATA SET	C8
DISO	- ISOTOPE NAME	C8
ITDC	- INTEGER TEMPERATURE (C)	I4
REAC	- REACTION TYPE = 'TOTAL', 'CAPTURE', 'FISSION', 'NU', 'TRANSPRT', OR 'N2N'	C8
NCSX	- NUMBER OF LATTICE/BUCKLING COMBINATIONS	I4
SIG	- ONE-GROUP CROSS SECTION. THE N-TH ENTRY CORRESPONDS TO THE N-TH ENTRY IN THE 'SPECTRA' RECORD	F4

Figure 9. Cross Sections by Isotope(DISO), Temp(ITDC),
Reaction(REAC).

JOSHUA
DPSTM-500

Title Part B - Programming Facilities

Vol.
II

Page
B.O.1

Rev.

Date
10/70

Contents

B.1 - Modular Programming Facilities

B.2 - An Example - Part II

JOSHUA DPSTM-500	Title Part B - Programming Facilities	Vol. II	Page B.O.2
		Rev.	Date 10/76

(Blank)

Work Done By

J. E. Suich
R. S. Wood
A. H. Dobyns
E. A. Crider
J. C. Jensen

Report Written By

J. C. Jensen
R. S. Wood

Contents

	<u>Page</u>
1. Introduction	B.1.3
2. Language Extensions	B.1.5
2.1 Control Statements	B.1.5
2.1.1 MØDULE Statement	B.1.5
2.1.2 STØP Statement	B.1.5
2.1.3 EXECUTE Statement	B.1.6
2.2 Input/Output Statements	B.1.8
2.2.1 General Form of Input/Output Statements	B.1.8
2.2.2 READ Statement	B.1.10
2.2.3 WRITE Statement	B.1.11
2.2.4 DELETE Statement	B.1.12
2.2.5 Shorthand Statements	B.1.13
2.2.6 Error Handling	B.1.14
2.2.7 DEFINE DATASET Statement	B.1.17
2.2.8 DEFAULT DATASET Statement	B.1.17
2.2.9 PRINT Statement	B.1.17
3. Precompiler Facilities	B.1.21
3.1 Reserved Words	B.1.21
3.2 Program Timing	B.1.21
3.3 Error Messages	B.1.22
4. Programming Hints and Restrictions	B.1.26
4.1 Conventional Fortran Input/Output Statements	B.1.26

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.2
		Rev. 1	Date 3/73

(Blank)

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.3
		Rev.	Date 10/70

1. Introduction

JOSHUA computational modules are programmed in an extended version of IBM 360 FORTRAN IV language. The language extensions enable the use of "named data records" and provide the capability of dynamic load module execution. The source modules are translated into standard FORTRAN IV statements by a precompiler and then compiled with an OS/360 FORTRAN compiler. The remaining paragraphs describe the use of the extended statements and other features of the JOSHUA precompiler.

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol.	Page
		II	B.1.4
		Rev.	Date
			10/70

(Blank)

2. Language Extensions

2.1 Control Statements

Three control statements, MØDULE, STØP, and EXECUTE are provided. These statements may not appear as the terminal statement of a DØ loop or in a logical IF statement.

2.1.1 MØDULE STATEMENT

General Form

MØDULE (arg₁,....arg_n)

Where: arg is optional and is equivalent to a FORTRAN subroutine argument.

The MØDULE statement must be the first statement in all main programs. If arguments are supplied the module may be executed only by another module.

Examples:

- (1) MØDULE
- (2) MØDULE (A,B,C)

2.1.2 STØP STATEMENT

Forms

- (1) STOP
- (2) STOP'Hollerith text'
- (3) STOPn
- (4) STOPn'Hollerith text'

Where: n is a positive integer constant or variable name. Hollerith text is any message.

The first or second form of the STØP statement must be used to indicate normal termination of a module. The statement may also appear within any subroutine of the computational module if normal termination does not occur within the main program. The second form will cause the message specified by the Hollerith text to be displayed.

The third and fourth forms of the STØP statement may be used to indicate abnormal termination of a module. They may also appear in any subroutine. A STØP statement with a nonzero integer value also causes an error flag to be set that may be tested

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol.	Page
		II	B.1.6
		Rev.	Date
			10/70

with the EXECUTE statement. The error number, n, is also available to the module issuing the EXECUTE statement. Each of these forms cause a message to be displayed. For Form 3, it is assumed that a message is stored within the system that corresponds to the error number, n.

Examples:

(1) STØP

Normal termination.

(2) STOP'JOB X IS CØMLETE'

Normal termination. The message enclosed in quotes will be displayed with the running count of STØP statements that have been issued during the execution of job step. (Each module in a path may issue a stop of this form.)

(3) STØP 3

Abnormal termination. Action is the same as Example 2 except that the error flag is turned on, and the message is contained within the system.

(4) STOP IX'ERROR OCCURRED IN SUBROUTINE ABC'

Abnormal termination. Action is the same as Example 2 except that the error flag is turned on, and the current value of IX is displayed.

2.1.3 EXECUTE STATEMENT

General Form

EXECUTE name (arg₁,...arg_m) ERR = n

Where: name is the member name of a JOSHUA load module in a partitioned data set.

arg_i is optional and is equivalent to a FORTRAN subroutine argument.

ERR=n is optional and n is the number of the statement to which transfer is made if a STOP indicating abnormal termination occurred during the execution of the named module.

The EXECUTE statement is used to initiate the execution of another JOSHUA load module.

Transfer is made to the statement specified by the ERR parameter if a STOP with a nonzero value occurred in the module specified in the EXECUTE statement. If the ERR parameter is omitted, program execution is terminated if a nonzero STOP occurred during execution of the named module.

Examples:

- (1) EXECUTE PRØGRAM1
- (2) EXECUTE PRØGRAM2 ERR=42
- (3) EXECUTE PRØGRAM3 (A,B,C) ERR=42

If an abnormal STOP is issued in PRØGRAM1 execution of the module will be terminated. In Example 2 an abnormal STOP in PRØGRAM2 would cause transfer of control to statement number 42 where the error number can be tested if desired. The error number must be referred to as \$ERRNØ*. Example 3 is the same as Example 2 except the arguments A, B, and C have been defined as variables within the module that issues the EXECUTE statement.

Alternate Form

EXEC name (arg₁,...arg_m) ERR=n

This statement is the same as EXECUTE except name is a FORTRAN variable.

Example:

```
REAL*8 PRØG/'PRØGRAM1'/
EXEC PRØG
```

The result of this statement is the same as Example (1) above.

* See Section 3.1, Reserved Words

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.8
		Rev. 1	Date 3/73

2.2 Input/Output Statements

Since JOSHUA modules communicate primarily with named data records, three input/output statements have been provided, READ, WRITE, and DELETE. These statements are programmed precisely as FORTRAN unformatted READ/WRITE statements with the data record name replacing the data set reference number. Restrictions on the associated list are discussed below. In addition to the input/output statements a method of defining a shorthand form of a data record name is provided. Input/Output statements may not appear as the terminal statement of a DO loop or in a logical IF statement.

2.2.1 General FORM of Input/Output Statements

READ

WRITE (Name, ERR=m) list

DELETE

PRINT

Name is a name of a JOSHUA data record. The name may be written as a string of qualifiers separated by periods or as the beginning and ending locations of a vector separated by a colon that contains the Hollerith form of the name qualifiers. The number of qualifiers in a name must be between 2 and 12 inclusive except for DELETE which may have one.

If the name is written as a string of qualifiers they may be of the following form:

- (1) a literal of up to 8 characters (e.g., 'LATTICE')
- (2) an integer variable ≥ 0
- (3) a non-integer variable that contains the Hollerith form of the qualifier
- (4) an integer constant ≥ 0
- (5) a pseudo-variable previously defined in a shorthand statement*.

* See 2.2.5, Shorthand Statements

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.9
		Rev. 1	Date 3/73

Restrictions:

- (1) The first qualifier of a name must not begin with a numeric character.
- (2) No qualifier may begin with a question mark.
- (3) No qualifier may contain imbedded periods.

Examples:

Specification Statements

```

REAL*8 A(10)
REAL*4 TYPE/'HOT'/,B(16)
'LATTICE'.'GEOM'.TYPE→SHORT*

```

Names:

- (1) 'LATTICE'.'GEOM'.TYPE.NCASE.1
- (2) A(1):A(5)
- (3) B(I+J/K):B(5+I+J/K)
- (4) SHORT.NCASE.1

In Case 1 the first two qualifiers of the name, LATTICE and GEOM, are literal strings. The qualifier TYPE is a Real*4 variable that has been initialized with the literal string, 'HOT'. NCASE is an Integer*4 variable by default. At execution time the current value of NCASE will be converted to a literal string and left-adjusted. The final qualifier, 1, is an integer constant and will also be converted to a literal and left-adjusted.

In Cases 2 and 3, the colon form of a name is used. The arrays A and B are assumed to contain the literal or Hollerith values of the name qualifiers between the limits specified by the indices. Note that any expression may be used as an index.

In Case 4 a shorthand form of a name is used. Since the qualifier SHORT appears in a shorthand specification statement referring to 'LATTICE'.'GEOM'.TYPE, the

* See 2.2.5, Shorthand Statements

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.10
		Rev. 1	Date 3/7

name in Case 4 is identical to Case 1.

ERR=m is optional and m is the number of the statement to which transfer is made if the data management routine, BNAM, cannot locate or catalog the requested record name. The number of the error that occurred may be tested by referring to \$ERRNØ*.

List is a FORTRAN I/Ø list. The I/Ø is restricted somewhat for ease of handling by the JOSHUA system routines. Indexing must be explicit, for all WRITE statements, and all variable names used as index limits must appear before their use as an index limit. In any given nest of implied DØ loops, all variable names must appear to the left of the indexing parameter of the innermost DØ loop. No more than one implied DØ loop in any nest may have triangular indexing (i.e., refer to an outer loop index as an indexing parameter). Character strings must appear in the list as LØGICAL*1 arrays. Note that a list is not used in a DELETE statement.

2.2.2 READ Statement

The READ statement is used to transfer data from JOSHUA named data records to the storage locations specified in the FORTRAN I/O list. For each READ statement a call is generated to the data management routine, BNAM, in order to locate the position of the data. If the data record cannot be located, control will be transferred to the statement number specified in the ERR option. If the ERR option is omitted, execution will be terminated and an error message displayed.

Example:

In this example the variable TYPE is assumed to contain the Hollerith characters, 6B, that were either previously read in as data or defined in a data initialization statement. The variable NCASE is an integer whose current value is 3.

```
READ('LATTICE'.'GEØM'.TYPE.NCASE,ERR=23)N,(A(I),I=1,N)
```

This statement causes a search of the JOSHUA data base for a record called LATTICE.GEØM.6B.3. If the record name is located, data will be read into the locations

* See Section 3.1, Reserved Words and 2.2.6, Input/Output Errors.

specified in the list.

If the Hollerith form of the name qualifiers were stored in a REAL*8 array named B, the above statement could be written as:

```
READ(B(1):B(4))N,(A(I),I=1,N)
```

This statement is identical to the first example except that the ERR option is not coded. In this case, if the record name does not exist, execution of the module will be terminated and an error message displayed.

2.2.3 WRITE Statement

The WRITE statement is used to create a new data record name and to transfer the data specified in the I/O list to the JOSHUA data base. For each WRITE statement a call is generated to the data management routine, BNAM, in order to enter the name qualifiers in the tree catalog and establish a position in the data base to store the data. If the record name already existed, the data will be stored in the same position if the list length is less than or equal to the previous list length. If the list length is longer than the previous length, a new position will be established for data storage and the tree catalog will be updated. Control will be transferred to the statement number specified in the ERR option if an improper name was coded or if there was insufficient space available to catalog or store the data. If the ERR option is omitted, execution will be terminated and an error message displayed.

Improper names result from the following conditions:

- (1) The name given referred to an existing subtree: the name was too short.
- (2) The name given contained an initial string of qualifiers that had previously been defined as a data record: the name was too long.

Example

```
WRITE('ASSEMBLY','MATERIAL','5R',ERR=50)  
X NTUBE,(MAT(I,J),J=1,10),I=1,NTUBE)
```

This statement causes the data manager to catalog the name ASSEMBLY.MATERIAL.5R in the JOSHUA data base. The space allocated for data storage is

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.12
		Rev. 1	Date 3/77

determined from the types of the variables in the list and the values of the indexing limits. The data is then transferred to the data base from the core storage locations specified in the list.

If the record name cannot be cataloged, control will be transferred to statement number 50. Execution of the module will be terminated under these conditions if the ERR option is not coded. If we assume that a record by the name of ASSEMBLY.MATERIAL.5R already exists, then the following names may not be cataloged:

- (1) ASSEMBLY.MATERIAL (too short)
- (2) ASSEMBLY.MATERIAL.5R.ANYNAME (too long)

2.2.4 DELETE Statement

The DELETE statement is used to remove named data records from the JOSHUA data base. Records may be deleted individually or by groups, (delete a subtree). Groups of records are deleted by coding a partial name in the DELETE statement. For each DELETE statement a call is generated to the data management routine, BNAM, to erase the terminal qualifier of the coded name from the tree catalog. The erase operation is done by overlaying the terminal qualifier in the catalog with non-Hollerith characters. The space used for storing both the name and data is not recovered. If the coded name cannot be located, control will be transferred to the statement number specified in the ERR option. If the ERR option is omitted execution of the module will be terminated and a message displayed. Note that the record name must exist in the Job or Scratch data set for the DELETE operation to be successful.

Examples:

- (1) DELETE('LATTICE'.'GEOM'.'6B'.3,ERR=20)

This statement causes the data manager to overlay the terminal qualifier 3 in the tree catalog with non-Hollerith characters. Control will be transferred to statement number 20 if the record cannot be located in the Job data set. Since data of the same name may exist in either the User or Standard data sets, it is possible to retrieve a data record after a successful DELETE operation.

(2) DELETE('LATTICE','GEØM',ERR=20)

This statement causes the data manager to overlay the terminal qualifier GEØM in the tree catalog with non-Hollerith characters. In this case, the complete subtree headed by the qualifier, GEØM, would be deleted.

(3) DELETE('SCRATCH',ERR=20)

This DELETE statement causes the entire SCRATCH data set to be deleted. Since the SCRATCH data set is handled in a special manner all space previously used in both the tree catalog and data set will be recovered. Note that an error will occur if the data set has no entries when the DELETE statement is issued.

2.2.5 Shorthand Statements

Shorthand statements may be used to define a name or a portion of a name that may be used in conjunction with an input/output statement. The form is as follows:

name → V, ERR=m

where: name is defined in Section B.2.2.1

ERR=m is defined in Section B.2.2.1

→ is an arrow composed of a minus sign and a "greater than" sign.

V is a pseudo variable that represents the information on the left side of the arrow.

The variable, V, does not assume a value and therefore may not appear in COMMON or in an argument list. The ERR option will be effective only if it is not specified in the I/O statement that contains the pseudo variable. All pseudo variables must be defined before they are used in any I/O statement. They may be used in the definition of other pseudo variables and they may be concatenated.

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.14
		Rev. 1	Date 3/

Examples:

(1) 'LATTICE'. 'GEOM'. TYPE→SNAM

The following READ statements produce the identical code.

```

READ(SNAM.NCASE,ERR=20)N,(A(I),I=1,N)
READ('LATTICE'. 'GEOM'. TYPE.NCASE,ERR=20)
  N,(A(I),I=1,N)

```

(2) 'GEOM'. TYPE→SHORT, ERR=50

The following WRITE statements produce the same code.

```

WRITE('LATTICE'.SHORT.NCASE)N,(A(I),I=1,N)
WRITE('LATTICE'. 'GEOM'. TYPE.NCASE,ERR=50)
  N,(A(I),I=1,N)

```

2.2.6 Error Handling

When the data management routine, BNAM, is unable to satisfy a request, it sets a number in the reserved word \$ERRNØ* that indicates the type of problem it encountered. If the ERR option is not used in the I/Ø statement, execution of the module will be terminated, (user code 000017), and an appropriate message issued. Table 1 contains a list of the possible error numbers and their explanations.

* See Section 3.1, Reserved Words

Table 1

BNAM ERRORS

<u>Error Number</u>	<u>Explanation</u>
2	BNAM received a name with only one qualifier from a read or write request. The name given for a read or write request must contain at least two qualifiers.
3	BNAM received a write request. The space allocated for the data set tree catalog was full.
4	BNAM received a write request. The space allocated for the data set was full.
6	An attempt was made to use one of the constants CATALOG, WORK, or FREE as a data set name; to use \$\$\$ as a qualifier.
7	BNAM received a read, write, or delete request. The data set name could not be located. Note that the data set name could exist in a User or Job data set for a read request. In this case, if the record name did not exist in the Job or User data set, and the data set name was not a Standard data set, this error would occur.
8	BNAM received a read or delete request. The first name qualifier was in the name catalog but had no corresponding data entries. The data set has probably been deleted.
9	BNAM received a read or a delete request with a name of more than one qualifier. The initial name qualifier was located, but one or more of the remaining qualifiers could not be located.
11	An attempt has been made to retrieve unauthorized data.

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.16
		Rev. 2	Date 9/73

Table 1 (Cont.)

BNAM ERRORS

<u>Error Number</u>	<u>Explanation</u>
12	BNAM received a read, a write, or a delete request. A name already existed containing fewer qualifiers than the given name. Since the terminal qualifier of a name cannot point to both data and another qualifier, this is an illegal operation (e.g., Record A.B.C. exists. A request with a name of A.B.C.D. is illegal.)
13	BNAM received a read or write request. A name already existed containing more qualifiers than the given name. Since the terminal qualifier of a name cannot point to both data and another qualifier this is an illegal operation (e.g., Record A.B.C.D. exists. A request with a name of A.B.C. is illegal.)
14	An invalid functional request number has been received by BNAM. See a member of the Systems Development Group.
15	The number of bytes requested on a write exceeds 262136.
21	BNAM received a write request. The length of the I/O list that was passed as an argument was less than zero.
22	The number of name qualifiers passed to BNAM was out of range.
24	An attempt has been made to create data without proper authorization.
26	The catalog in the terminal region is different from the catalog in the batch region for a given data set. This usually is a result of a batch update occurring while a catalog entry is held in core in the terminal region.

Table 1 (Cont.)BNAM ERRORS

<u>Error Number</u>	<u>Explanation</u>
27	Illegal rename request. "?" cannot be used as the rename qualifier.
29	The requested protection category does not exist in the specified data set. This would result if a job were run using an old Job data set with a different protection category.
30	BNAM received a write request from a monitor function, e.g., COPY. The specified record name existed in both data sets, each with a different protection category.
33	BNAM received a write request directed to a User data set. Either a Set Write function had not been done for a batch monitor run, or the user was not on the access list for the specified User data set when signed on a terminal.
36	BNAM received a rename request. The new name qualifier already existed in the tree catalog.
37	System parameters for list options are out of range. See a member of the Systems Development Group.
38	During the execution of the DDISPLAY module (this results from the PRINT statement in batch module), a protected template did not agree with the category of the data.

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.18
		Rev. 1	Date 3/73

(Blank)

2.2.7 DEFINE DATASET Statement

General Form

DEFINE DATASET (dsn₁,...dsn_m)

where: dsn is a JOSHUA data set name

m is $> 1 \leq 4$

This statement is used to change the normal, or default, data set search sequence, or hierarchy, used by the data manager, BNAM, when retrieving named data records. Both User and Job data set names are permissible. The literal, 'STD', is used to include Standard data sets in the sequence.

Example:

```
REAL*8 VAR/'0777'/  
DEFINE DATASET (0666,VAR,'STD')
```

This statement causes the default search sequence to be overridden with the sequence 0666, 0777, Standard. If the module containing the statement is part of a path, the default sequence will be restored when the module terminates.

2.2.8 DEFAULT DATASET Statement

Form

DEFAULT DATASET

This statement has no arguments and is used to re-establish the default search sequence within a module subsequent to a DEFINE DATASET statement.

2.2.9 PRINT Statement

The PRINT statement is used to obtain a formatted printout of a data record. The format of the printout is contained in a "template" for the specified record type. (Templates are described in detail in Part E of this Volume.) If a record or template cannot be located, control will be transferred to the statement number specified in the ERR option. If the ERR option is omitted, execution will be terminated and an error message displayed.

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.20
		Rev. 1	Date 3/7

The print operation is done by dynamically linking to the JMS Data Display routine for each record name requested. Therefore, an additional 16K bytes of core storage is required when this feature is used.

Example:

```
PRINT('LATTICE'. 'GEOM'. TYPE. NCASE, ERR=60)
```

The record name used in this example is identical to that used in the READ example. If the record name, LATTICE.GEOM.6B.3, or the template, LATTICE.GEOM.?.?, cannot be located, control will be transferred to statement number 60. If the operation is successful, the data will be printed in ten-line increments identical to the "pages" that are displayed with JMS.

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.21
		Rev. 1	Date 3/73

3. Precompiler Facilities

3.1 Reserved Words

In order for a module to communicate with the various systems routines, the precompiler generates a number of mnemonics for subroutine and variable names. These mnemonics all have the initial character \$, and are referred to as reserved words. To avoid conflict with code generated by the precompiler, the user should not begin any variable or subroutine with a \$.

Four of the generated mnemonics may be of use to the programmer. These values should be used on a fetch-only basis, i.e., X = \$JØB, and must not be changed.

- 1) \$JØB - Real*8 variable that contains the EBCDIC representation of the job name.
- 2) \$BADGE - Real*8 variable that contains the EBCDIC representation of the users data set name.
- 3) \$SYS - Integer*4 variable that contains the data set reference number of the system's output device.
- 4) \$ERRNØ - Integer*4 variable that contains the error number that was returned from the data management routine or that was used in a STØP statement.

3.2 Program Timing

Execution time for each subroutine of a JOSHUA module may be obtained through the use of a PRØGRAM TIMINGS statement.

General Form:

PRØGRAM TIMINGS n, ØVLY

where: n is an integer greater than or equal to the number of subroutines in the source deck.

ØVLY is specified only if the resulting module is placed in an overlay structure.

One PRØGRAM TIMINGS statement may be included at any place in a source deck. Statements will be generated by the precompiler for each subroutine following the PRØGRAM TIMINGS statement that accumulate the CPU time for each subroutine and the number of times a subroutine was entered. These statistics may be

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol.	Page
		II	B.1.22
		Rev.	Date
		1	3/73

printed at the end of a job by the use of the TIMER parameter on the execute card. If the module terminates with other than a STOP statement the statistics cannot be obtained.

The subroutines used by the PROGRAM TIMINGS function may also be called directly by the FORTRAN program. The following routines are available:

- o \$\$1 - turns on the program timings function;
- o \$\$2 - turns off the program timings function; Accumulated statistics previous to a CALL \$\$2 statement will be printed at job termination.

Note: If both a CALL \$\$1 and a CALL \$\$2 are made they must be made from the same subroutine.

- o \$\$5 - prints accumulated statistics.

3.3 Error Messages

The following error messages, when issued by the Precompiler, will be generated as comments cards and included with the subprogram.

JOSHUA PRECOMPILER ERROR → 1 ←***
CALL CORE IS ILLEGAL

The subroutine CORE (see DPSTOM-55) cannot be used in any JOSHUA module.

JOSHUA PRECOMPILER ERROR → 2 ←***
NAMELIST TYPE I/O IS NOT ALLOWED

Due to a current restriction in the JOSHUA operating system, all NAMELIST definitions are flagged as errors.

JOSHUA PRECOMPILER ERROR → 3 ←***
CARD HAS INVALID JOSHUA KEYWORD

The source card did not have a recognizable keyword.

JOSHUA PRECOMPILER ERROR → 4 ← ***
INCORRECT FORTRAN TYPE STATEMENT

The syntax on an explicit FORTRAN type statement is incorrect, e.g., REAL*3.

JOSHUA PRECOMPILER ERROR → 5 ← ***
MODULE CARD MUST BE FIRST CARD

The MODULE statement must appear as the first card of a subprogram, exclusive of any comments cards.

JOSHUA PRECOMPILER ERROR → 6 ← ***
ONLY ONE MODULE CARD PER COMPILE

The MODULE statement can only be used once per precompile.

JOSHUA PRECOMPILER ERROR → 7 ← ***
NAME HAS TOO MANY QUALIFIERS

JOSHUA record names are limited to 12 qualifiers.

JOSHUA PRECOMPILER ERROR → 9 ← ***
TABLE OVERFLOW-NEWTAB SUBROUTINE

The program is too large to precompile.

JOSHUA PRECOMPILER ERROR → 12 ← ***
UNBALANCED PARENTHESES

An I/O list in a READ/WRITE statement has an incorrect pairing of parentheses.

JOSHUA PRECOMPILER ERROR → 14 ← ***
DIRECT ACCESS READ/WRITE ILLEGAL

No direct access I/O statements are allowed.

JOSHUA PRECOMPILER ERROR → 15 ← ***
ERROR IN PROGRAM TIMINGS CARD

In a PROGRAM TIMINGS statement, either the first parameter was not an integer constant or the second parameter was not ",OVLY".

JOSHUA PRECOMPILER ERROR → 16 ← ***
MODULE NAME IS INVALID

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.24
		Rev. 1	Date 3/73

The program name on an EXECUTE card is not a FORTRAN mnemonic.

JOSHUA PRECOMPILER ERROR → 17 ← ***
KEYWORD NOT ERR OR BAD LABEL

An error was found in the ERR= option in a record name.

JOSHUA PRECOMPILER ERROR → 18 ← ***
DUPLICATE ERR KEYWORD

Duplicate ERR= conditions were found.

JOSHUA PRECOMPILER ERROR → 20 ← ***
NAMESTRING NOT FOUND

No name was found in an I/O statement, i.e., READ, WRITE or DELETE.

JOSHUA PRECOMPILER ERROR → 21 ← ***
NULL LITERAL FIELD FOUND

No characters were found between a pair of matching quote marks in a record name.

JOSHUA PRECOMPILER ERROR → 22 ← ***
DELIMITER NOT A PERIOD

The qualifiers in a record name were not separated by a period.

JOSHUA PRECOMPILER ERROR → 23 ← ***
INVALID VARIABLE IN NAME

A special character was found in a record name qualifier.

JOSHUA PRECOMPILER ERROR → 24 ← ***
NULL PAREN. FIELD FOUND

A matching set of parentheses in a name contained only blanks.

JOSHUA PRECOMPILER ERROR → 25 ← ***
COLON IN WRONG POSITION IN NAME

A colon did not separate the two subscripted variables in a colon record name.

JOSHUA

DPSTM-500

Title Modular Programming Facilities

Vol.

II

Page

B.1.25

Rev.

1

Date

3/73

JOSHUA PRECOMPILER ERROR → 26 ← ***
INVALID ERR OPTION IN NAME

The ERR= option in a colon record name had invalid syntax.

JOSHUA PRECOMPILER ERROR → 27 ← ***
MISSING LABEL IN ERR OPTION

No label was found in an ERR= option of a colon record name.

JOSHUA DPSTM-500	Title Modular Programming Facilities	Vol. II	Page B.1.26
		Rev. 1	Date 3/73

4. Programming Hints and Restrictions

4.1 Conventional FORTRAN Input/Output Statements

Although the use of conventional input/output statements is discouraged, it is recognized that some applications will require these means for a practical solution. The following items should be considered before using these means.

1) System Output

The system output device, normally a printer, may be used in any module. Since the JOSHUA system is normally accessed from a terminal, system output may be of little value to the remote terminal user.

2) System Input

The system input device, normally a card reader, may be used for special applications, e.g., building cross-section libraries from raw data. Part C of this volume describes the use of module input data cards.

3) Sequential Data Sets

Any sequential data set residing on tape or direct access storage may be used for special applications. All data sets will be closed, (rewound), at the termination of each module execution within a path.

4) Direct Access Data Sets

Data sets outside of the JOSHUA data base with a direct access organization may not be used.

JOSHUA

DPSTM-500

Title An Example - Part II

Vol. II. Page
B.2.1

Rev. Date
10/70

Work Done By

F. D. Knight
H. C. Honeck
R. S. Wood

Report Written By

F. D. Knight
H. C. Honeck
R. S. Wood

Contents

	<u>Page</u>
1. The CØG Module.	B.2.3
2. Precompiler Output	B.2.13

JOSHUA DPSTM-500	Title An Example - Part II	Vol. II	Page B.2.2
		Rev.	Date 10/70

(Blank)

1. The CØG Module

There are two things the user must specify to create a ØNEGRP data set:

1. The version of MULTIGRP to be used (VERSN).
2. The list of lattices to be used (DLAT (NL), VLAT (NL), NL=1,NLX)

The following input record is defined to supply these data to CØG.

Name: 'INPUT'.'CØG'.\$JØB

List: VERSN,NLX,(DLAT(NL),VLAT(NL),NL=1,NLX)

The qualifier \$JØB is added to permit several cases to be set up and executed. The details of how this is accomplished will be described later.

Having fully defined the input and output data sets, the logic for the CØG module is straightforward and is summarized by Figure 1.

If CØG were to be a production module on the JOSHUA system, we would have to include two more features.

1. Updating an existing ØNEGRP data set (addition, deletion, correction of isotopes, lattices, etc.).
2. Additional checking to verify the consistency of the energy group structures.

Since the inclusion of these features would not illustrate any new facilities of the JOSHUA system, they are not included.

Figure 1. CØG Flow Diagram

JOSHUA

DPSTM-500

Title An Example - Part II

Vol.
II

Page
B.2.5

Rev.

Date
10/70

The source deck for CØG is shown in Figure 2 with the comments listed below keyed to the circled numbers in the figure.

```

① MODULE
C=====COG, CREATE ONEGROUP DATA SET=====
  REAL*8 DISOI(100),DISO(600),REAC(600),DLATI(50),VLATI(50),
  *      DLAT(100),VLAT(100),VERSN
  REAL*4 BSQD(100),F(4,100),FL(100,100),TF(100),SIG(100),SIGP(100),
  *      S(7,100)
  INTEGER*4 ITDCI(50),ITDC(600)
  INTEGER*4 NEGX/100/,ISOXX/100/,NCSXX/100/,NIRXX/600/,NLXX/50/
  REAL*8 RNAME(6)/'TOTAL','CAPTURE','FISSION','NU','TRANSPRT','N2N'/
C=====SHORTHAND
  'LATTICE'.DLATI(NL).VLATI(NL) -> L
  'MULTIGRP'.VERSN -> M
  'ONEGRP'.VERSN -> O
C=====READ INPUT DATA
  READ('INPUT','COG',5JOB,ERR=906)
  *      VERSN,NLX,(DLATI(N),VLATI(N),N=1,NLX)
  IF(NLX.GT.NLXX) GO TO 901
C=====READ ISOTOPE LIST
  READ(M,'ISOTOPE',ERR=907) NEG,ISOX,(DISOI(N),ITDCI(N),N=1,ISOX)
  IF(ISOX.GT.ISOXX) GO TO 902
  IF(NEG.GT.NEGX) GO TO 905
C=====READ SPECTRUM RECORDS
  100 NCS=0
  DO 130 NL=1,NLX
    ④ → READ(L,'KEFF',ERR=908) NSOX,(Z,Z,BSQD(NCS+NSO),Z,Z,NSO=1,NSOX) ← ⑤
    IF(NCS+NSOX.GT.NCSXX) GO TO 903
    DO 120 NSO=1,NSCX
      ④ → READ(L,'LFLUX'.NSO,ERR=908) LPX,NGP,(Z,(F(LP,I),LP=1,LPX),I=1,NGP)
      IF(NGP.NE.NEG) GO TO 910
      NCS=NC S+1
  
```

Figure 2. Listing of the COG Module.

- ① Must be the first card in deck.
- ② Many of these could be equivalenced to conserve space.
- ③ Shorthand statements may appear anywhere prior to their use. The symbols L, M, Ø can also be used in the program as variable names.
- ④ The character string denoted by L is inserted by the precompiler whenever L is referred to in a name. Hence the variable NL must have a value each time L is used.
- ⑤ Z is a dummy variable. We only want to pick the values of BSQD out of this record.

```

      TF(NCS)=0.0
      DO 110 I=1,NEG
      FL(I,NCS)=F(1,I) ← ⑥
110   TF(NCS)=TF(NCS)+F(1,I)
      DLAT(NCS)=DLATI(NL)
120   VLAT(NCS)=VLATI(NL)
130   CONTINUE
      NCSX=NCS
      WRITE(0,'SPECTRA',ERR=909) NCSX,(DLAT(N),VLAT(N),BSQD(N),N=1,NCSX)
C=====LOOP OVER ISOTOPES AND READ CROSS SECTIONS
      NIR=0
      DO 170 ISO=1,ISOX
      READ(M,DISOI(ISO).ITDCI(ISO),'XSECTION',ERR=907)
      *   NEG,((S(K,I),K=1,7),I=1,NEG) ← ⑦
C-----LOOP OVER ALL REACTION TYPES EXCEPT CHI (TYPE 7)
      DO 160 NR=1,6
C       LOOP OVER ALL SPECTRA
      DO 150 NCS=1,NCSX
      SIG(NCS)=0.0
      DO 140 I=1,NEG
      IF(NR.NE.4) SIG(NCS)=SIG(NCS)+FL(I,NCS)*S(NR,I)
      IF(NR.EQ.4) SIG(NCS)=SIG(NCS)+FL(I,NCS)*S(4,I)*S(3,I) } ⑧
140   CONTINUE
      IF(SIG(NCS).EQ.0.0) GO TO 160 ← ⑨
      IF(NR.EQ.3) SIGP(NCS)=SIG(NCS)
      IF(NR.EQ.4) SIG(NCS)=SIG(NCS)/SIGP(NCS) } ⑩
      IF(NR.NE.4) SIG(NCS)=SIG(NCS)/TF(NCS)
150   CONTINUE

```

Figure 2. Listing of the CØG Module (Continued)

- ⑥ Only the isotropic flux (LP=1) is used.
- ⑦ Note redefinition of list defined in Figure 1, Section A.2. This is done for ease of indexing.
- ⑧ Special averaging for v .
- ⑨ If the value is zero for the first case, it will be zero for all cases, and the record can be omitted.

```

NIR=NIR+1
IF(NIR.GT.NIRXX) GO TO 904
DISC(NIR)=DISC(ISO)
ITDC(NIR)=ITDC(ISO)
REAC(NIR)=RNAME(NR)
WRITE(C,'XSECTION'.DISC(NIR).ITDC(NIR).REAC(NIR),ERR=909)
*      NCSX,(SIG(N),N=1,NCSX)
C-----END REACTION AND ISOTOPE LOOPS
160 CONTINUE
170 CONTINUE
NIRX=NIR
C=====WRITE ISOTOPES RECORD
WRITE(C,'ISOTOPES',ERR=909)NIRX,(DISC(N),ITDC(N),REAC(N),N=1,NIRX)
C=====FINISHED
STOP
901 STOP 1 'COG, TOO MANY LATTICE'
902 STOP 2 'COG, TOO MANY ISOTOPES'
903 STOP 3 'COG, TOO MANY LATTICE*BUCKLINGS'
904 STOP 4 'COG, TOO MANY ISOTOPES*REACTIONS'
905 STOP 5 'COG, TOO MANY GROUPS'
906 STOP 6 'COG, INPUT RECORD MISSING'
907 STOP 7 'COG, MULTIGRP RECORD MISSING'
908 STOP 8 'COG, LATTICE RECORD MISSING'
909 STOP 9 'COG, WRITING ERROR'
910 STOP10 'COG, INCONSISTANT GROUP STRUCTURE'
END

```

Handwritten annotations: A circled '10' with an arrow pointing to the 'STOP' line. A circled '12' with an arrow pointing to the 'END' line. A bracket on the right side of lines 901-910 with a circled '11' next to it.

Figure 2. Listing of the COG Module (Continued)

10

Normal stop.

11

Error stops.

12

Blanks ignored.

JOSHUA DPSTM-500	Title An Example - Part II	Vol. II	Page B.2.12
		Rev.	Date 10/70

(Blank)

2. Precompiler Output

Once the module source program has been written in the extensions of FORTRAN IV offered by the JOSHUA system, the program must be precompiled by the JOSHUA precompiler. On the following pages is shown the precompiled listing of the CØG module with descriptions of all JOSHUA generated statements. In the listing, all user statements containing JOSHUA syntax appear as comment cards and are followed by the corresponding JOSHUA generated statements. The JOSHUA generated statements are indented for ease of distinction.

```

ISN 0002 ① { CJ*** MODULE
                SUBROUTINE MAIN($MCB )
ISN 0003      C=====COG, CREATE ONEGROUP DATA SET=====
ISN 0004      EXTERNAL $CDINT,$RAJ,$FLG06
ISN 0005      COMMON /$JCOMA/ $JCB(16),$NAMES(18),$BNAMI
ISN 0006      REAL*8 $NAMES,$JCB,$CDINT,$RAJ,$LNAM$(1),$JOB,$BADGE
ISN 0007      INTEGER*4 $BNAMI(8),$DSNAM,$RECRD,$JCB(32),$SYS,$ERRND
                EQUIVALENCE ($JCB(1),$JCB(1)),$BNAMI(2),$DSNAM),
                * ($BNAMI(3),$RECRD),($JCB(5),$SYS),($BADGE,$JCB(2)),
                * ($ERRND,$JCB(8)),$JOB,$JCB(1))
ISN 0008      LOGICAL*1 $LGL$(1)/F/,$$$,$FLG06*4
ISN 0009      REAL*8 $MCB(18)
ISN 0010      CALL $YUDEX('JIBINIT',$MCB,$JCB)
ISN 0011      CALL $YUDFL($JCB)
ISN 0012      REAL*8 DISOI(100),DISO(600),REAC(600),DLATI(50),VLATI(50),
                * DLAT(100),VLAT(100),VERSN
ISN 0013      REAL*4 BSQD(100),F(4,100),FL(100,100),TF(100),SIG(100),SIGP(100),
                * S(7,100)
ISN 0014      INTEGER*4 ITDCI(50),ITCC(600)
ISN 0015      INTEGER*4 NEGX/100/,ISOXX/100/,NCSXX/100/,NIRXX/600/,NLXX/50/
ISN 0016      REAL*8 RNAME(6)/'TOTAL','CAPTURE','FISSION','NU','TRANSPRT','N2N'/
                C=====SHORT HAND
                CJ*** 'LATTICE'.DLATI(NL).VLATI(NL) -> L
                CJ*** 'MULTIGRP'.VERSN -> M
                CJ*** 'ONEGRP'.VERSN -> O
                C=====READ INPUT DATA
                CJ*** READ('INPUT'.'.COG'.'. $JOB,ERR=906)
                CJ *   VERSN,NLX,(DLATI(N),VLATI(N),N=1,NLX)
ISN 0017      $NAMES(3)=$JCB
ISN 0018      CALL $YUD2($LGL$(2),$LNAM$,1,$906)
ISN 0019      READ($DSNAM,$RECRD)VERSN,NLX,(DLATI(N),VLATI(N),N=1,NLX)
                *
                *

```

Figure 3. Listing of the Precompiler Output

1

The MODULE keyword is replaced by a SUBROUTINE statement. The argument, \$MCB, contains control information such as the job name, \$JOB. Note that all generated mnemonics begin with a dollar sign, \$.

2

This set of FORTRAN specification statements is standard generation for all precompiled programs. These statements identify commonly used JOSHUA variables and functions.

3

These three statements appear only in JOSHUA main programs as defined by the MODULE statement.

4

L, M, and Ø are pseudo variables which are used as shorthand notation in data record names.

5

All I/O keywords (READ, WRITE, DELETE) generate a call to the \$YUD routines. The particular \$YUD routine is determined by the I/O keyword. Here the READ keyword generates a CALL \$YUD2. \$NAMES(1) and \$NAMES(2) will be set to 'INPUT' and 'CØG', respectively while \$NAMES(3) is set inline to the contents of \$JOB. All record name qualifiers which are FORTRAN variables, i.e., \$JOB, are loaded inline, while all invariant names, i.e., 'INPUT', are loaded by the \$YUD routines. The \$YUD routine then calls the data manager to determine the location of the record. If the record cannot be located, control will return to statement 906. The last statement generated is a FORTRAN direct access I/O statement to access the data.

```

ISN 0020          IF(NLX.GT.NLXX) GO TO 901
                  C=====READ ISOTOPE LIST
(6)→ CJ*** READ(M.'ISOTOPES',ERR=907) NEG,ISOX,(DISOI(N),ITDCI(N),N=1,ISOX)
ISN 0022          $NAMES(2)=VERSN
ISN 0023          CALL $YUD2 ($LGL$(6),$LNAM$,1,&907)
ISN 0024          READ($DSNAM'$RECRD)NEG,ISOX,(DISOI(N),ITDCI(N),N=1,ISOX)
                  *
                  *
ISN 0025          IF(ISOX.GT.ISOXX) GO TO 902
ISN 0027          IF(NEG.GT.NEGX) GO TO 905
                  C=====READ SPECTRUM RECORDS
ISN 0029          100  NCS=0
ISN 0030          DO 130 NL=1,NLX
(8)→ CJ*** READ(L.'KEFF',ERR=908) NSOX,(Z,Z,BSQD(NCS+NSO),Z,Z,NSO=1,NSOX)
ISN 0031          $NAMES(2)=DLATI(NL)
ISN 0032          $NAMES(3)=VLATI(NL)
ISN 0033          CALL $YUD2 ($LGL$(11),$LNAM$,1,&908)
ISN 0034          READ($DSNAM'$RECRD)NSOX,(Z,Z,BSQD(NCS+NSO),Z,Z,NSO=1,NSO
                  *
                  *
ISN 0035          IF(NCS+NSOX.GT.NCSXX) GO TO 903
ISN 0037          DO 120 NSO=1,NSOX
                  CJ*** READ(L.'LFLUX'.NSC,ERR=908) LPX,NGP,(Z,(F(LP,I),LP=1,LPX),I=1,NGP)
ISN 0038          $NAMES(2)=DLATI(NL)
ISN 0039          $NAMES(3)=VLATI(NL)
ISN 0040          $NAMES(5)=$RAJ(4,NSO)
ISN 0041          CALL $YUD2 ($LGL$(17),$LNAM$,1,&908)
ISN 0042          READ($DSNAM'$RECRD)LPX,NGP,(Z,(F(LP,I),LP=1,LPX),I=1,NGP
                  *
                  *
ISN 0043          IF(NGP.NE.NEG) GO TO 910
ISN 0045          NCS=NCS+1
ISN 0046          TF(NCS)=0.0
ISN 0047          DO 110 I=1,NEG
ISN 0048          FL(I,NCS)=F(1,I)
ISN 0049          110  TF(NCS)=TF(NCS)+F(1,I)
ISN 0050          DLAT(NCS)=DLATI(NL)
ISN 0051          120  VLAT(NCS)=VLATI(NL)
ISN 0052          130  CONTINUE

```

Figure 3. Listing of the Precompiler Output (Continued)

6

Note the use of the pseudo variable M in the data record name. This statement will access the record named 'MULTIGRP'.VERSN.'ISOTOPES'.

7

I/O keyword generation.

8

Note the use of the pseudo variable L in the data record name. This statement will access the record named 'LATTICE'.DLATI(NL).VLATI(NL).'KEFF'.

9

I/O keyword generation.

10

I/O keyword generation. The \$RAJ routine is called in this case to convert the integer NSØ to a left adjusted EBCDIC value of length 8. Any variable qualifier that is not declared as REAL*8 will be handled by the \$RAJ routine.

```

ISN CC53      NCSX=NCS
(11) → CJ*** WRITE(O,'SPECTRA',ERR=909) NCSX,(DLAT(N),VLAT(N),BSQD(N),N=1,NCSX)
ISN 0054      (12) → $BNAMI(5)=+4+(NCSX)*(+20)
ISN 0055      $BNAMI(7)=4
ISN 0056      (13) { $NAMES(2)=VERSN
ISN 0057      CALL $YUD3 ($LGL$(23),$LNAM$,1,&909)
ISN 0058      WRITE($DSNAM$&RECRD)NCSX,(DLAT(N),VLAT(N),BSQD(N),N=1,NCSX)
              *
              *
C=====LOOP OVER ISOTOPES AND READ CROSS SECTIONS
ISN 0059      NIR=0
ISN 0060      DO 170 ISO=1,ISOX
CJ*** READ(M,DISOI(ISO),ITDCI(ISO),'XSECTION',ERR=907)
CJ *        NEG,((S(K,I),K=1,7),I=1,NEG)
ISN 0061      $NAMES(2)=VERSN
ISN 0062      $NAMES(3)=DISOI(ISO)
ISN 0063      (14) { $NAMES(4)=$RAJ(4,ITDCI(ISO))
ISN 0064      CALL $YUD2 ($LGL$(28),$LNAM$,1,&907)
ISN CC65      READ($DSNAM$&RECRD)NEG,((S(K,I),K=1,7),I=1,NEG)
C-----LOOP OVER ALL REACTION TYPES EXCEPT CHI (TYPE 7)
ISN 0066      DO 160 NR=1,6
C          LOOP OVER ALL SPECTRA
ISN 0067      DO 150 NCS=1,NCSX
ISN CC68      SIG(NCS)=0.0
ISN CC69      DO 140 I=1,NEG
ISN 0070      IF(NR.NE.4) SIG(NCS)=SIG(NCS)+FL(I,NCS)*S(NR,I)
ISN 0072      IF(NR.EQ.4) SIG(NCS)=SIG(NCS)+FL(I,NCS)*S(4,I)*S(3,I)
ISN 0074      140 CONTINUE
ISN 0075      IF(SIG(NCS).EQ.0.0) GO TO 160
ISN CC77      IF(NR.EQ.3) SIGP(NCS)=SIG(NCS)
ISN 0079      IF(NR.EQ.4) SIG(NCS)=SIG(NCS)/SIGP(NCS)
ISN CC81      IF(NR.NE.4) SIG(NCS)=SIG(NCS)/TF(NCS)
ISN 0083      150 CONTINUE
ISN 0084      NIR=NIR+1
ISN 0085      IF(NIR.GT.NIRXX) GO TO 904
ISN 0087      DISO(NIR)=DISOI(ISO)
ISN CC88      ITDC(NIR)=ITDCI(ISO)
ISN C089      REAC(NIR)=RNAME(NR)

```

Figure 3. Listing of the Precompiler Output (Continued)

11

Note the use of the pseudo variable $\emptyset$ in the data record name. This statement will store the data specified in the list, and catalog the name ' $\emptyset$ NEGRP'.VERSN.'SPECTRA'.

12

All WRITE statements generate an expression for determining the number of bytes to be written. Therefore, the indexing in the list must be explicit.

13

I/O keyword generation.

14

I/O keyword generation.

```

CJ*** WRITE(0,'XSECTION',DISO(NIR),ITDC(NIR),REAC(NIR),ERR=909)
CJ *      NCSX,(SIG(N),N=1,NCSX)

ISN C090      $BNAMI(5)=+4+(NCSX)*(+4)
ISN C091      $BNAMI(7)=0
ISN C092      $NAMES(2)=VERSN
ISN C093      $NAMES(4)=DISO(NIR)
ISN C094      $NAMES(5)=$RAJ(4,ITDC(NIR))
ISN C095      $NAMES(6)=REAC(NIR)
ISN C096      CALL $YUD3 ($LGL$(35),$LNAM$,1,&909)
ISN C097      WRITE($DSNAM$RECRD)NCSX,(SIG(N),N=1,NCSX)
C-----END REACTION AND ISOTOPE LOOPS

ISN C098      160  CONTINUE
ISN C099      170  CONTINUE
ISN C100      NIRX=NIR

C=====WRITE ISOTOPES RECORD
CJ*** WRITE(0,'ISOTOPES',ERR=909)NIRX,(DISO(N),ITDC(N),REAC(N),N=1,NIRX)
ISN C101      $BNAMI(5)=+4+(NIRX)*(+20)
ISN C102      $BNAMI(7)=4
ISN C103      $NAMES(2)=VERSN
ISN C104      CALL $YUD3 ($LGL$(40),$LNAM$,1,&909)
ISN C105      WRITE($DSNAM$RECRD)NIRX,(DISO(N),ITDC(N),REAC(N),N=1,NIRX)
*
*
```

Figure 3. Listing of Precompiler Output (Continued)

JOSHUA

DPSTM-500

Title An Example - Part II

Vol.
II

Page
B.2.21

Rev.

Date
10/70

I/O keyword generation.

I/O keyword generation.

15

16

```

C=====FINISHED
CJ*** STOP
ISN 0106      CALL $YUD98
CJ*** STOP 1 'COG, TOO MANY LATTICE'
ISN 0107      901      CALL $YUD99($LGL$(45),$LNAM$,1)
CJ*** STOP 2 'COG, TOO MANY ISOTOPES'
ISN 0108      902      CALL $YUD99($LGL$(50),$LNAM$,2)
CJ*** STOP 3 'COG, TOO MANY LATTICE*BUCKLINGS'
ISN 0109      903      CALL $YUD99($LGL$(55),$LNAM$,3)
CJ*** STOP 4 'COG, TOO MANY ISOTOPES*REACTIONS'
ISN 0110      904      CALL $YUD99($LGL$(61),$LNAM$,4)
CJ*** STOP 5 'COG, TOO MANY GROUPS'
ISN 0111      905      CALL $YUD99($LGL$(68),$LNAM$,5)
CJ*** STOP 6 'COG, INPUT RECORD MISSING'
ISN 0112      906      CALL $YUD99($LGL$(73),$LNAM$,6)
CJ*** STOP 7 'COG, MULTIGRP RECORD MISSING'
ISN 0113      907      CALL $YUD99($LGL$(79),$LNAM$,7)
CJ*** STOP 8 'COG, LATTICE RECORD MISSING'
ISN 0114      908      CALL $YUD99($LGL$(85),$LNAM$,8)
CJ*** STOP 9 'COG, WRITING ERROR'
ISN 0115      909      CALL $YUD99($LGL$(91),$LNAM$,9)
CJ*** STOP10 'COG, INCONSISTANT GROUP STRUCTURE'
ISN 0116      910      CALL $YUD99($LGL$(96),$LNAM$,10)

```

17

Figure 3. Listing of Precompiler Output (Continued)

JOSHUA

DPSTM-500

Title An Example - Part II

Vol. II Page B.2.23

Rev. Date 10/70

Generated call to \$YUD in lieu of simple STØP statement
(ISNO106) and qualified STØP statements (ISNO107 - ISNO116).
The messages on the abnormal STØP statements will be written
to the printer and to the JØB data set.

ISN C117

ISN C118

ISN 0119

ISN 0120

ISN C121

```

      REAL*8 $00001(41 )
      /* INPUT      , 'COG      ' , 'MULTIGRP' , 'ISOTOPES' , 'LATTICE '
      * , 'KEFF      ' , 'LFLUX   ' , 'ONEGRP  ' , 'SPECTRA ' , 'XSECTION'
      * , ' COG, TO' , 'O MANY L' , 'ATTICE   ' , 'O MANY I' , 'SOTOPES '
      * , 'ATTICE*B' , 'UCKLINGS' , 'SOTOPES*' , 'REACTION' , 'S
      * , 'O MANY G' , 'ROUPS    ' , ' COG, IN' , 'PUT RECO' , 'RD MISSI'
      * , 'NG        ' , ' COG, MU' , 'LTIGRP R' , 'ECORD MI' , 'SSING   '
      * , ' COG, LA' , 'TTICE RE' , 'CORD MIS' , 'SING    ' , ' COG, WR'
      * , 'TING ER' , 'ROR      ' , 'CONSISTA' , 'NT GROUP' , ' STRUCTU'
      * , 'RE          '
      *
      EQUIVALENCE ($LNAM$(1 ),$00001(1))
      LOGICAL*1 $L0001(1Q2)
      /*
      *      /Z00,Z02,Z03,Z01,Z02,Z03,Z03,Z03,Z00,Z04
      *      ,Z04,Z04,Z05,Z00,Z00,Z06,Z04,Z05,Z05,Z00
      *      ,Z00,Z07,Z03,Z03,Z08,Z00,Z09,Z05,Z05,Z03
      *      ,Z00,Z00,Z00,Z0A,Z03,Z06,Z08,Z00,Z0A,Z03
      *      ,Z03,Z08,Z00,Z04,Z03,Z03,Z08,Z0C,Z0D,Z03
      *      ,Z03,Z0B,Z0E,Z0F,Z04,Z04,Z0B,Z0C,Z10,Z11
      *      ,Z05,Z05,Z0B,Z0E,Z12,Z13,Z14,Z03,Z03,Z0B
      *      ,Z15,Z16,Z04,Z04,Z17,Z18,Z19,Z1A,Z04,Z04
      *      ,Z1B,Z1C,Z1D,Z1E,Z04,Z04,Z1F,Z20,Z21,Z22
      *      ,Z03,Z03,Z23,Z24,Z25,Z05,Z05,Z17,Z26,Z27
      *      ,Z28,Z29/
      *
      EQUIVALENCE ($LGL$(1 ),$L0001(1))
      END
  
```

Figure 3. Listing of Precompiler Output (Continued)

18

This generated array contains all invariant qualifier record names and STØP messages. It is used by the \$YUD routines.

19

This generated array is used by the \$YUD routines to select either invariant qualifiers or STØP messages from the array above.

JOSHUA

DPSTM-500

Title Part C - Module Preparation and Execution.

Vol. II Page
C.O.1

Rev. 1 Date
3/73

Contents

C.1 - Module Preparation and Execution

C.2 - An Example - Part III

JOSHUA DPSTM-500	Title Part C - Module Preparation and Execution.	Vol. II	Page C.O.2
		Rev. 1	Date 3/73

(Blank)

JOSHUA DPSTM-500	Title Module Preparation and Execution	Vol. II	Page C.1.1
		Rev. 1	Date 3/73

Work Done By

R. L. Boyce
A. H. Dobyns
J. C. Jensen

Report Written By

J. C. Jensen

Contents

	<u>Page</u>
1. Introduction	C.1.3
2. Compilation	C.1.4
3. Link Editing	C.1.6
4. Module Execution	C.1.8
4.1 General	C.1.8
4.2 Symbolic Parameters	C.1.8
4.3 Monitor Input Data Cards	C.1.10
4.4 Monitor Output Data	C.1.11
4.5 Module Input Data Cards	C.1.12
5. Procedure Listings	C.1.15

JOSHUA
DPSTM-500

Title Module Preparation and Execution

Vol.

II

Page

C.1.2

Rev.

1

Date

3/73

(Blank)

1. Introduction

This section deals with the methods and Job Control Language that are used to compile a source module, produce a JOSHUA load module, and execute the module with conventional batch facilities. The execution of a module from a terminal is described in Part E of this volume.

OS/360 catalogued procedures are provided to accomplish the following steps:

- 1) Translate a source deck into standard FORTRAN IV statements.
- 2) Compile the standard FORTRAN source deck.
- 3) Link edit the object code into a JOSHUA load module that may reside in a public or private library.
- 4) Execute the module.

The following sections describe the use of these procedures. Section C.5 contains a complete listing of each procedure.

JOSHUA DPSTM-500	Title Module Preparation and Execution	Vol.	Page
		II	C.1.4
		Rev.	Date
			10/79

2. Compilation

The compilation of a JOSHUA module or subroutine involves the translation of the source code to standard FORTRAN IV statements followed by the compilation of the FORTRAN statements to produce an object module. This is done in one job step by dynamically invoking both the precompiler and the FORTRAN H or G-level compiler. The catalogued procedure, FØRTJC, is used for this step. In addition to the standard symbolic parameters for a FORTRAN compile step, the following parameters are provided.

- 1) PRINT - Defines the data set for precompiler output listing.
- 2) PRECØMP - Member name of the JØSHUA precompiler.
- 3) PCMPDS - Name of the library containing the JOSHUA precompiler.

Table I contains a complete list of the symbolic parameters for a compile step.

TABLE I

Symbolic Parameters for JOSHUA Compilation

<u>Keyword</u>	<u>Definition</u>	<u>Default</u>	<u>Options</u>
FØUT	Data set for compiler output listings and system messages.	'SYSØUT=A'	Any sequential data set.
PRINT	Data set for precompiler output listings.	'SYSØUT=A'	Any sequential data set or DUMMY.
PRØGRAM	Name of FORTRAN compiler.	IEKAAOO (H-level)	IEYFØRT (G-level)
ID	FORTTRAN ID option.	ID	NOID
SØURCE	FORTTRAN SOURCE option	SOURCE	NOSOURCE
LIST	FORTTRAN LIST option.	NOLIST	LIST
XREF	FORTTRAN H XREF option	NØXREF	XREF
ØPT	FORTTRAN H optimization level.	0	1, 2
CTIME	CPU time (min., sec.) for compilation step.	'(05,00)'	'(01,00)' → '(99,59)'
MAP	FORTTRAN MAP option	MAP	NØMAP
LIBRARY	Library name that contains the FORTRAN compiler.	'SYS1.LINKLIB'	Any PDS name that contains the member specified by PRØGRAM.
FSIZE	Region size for compilation step.	240K	A smaller size may be used if the G-level compiler is used (~ 110K).*
PRECØMP	Name of precompiler.	PRECØM2	Any name.
PCMPDS	Library name that contains the precompiler.	JØS.PRECØMP	Any name.

Note: These parameters are applicable to the following catalogued procedures: FØRTJC, FØRTJCL, FØRTJCLP, FØRTJCLG

*The size may be increased for either compiler to enable the compilation of very large programs.

JOSHUA DPSTM-500	Title Module Preparation and Execution	Vol. II	Page. C.1.6
		Rev. 2	Date 9/73

3. Link Editing

A JOSHUA load module is prepared by link editing an object module produced in a compile step and including a linkage routine that provides the module with dynamic linkage capability to the input/output programs and to other JOSHUA modules. The resulting module, therefore, does not contain FORTRAN input/output routines, (IBCOM), the JOSHUA data management routine, (BNAM), or any modules that are referenced in EXECUTE statements.

Three catalogued procedures, FØRTJCLP, FØRTJCL, and FØRTJCLG may be used for this step.

The FØRTJCLP procedure is used to place the module in a private library. The library name is specified by the PDS parameter and the module name by the MEMBER parameter. The entry will always be MAIN.

The FØRTJCL procedure is used to place the module in a "public" JOSHUA library, "JOS.TESTLIB". Modules should only be placed in this library when they are being tested as part of a JOSHUA path. The symbolic parameters PDS and MEMBER may not be used; therefore, ENTRY and NAME cards must be supplied as input data. In order to effectively maintain this library, Computer Operations must be notified whenever a new module is placed in the library and when it is no longer needed.

The symbolic parameters used in the link edit step are shown in Table II. They are the same as the standard parameters with the exception of LINK. This parameter defines the name of the linkage routine that is included in each JOSHUA module.

The FØRTJCLG procedure is used to place a module in a private library or a temporary library. The library will be temporary if the PDS and MEMBER parameters are not specified. This procedure contains an execution step, (described in the next section), but is otherwise the same as the FØRTJCLP procedure.

TABLE II

Symbolic Parameters for JOSHUA Link Edit Steps

<u>Keyword</u>	<u>Definition</u>	<u>Default</u>	<u>Options</u>
ØVLY	Link Editor option.	'('	May be used to add any link editor options to the default list.
LREF	Link Editor option.	MAP	XREF
LØUT	Output data set for Link Editor messages.	'SYSØUT=A'	Any sequential data set.
LINK	Name of JOSHUA linkage routine.	\$LINK	Any name.
LTIME	CPU time (min., sec.) for link edit step.	'(00,30)'	'(00,01)'→'(99,59)'
LSIZE	Region size for Linkage Editor.	110K	Any value.
PDS*	Name of private library.	'&&GØSET'	Any name.
MEMBER*	Name of JOSHUA module.	ZØ	Any name.
IEWL*	Name of Linkage Editor	XEWL	None.
S2*	Link Editor Option	13K	None.
EXTLIN	REPLACE control cards.	SRL.RCH33782	None.
SYSLIB	Name of private sub-routine library.	SRL.DUMMYLIB	Any name.

* Used only with the procedures: FØRTJCLP and FØRTJCLG

JOSHUA DPSTM-500	Title Module Preparation and Execution	Vol. II	Page C.1.8
		Rev. 1	Date 3/7

4. Module Execution

4.1 General

Modules are executed by the JOSHUA Monitor System (JMS) which is utilized thru the FØRTJCLG and JOSHUA catalogued procedures. The MEMBER symbolic parameter is used to specify the module name. The JOSHUA procedure is for execution only, while FØRTJCLG is a compile, link edit, and execute procedure.

In addition to controlling the execution of the specified module, JMS performs several setup and housekeeping functions. The input for these functions is supplied through symbolic parameters on the EXEC card and through monitor input cards. Special delimiter cards are used to separate monitor input data cards from module input data cards.

The remainder of this section discusses the use of the above-mentioned procedures to accomplish the execution of a JOSHUA module. Further information concerning JMS operation can be found in Part E of this volume.

4.2 Symbolic Parameters

The following symbolic parameters may be used to specify the execution of a JOSHUA module. Table III contains a complete list of the parameters including their default values.

MEMBER=module name specifies the name of the JOSHUA module to be executed.

PDS='PDS name' specifies the library that contains the module to be executed. Three additional libraries will be searched if the module is not located in the specified library (JØS.BATCH, JØS.SYSTEM3, JØS.PGMSØ1)

TIMER=ON or OFF specifies whether or not to produce program timing data for the particular run. Note that this data will only be produced if the module has previously been precompiled with the program timings option card.

BUFNØ=n specifies the number of in-core buffers to be used by the data management routine, BNAM. If the module to be executed performs a large number of I/O operations, increasing BUFNØ from 10 to 20 or 30 may significantly reduce the over-all I/O time for the job. Each buffer occupies 816 bytes of core storage, so the GØSIZE parameter must be increased when BUFNØ is increased.

USER=User data set name specifies the name of the User data set that will be accessed during job execution. This name must always consist of four numeric characters.

JOB=jobname specifies the name of the Job data set to be used during module execution. All output records will be written to this data set. The name must be of the form: AAAAnnnn

where: AAAA is an arbitrary four-character name that does not begin with a numeric or special characters

nnnn should be the user's badge number.

Note that this name is passed from the monitor to the module and may be referred to in the module as \$JØB.

SPACE=n specifies the number of blocks of direct access space to be allocated to the Job data set. A block of space is currently defined as 100 physical data records of 400 bytes each and two associated tree records. A catalog display will appear at the end of each job that will show the exact number of records currently being used in the data set. If SPACE=0 is specified, a Job data set will not be allocated.

DISP=NEW,TMP, or OLD specifies the actions to be taken with respect to the Job data set.

NEW causes JMS to allocate a data set with a name specified by JØB and size specified by SPACE. If the same name already exists in the JOSHUA catalog the job will be immediately terminated. If allocation is successful, the data set will be retained when the job is terminated. Data sets currently have a retention period of 8 days.

TMP causes the same action as NEW except that the data set is not retained when the job is terminated.

ØLD causes JMS to locate the data set in the JOSHUA catalog and update the expiration date by the retention period. If the data set cannot be located, the disposition is changed to NEW.

RLSE=YES or NO specifies whether or not to release unused space in the Job data set when the job is terminated. NO should only be specified if records are to be added to the data set at a later time.

JOSHUA DPSTM-500	Title Module Preparation and Execution	Vol. II	Page C.1.10
		Rev. 2	Date 9/73

SSPACE=n specifies the number of blocks of direct access space to be allocated to a Scratch data set. A block of space is currently defined as 100 physical data records of 400 bytes each and 2 associated tree records. A catalog display will appear at the end of each job that will show the exact number of Scratch records that were used during the job. If SSPACE=0 is specified, a Scratch data set will not be allocated for the job. This data set is never retained after a job terminates.

4.3 Monitor Input Data Cards

Monitor input data is used for four purposes: 1) Scratching unwanted Job data sets, 2) changing the standard BNAM read hierarchy, 3) establishing the protection category of the Job and Scratch data sets, and 4) performing monitor function requests. Item 4 is discussed in PART E of this volume.

Old, unwanted Job data sets may be scratched by placing the keyword SCRATCH= in columns one through eight of a card and placing the jobname in columns nine through sixteen, e.g.,

```

Column 1
↓
SCRATCH=TEST2791

```

There is no limit to the number of SCRATCH= cards. This scratch operation is done prior to Job data set allocation so that the current jobname may be specified in a SCRATCH= card, thus preventing a duplicate jobname error. Note that, if the jobname specified in the SCRATCH= card does not exist, the card will be ignored by the system. The card will also be ignored if the data set contains protected data and a file category is not specified. (See Part E, Section 12, "Data Protection Facilities" for further information.)

The standard BNAM search sequence of Job-User-Standard may be changed by placing the keyword DATASET= in columns one through eight of a card. The desired sequence of up to four data set names should be entered starting in column nine and should be separated by commas, e.g.,

```

Col.1    Col.9
↓        ↓
DATASET=2791,3245,2816,STD

```

In this example the three specified User data sets will be searched before the standard data base. Note that the keyword STD must be used if the standard data base is to be searched. Only the data sets specified in this card will be searched if this card is used, however, all output records will still be written to the specified Job data set.

If protected data is used in a job, a card with the keyword, CATEGORY=, in columns one through nine must be used to specify the category of the Job and Scratch data sets. The number of the category should start in column ten, e.g.,

Col.1 Col.10

↓ ↓
CATEGORY=1. If a value of zero is specified, the card will be ignored by the system. All jobs containing these cards must be submitted by operations as "Special Handling." This card must precede all other monitor input cards.

4.4 Monitor Output Data

All monitor input data cards will appear on the output listing. In addition, there may be one or more error messages, BNAM statistics, and catalog displays for the Job and Scratch data sets. Error messages are discussed in the next section.

BNAM statistics are listed in the following form:

NUMBER OF READ REQUESTS=

NUMBER OF WRITE REQUESTS=

NUMBER OF DATA ACCESSES=

NUMBER OF CATALOG ACCESSES=

The number of read and write requests pertain strictly to JOSHUA I/O statements. All requests are tallied even when error conditions occur. These statistics may be helpful in the following ways:

- 1) The number of read or write requests may be useful debugging data.
- 2) The number of data and catalog accesses should be approximately the same as the number of disk accesses appearing at the end of the JCL listing. If the number is much smaller, a considerable amount of system overhead is occurring. This could be caused by a poorly designed overlay structure.
- 3) The ratio of the number of I/O requests to the number of catalog accesses should be fairly high, i.e., at least 2 to 1. If it is not, the data structures used in the module should be examined to determine the cause.

Figure 1 contains a sample catalog display of a Job data set. A complete description of all the parameters appears in Part E of this volume in the discussion of the Display

JOSHUA DPSTM-500	Title Module Preparation and Execution	Vol. II	Page C.1.12
		Rev. 2	Date 9/73

Catalog monitor function. This display may be useful debugging data. Also, it shows the exact amount of space that has been used in the data set, thus permitting better space allocation estimates.

```

          DATA SET MMAT3947  IS ONLINE  -FILE CATEGORY  00000000
RECORD NUMBER  76 - CREATED 102372 - EXPIRES 103172
TREE NUMBER           23          DATA NUMBER           28
BASE                  3814        BASE                   18721
RECORDS USED          5          RECORDS USED            265
LAST NODE             24        EXTENT                   1
MAX RECORDS           6          MAX RECORDS             300

```

Figure 1. Catalog Display

4.5 Module Input Data Cards

Input data cards may be read from one or more modules in a path. A special monitor input command, Execute Member (EM), must be used whenever data cards are read. In addition, cards that are ready by modules in a path must be separated by a special delimiter card, S/E, for each module execution. Figure 2 illustrates the use of input data cards.

```

//GO.SYSIN DD *
CATEGORY=1
SCRATCH=TEMP2791
DATASET=2791,3208   These cards are optional.
S/E
EM
S/E

    The first set of data cards to be read goes here.
S/E

    Subsequent sets of data cards are separated by the
    special delimiters, S/E.
S/E
/*

```

Figure 2

*See Part E, Section 8, "Using JMS as a Batch Monitor" for further information.

TABLE III

Symbolic Parameters for JOSHUA Module Execution

<u>Keyword</u>	<u>Definition</u>	<u>Default</u>	<u>Options</u>
GØUT	System output device.	'SYSØUT=A'	Any sequential data set.
GØ	Name of control program.	JTMON	Any name.
GØSIZE	Region size for module execution.	120K	120K→nK
GTIME	CPU time for module execution.	'(01,00)'	'(00,01)' → '(99,59)'
PDS	Name of library containing module to be executed.	JOS.PGMSOL	Any name.
PØUT	Device for punched output.	'SYSØUT=B'	Any sequential data set
JØB	Jobname for Job data set.	'TEMPO0000'	1) An 8-character name beginning with an alpha character and ending with the user badge number.
USER	User data set name.	'2222'	0000→9999
SPACE	Job data set space to be allocated.	1	0 → n
SSPACE	Scratch data set space to be allocated.	1	0 → n
BUFNØ	Number of BNAM buffers.	10	4 → n
DISP	Status of Job data set.	NEW	OLD or TMP
RLSE	Space release parameter.	YES	NO
MEMBER	Name of module to be executed.	ZØ	Module name.
DUMP	Output data set for SYSUDUMP.	DUMMY	'SYSØUT=A'
TIMER	Program timing parameter.	OFF	ON

JOSHUA DPSTM-500	Title Module Preparation and Execution	Vol. II	Page C.1.14
		Rev. 1	Date 3/77

TABLE III (Cont.)

Symbolic Parameters for JOSHUA Module Execution

<u>Keyword</u>	<u>Definition</u>	<u>Default</u>	<u>Options</u>
ACT1	Accounting data.	L4120000	None

Note: These parameters are used with procedures FØRTJCLG and JØSHUA.

5. Procedure Listings

JOSHUA PROCEDURES

```

MEMBER NAME  FORTJC
//FRTJPROC   PROC FOOT='SYSOUT=A',PRINT='SYSOUT=A',PROGRAM=IEKAA00,      C
//           ID=ID,SOURCE=SOURCE,LIST=NOLIST,XREF=NOXREF,OPT=0,          C
//           CTIME='(05,00)',MAP=MAP,LIBRARY='SYS1.LINKLIB',            C
//           FSIZE=240K,PRECOMP=PRECOM2,PCMPDS='JOS.PRECOMP'
//FRT       EXEC PGM=JEXPCOMP,ACCT=(L4101-09,J),REGION=&FSIZE.,          C
//           TIME=&CTIME.,                                                C
//           PARM='&MAP.,&ID.,&LIST.,&SOURCE.,&XREF.,OPT=&OPT'
//PRECOMP    DD  DSNAME=&PCMPDS.(&PRECOMP),DISP=SHR
//MEMBER     DD  DSNAME=&LIBRARY.(&PROGRAM),DISP=SHR
//SYSPRINT    DD  &FOOT.,DCB=BLKSIZE=121
//FRTPRNT    DD  &FOOT
//SYSOUT1    DD  DSNAME=&&OUT1,SPACE=(3072,(200,20),,,ROUND),UNIT=SYSDA
//SYSOUT2    DD  DSNAME=&&OUT2,UNIT=SYSDA,SPACE=(3072,(40,20),,,ROUND),    C
//           DCB=(RECFM=FB,LRECL=1024,BLKSIZE=4096,BUFNO=1)
//SYSLIN     DD  DSNAME=&&MAIN,UNIT=SYSDA,DISP=(MOD,PASS),                C
//           SPACE=(CYL,(3,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT05FOOT   DD  &PRINT
//FT10FOOT   DD  DSNAME=&&LIN,UNIT=SYSDA,SPACE=(3072,(100,20),,,ROUND),    C
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)

```

JOSHUA PROCEDURES

MEMBER NAME FORTJCLP

```
//FORTJCLP PROC FSIZE=240K,CTIME='(05,00)',
//          MAP=MAP,ID=ID,LIST=NOLIST,XREF=NOXREF,OPT=0,
//          SOURCE=SOURCE,FOUT='SYSOUT=A',PRINT='SYSOUT=A',
//          LIBRARY='SYS1.LINKLIB',PROGRAM=IEKAA00,
//          PCMPDS='JOS.PRECOMP',PRECOMP=PRECOM2,
//          IEWL=XEWL,LSIZE=110K,LTIME='(00,30)',
//          LREF=MAP,S2=13K,OVLY=',',LINK=LINK,LOUT='SYSOUT=A',
//          PDS='&&GOSET',MEMBER=ZU,SYSLIB='SRL.DUMMYLIB',
//          EXTLIN='SRL.RCH33782'
// *
//FORT      EXEC  PGM=JEXPCOMP,ACCT=(L4101-09,J),
//              REGION=&FSIZE.,TIME=&CTIME.,
//              PARM='&ID.,&LIST.,&MAP.,&SOURCE.,&XREF.,OPT=&OPT'
//FORTPRINT DD &FOUT

//FTJ05F001 DD &PRINT
//FTJ06F001 DD DSN=&EXTLIN,UNIT=SYSDA,SPACE=(3072,(100,20),,,ROUND),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//MEMBER    DD DISP=SHR,DSN=&LIBRARY.(&PROGRAM)
//PRECOMP   DD DISP=SHR,DSN=&PCMPDS.(&PRECOMP)
//SYSLIN    DD DSN=&&MAIN,UNIT=SYSDA,DISP=(MOD,PASS),SPACE=(CYL,(3,1)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT  DD &FOUT.,DCB=BLKSIZE=121
//SYSOUT1   DD DSN=&OUT1,UNIT=SYSDA,SPACE=(3072,(200,20),,,ROUND),
//          DISP=(,PASS)
//SYSOUT2   DD DSN=&OUT2,UNIT=SYSDA,SPACE=(3072,(40,20),,,ROUND),
//          DCB=(RECFM=FB,LRECL=1024,BLKSIZE=4096,BUFNO=1)
// *
//LKED      EXEC  PGM=&IEWL,REGION=&LSIZE.,TIME=&LTIME.,
//              PARM='LET,LIST,&LREF.,&OVLY.,SIZE=(&LSIZE,&S2)',
//              ACCT=(L4105-08,J),COND=(9,LT,FORT)
//PROGRAM   DD DSN=&PDS.(&MEMBER),DISP=(MOD,PASS),
//              UNIT=SYSDA,SPACE=(1RK,0)
//SYSLIB DD DSN=&SYSLIB,DISP=SHR
//          DD DISP=SHR,DSN=JOS.JOSHLIB
//          DD DISP=SHR,DSN=SYS1.FORTLIB
//          DD DISP=SHR,DSN=SRL.FORTLIB
//SYSLIN    DD DISP=SHR,DSN=PNCH3135(&LINK)
//          DD DISP=SHR,DSN=SYS1.SETSSI
//          DD DISP=SHR,DSN=SYS1.PROCLIB(NTRYMAIN)
//          DD DISP=(OLD,DELETE),DSN=&&MAIN
//          DD DISP=SHR,DSN=&EXTLIN
//          DD DDNAME=SYSIN
//SYSLMOD   DD DSN=&T(&MEMBER),DISP=(,DELETE),
//              UNIT=SYSDA,SPACE=(CYL,(2,1,1),RLSE),VOL=REF=*.PROGRAM,
//              DCB=(RECFM=U,BLKSIZE=7294)
//SYSPRINT  DD &LOUT
//SYSOUT1   DD DSN=&OUT1,DISP=(OLD,DELETE)
```

JOSHUA PROCEDURES

```

MEMBER NAME FORTJCL
//FORTJCL PROC FSIZE=240K,CTIME='(05,00)',
//          MAP=MAP,LD=LD,LIST=NOLIST,XREF=NOXREF,OPT=0,
//          SOURCE=SOURCE,FOOT='SYSOUT=A',PRINT='SYSOUT=A',
//          LIBRARY='SYS1.LINKLIB',PROGRAM=IEKAA00,
//          PCMPDS='JDS.PRECOMP',PRECOMP=PRECOM2,
//          LSIZE=110K,LTIME='(00,30)',
//          LREF=MAP,OVLY=' ',LINK=LINK,LOUT='SYSOUT=A',
//          EXTLIN='SRL.RCH35782',
//          SYSLIB='SRL.DUMMYLIB'
//FJRT EXEC PGM=JEXPCOMP,ACCT=(L4101-09,J),
//          REGION=&FSIZE.,TIME=&CTIME.,
//          PARM='&LD.,&LIST.,&MAP.,&SOURCE.,&XREF.,OPT=&OPT'
//FORTPRINT DD &FJRT

//FT05F001 DD &PRINT
//FT10F001 DD DSN=&EXTLIN,UNIT=SYSDA,SPACE=(3072,(100,20),,,ROUND),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//MEMBER DD DISP=SHR,DSN=&LIBRARY.(&PROGRAM)
//PRECOMP DD DISP=SHR,DSN=&PCMPDS.(&PRECOMP)
//SYSLIN DD DSN=&MAIN,UNIT=SYSDA,DISP=(MOD,PASS),SPACE=(CYL,(3,1)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT DD &FJRT.,DCB=BLKSIZE=121
//SYSUT1 DD DSN=&OUT1,UNIT=SYSDA,SPACE=(3072,(200,20),,,ROUND),
//          DISP=(,PASS)
//SYSUT2 DD DSN=&OUT2,UNIT=SYSDA,SPACE=(3072,(40,20),,,ROUND),
//          DCB=(RECFM=FB,LRECL=1024,BLKSIZE=4096,BUFNO=1)
//LKED EXEC PGM=IEWL,REGION=&LSIZE.,TIME=&LTIME.,COND=(9,LT,FORT),
//          PARM=(LET,LIST,&LREF.,&OVLY),ACCT=(L4106-08,J)
//SYSLIB DD DSNNAME=&SYSLIB,DISP=SHR
//          DD DISP=SHR,DSN=JDS.JOSHLIB
//          DD DISP=SHR,DSN=SYS1.FORTLIB
//          DD DISP=SHR,DSN=SRL.FORTLIB
//SYSLIN DD DISP=SHR,DSN=PNCH3135(&LINK)
//          DD DISP=SHR,DSN=SYS1.SETSSI
//          DD DISP=(OLD,DELETE),DSN=&MAIN
//          DD DISP=SHR,DSN=&EXTLIN
//          DD DDNAME=SYSIN
//SYSLMOD DD DISP=SHR,DSN=JDS.TESTLIB
//SYSPRINT DD &LOUT
//SYSUT1 DD DSN=&OUT1,DISP=(OLD,DELETE)

```

JOSHUA PROCEDURES

MEMBER NAME FORTJCLG

```
//FORTJCLG PROC FSIZE=240K,CTIME='(05,00)',
//      MAP=MAP,ID=10,LIST=NULIST,XREF=NOXREF,OPT=0,
//      SOURCE=SOURCE,FOUT='SYSOUT=A',PRINT='SYSOUT=A',
//      LIBRARY='SYS1.LINKLIB',PROGRAM=IEKAA00,
//      PCMPDS='JOS.PRECOMP',PRECOMP=PRECOM2,
//      IEWL=XEWL,LSIZE=110K,LTIME='(00,30)',
//      LREF=MAP,S2=13K,OVLY='',LINK=LINK,LOUT='SYSOUT=A',
//      PDS='&&GOSET',MEMBER=20,
//      EXTLIN='SRL.RCH33782',
//      GD=JTMON,GDSIZE=120K,GTIME=1,
//      BUFNO=10,DISP=NEW,JOB=TEMP0000,RLSE=YES,
//      SPACE=1,SSPACE=1,TIMER=OFF,USER=2222,
//      DUMP=DUMMY,GOUT='SYSOUT=A',POUT='SYSOUT=B',
//      SYSLIB='SRL.DUMMYLIB'
//FURT      EXEC PGM=JEXPCOMP,ACCT=(L4101-09,J),
//      REGION=&FSIZE.,TIME=&CTIME.,
//      PARM='&ID.,&LIST.,&MAP.,&SOURCE.,&XREF.,OPT=&OPT'
//FORTPRNT DD &FOUT

//FT06F001 DD &PRINT
//FT10F001 DD DSN=&&EXTLIN,UNIT=SYSDA,SPACE=(3072,(100,20),,,ROUND),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//MEMBER    DD DISP=SHR,DSN=&LIBRARY.(&PROGRAM)
//PRECOMP   DD DISP=SHR,DSN=&PCMPDS.(&PRECOMP)
//SYSLIN    DD DSN=&&MAIN,UNIT=SYSDA,DISP=(MOD,PASS),SPACE=(CYL,(3,1)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT  DD &FOUT.,DCB=BLKSIZE=121
//SYSUT1    DD DSN=&&UT1,UNIT=SYSDA,SPACE=(3072,(200,20),,,ROUND),
//      DISP=(,PASS)
//SYSUT2    DD DSN=&&UT2,UNIT=SYSDA,SPACE=(3072,(40,20),,,ROUND),
//      DCB=(RECFM=FB,LRECL=1024,BLKSIZE=4096,BUFNO=1)
//LKED      EXEC PGM=&IEWL,REGION=&LSIZE.,TIME=&LTIME.,
//      PARM='LET,LIST,&LREF.,&OVLY.,SIZE=(&LSIZE,&S2)',
//      ACCT=(L4106-08,J),COND=(9,LT,FORT)
//PROGRAM   DD DSN=&PDS.(&MEMBER),DISP=(MOD,PASS),
//      UNIT=SYSDA,SPACE=(TRK,0)
//SYSLIB    DD DSNNAME=&SYSLIB,DISP=SHR
//      DD DISP=SHR,DSN=JOS.JOSHLIB
//      DD DISP=SHR,DSN=SYS1.FORTLIB
//      DD DISP=SHR,DSN=SRL.FORTLIB
//SYSLIN    DD DISP=SHR,DSN=PNCH3135(&LINK)
//      DD DISP=SHR,DSN=SYS1.SETSSI
//      DD DISP=SHR,DSN=SYS1.PROCLIB(ENTRYMAIN)
//      DD DISP=(OLD,DELETE),DSN=&&MAIN
//      DD DISP=SHR,DSN=&EXTLIN
//      DD DDNAME=SYSIN
//SYSLMOD   DD DSN=&T(&MEMBER),DISP=(,DELETE),
//      UNIT=SYSDA,SPACE=(CYL,(2,1,1),RLSE),VOL=REF=*.PROGRAM,
//      DCB=(RECFM=U,BLKSIZE=7294)
//SYSPRINT  DD &LOUT
//SYSUT1    DD DSN=&&UT1,DISP=(OLD,DELETE)
```

JOSHUA PROCEDURES

MEMBER NAME FORTJCLG

```

// *
// GO      EXEC  PGM=&GO.,REGION=&GOSIZE.,TIME=(&GTIME,30),
//          ACCT=(L4120-00,J,&MEMBER),
//          PARM=(&USER,&JOB,&MEMBER,'(&GTIME,30)',
//          &SPACE,&SSPACE,&BUFNO,&RLSE,&DISP,&TIMER)
// STEPLIB DD DISP=(OLD,PASS),DSN=*.LKED.PROGRAM
//          DD DISP=SHR,DSN=JOS.BATCH
//          DD DISP=SHR,DSN=JOS.SYSTEM3
//          DD DISP=SHR,DSN=JOS.PGMS01
// JQSQUEUE DD DISP=SHR,DSN=JOS.JOSHUA.BQSQUEUE
// SYSPRINT DD SYSOUT=A
// SYSUDUMP DD &DUMP
// FT05F001 DD DDNAME=SYSIN
// FT06F001 DD &GOUT
// FT07F001 DD &PJUT
// FT20F001 DD DISP=SHR,DSN=JOS.JOSHUA.NCAT
// FT21F001 DD DISP=SHR,DSN=JOS.JOSHUA.JDSCB
// FT22F001 DD DISP=SHR,DSN=JOS.JOSHUA.JCEB
// FT23F001 DD DISP=SHR,DSN=JOS.JOSHUA.TREE1
// FT24F001 DD DISP=SHR,DSN=JOS.JOSHUA.TREE2
// FT25F001 DD DISP=SHR,DCB=BUFNO=1,DSN=JOS.JOSHUA.DATA1
// FT26F001 DD DISP=SHR,DCB=BUFNO=1,DSN=JOS.JOSHUA.DATA2
// FT27F001 DD DISP=SHR,DCB=BUFNO=1,DSN=JOS.JOSHUA.DATA3
// FT28F001 DD DISP=SHR,DCB=BUFNO=1,DSN=JOS.JOSHUA.DATA4
// FT31F001 DD DISP=SHR,DCB=BUFNO=1,DSN=JOS.JOSHUA.DATA7
// FT37F001 DD DSN=JOS.JOSHUA.RDATA2,DISP=SHR,DCB=BUFNO=1
// FT38F001 DD DISP=SHR,DCB=BUFNO=1,DSN=JOS.JOSHUA.RDATA3
// FT39F001 DD DISP=SHR,DCB=BUFNO=1,DSN=JOS.JOSHUA.RDATA1
// FT40F001 DD DISP=SHR,DSN=JOS.SECNAM
// FT61F001 DD DISP=SHR,DSN=JOS.JOSHUA.POOL

```


JOSHUA PROCEDURES

MEMBER NAME JOSHUA

```
//JOSHUA PROC GOUT='SYSOUT=A',GO=JTMON,GOSIZE=120K,GTIME=1,
//          POUT='SYSOUT=B',JOB=TEMPO000,DISP=NEW,
//          USER=2222,SPACE=1,MEMBER=ZO,RLSE=YES,TIMER=OFF,
//          DUMP=DUMMY,ACT1=L4120000,BUFNO=10,SSPACE=1,PDS='JOS.PGMS01'
//GO EXEC PGM=&GO,REGION=&GOSIZE,ACCT=(&ACT1,J,&MEMBER),
//      TIME=(&GTIME,30),PARM=(&USER,&JOB,&MEMBER,'(&GTIME,30)',
//      &SPACE,&SSPACE,&BUFNO,&RLSE,&DISP,&TIMER)
//STEPLIB DD DSN=JOS.PGMS01,DISP=SHR
//      DD DSN=JOS.BATCH,DISP=SHR
//      DD DSN=JOS.SYSTEM3,DISP=SHR
//      DD DSN=JOS.PGMS01,DISP=SHR
//JOSQUEUE DD DISP=SHR,DSN=JOS.JOSHUA.BUSQUEUE
//SYSPRINT DD SYSOUT=A
//FI05FOU1 DD DDNAME=SYSIN
//FI06FOU1 DD &GOUT
//FI07FOU1 DD &POUT
//SYSODUMP DD &DUMP
//FI20FOU1 DD DISP=SHR,DSN=JOS.JOSHUA.NCAT
//FI21FOU1 DD DISP=SHR,DSN=JOS.JOSHUA.JDSCB
//FI22FOU1 DD DSN=JOS.JOSHUA.JCEB,DISP=SHR
//FI23FOU1 DD DSN=JOS.JOSHUA.TREE1,DISP=SHR
//FI24FOU1 DD DSN=JOS.JOSHUA.TREE2,DISP=SHR
//FI25FOU1 DD DSN=JOS.JOSHUA.DATA1,DISP=SHR,DCB=BUFNO=1
//FI26FOU1 DD DSN=JOS.JOSHUA.DATA2,DISP=SHR,DCB=BUFNO=1
//FI27FOU1 DD DSN=JOS.JOSHUA.DATA3,DISP=SHR,DCB=BUFNO=1
//FI28FOU1 DD DSN=JOS.JOSHUA.DATA4,DISP=SHR,DCB=BUFNO=1
//FI31FOU1 DD DSN=JOS.JOSHUA.DATA7,DISP=SHR,DCB=BUFNO=1
//FI37FOU1 DD DSN=JOS.JOSHUA.RDATA2,DISP=SHR,DCB=BUFNO=1
//FI38FOU1 DD DISP=SHR,DCB=BUFNO=1,DSN=JOS.JOSHUA.RDATA3
//FI39FOU1 DD DSN=JOS.JOSHUA.RDATA1,DISP=SHR,DCB=BUFNO=1
//FI40FOU1 DD DSN=JOS.SEGNAM,DISP=SHR
//FI61FOU1 DD DSN=JOS.JOSHUA.POOL,DISP=SHR
```

JOSHUA

DPSTM-500

Title An Example - Part III

Vol.
II

Page
C.2.1

Rev.
1

Date
3/73

Work Done By

J. C. Jensen

Report Written By

J. C. Jensen

Contents

	<u>Page</u>
1. FØRTJC Procedure	C.2.3
2. FØRTJCLG Procedure	C.2.5
3. FØRTJCLP Procedure	C.2.9
4. FØRTJCL Procedure	C.2.11
5. JOSHUA Procedure	C.2.13

JOSHUA DPSTM-500	Title	An Example - Part III	Vol.	Page
			II	C.2.2
			Rev.	Date
			1	3/7

(Blank)

1. FØRTJC Procedure

```
//STEPl EXEC FORTJC,PROGRAM=IEYFORT,  
// FSIZE=110K,PRECOMP=TESTVSN  
//FORT.SYSIN DD *  
  
MODULE  
:  
:  
:  
STOP  
END  
  
/*
```

Figure 1. Use of FØRTJC Procedure

Figure 1 shows the use of the FØRTJC procedure to compile the module CØG. In this example a test version of the precompiler will be used, PRECØMP=TESTVSN, the FORTRAN G compiler will be used, PRØGRAM=IEYFØRT, and the region size will be reduced from 240K to 110K, FSIZE=110K.

JOSHUA DPSTM-500	Title An Example - Part III	Vol. II	Page C.2.4
		Rev. 1	Date 3/73

(Blank)

2. FØRTJCLG Procedure

```
//STEP1 EXEC FØRTJCLG,PRINT=DUMMY,XREF=XREF,MEMBER=CØG,
// USER=3208,JOB=TST12791,SPACE=3,DISP=TMP,
// SSPACE=0,GØSIZE=170K,DUMP='SYSØUT=A'
//FØRT.SYSIN DD *
      MØDULE
      .
      .
      STØP
      END
//GØ.SYSIN DD *
SCRATCH=TST12791
/*
```

Figure 2. Use of FØRTJCLG Procedure

Figure 2 shows the use of the FØRTJCLG procedure to compile the CØG module, link edit to a temporary PDS, and execute the module. In this example, the precompiler output is suppressed, PRINT=DUMMY, the FORTRAN H compiler is used, PRØGRAM=IEKAA00 by default, and a FORTRAN cross reference listing is requested, XREF=XREF, in the FØRT step.

In the LKED step, the module will be placed in a temporary PDS since a PDS name was not specified, and will be named CØG, MEMBER=CØG. The member name may be omitted if desired when using this procedure. Figure 3 shows a link edit map of the CØG module. Note that the data management routine, BNAM, and all of the FORTRAN I/O routines appear only as entry points in the control section, LINK. These routines are loaded by the system at execution time. The total system overhead including these routines is about 100K bytes of core storage. This value should be added to the module size to obtain the proper GØSIZE for module execution.

There are seven parameters specified for the GØ step in this example. Note that none of the parameters for this procedure are required. Some small applications could be run using the entire set of default values. However, in this example data will be retrieved from Job data set TST12791, JOB=TST12791, then User data set 3208, USER=3208, and then the Standard data base. Also, all results will be written to data set TST12791. Three blocks of space, 300 records of 400 bytes each and 6 tree pages, SPACE=3, will be allocated for the Job data set and it will be scratched at the termination of the job, DISP=TMP. There will be no Scratch data set allocated for this job, SSPACE=0. The

JOSHUA DPSTM-500	Title An Example - Part III	Vol.	Page
		II	C.2.6
		Rev.	Date
		1	3/73

(Blank)

MODULE MAP

CONTROL SECTION			ENTRY					
NAME	ORIGIN	LENGTH	NAME	LOC.	NAME	LOC.	NAME	LOC.
LINK	00	3E0	\$LINK	E	IBCCM#	88	DIOCS#	106
			\$BERR	1DA	IHCERRM	1E4	ERRMON	1EF
			ERRORX	202	IHCTRCH	20C	ERRTRA	216
			ERRSET	22A	ERRSAV	234	ERRSTR	23E
			IBERH#	252	EXIT	25C	\$BNAM	266
			BNAM	27A	RPAGE	284	WPAGE	28E
							INTSWCH	306
MAIN	3E0	100B8						
\$JCOMA	10498	150						
LCDINT *	105E8	11C						
			HCDINT	105F2	\$CDINT	105F8		
\$FLG06 *	10708	18						
\$RAJ *	10720	110						
\$YUDEX *	10830	150						
			\$YUDEXFL	10918				
\$YUD *	10980	242						
			\$YUD1	10980	\$YUD2	1098C	\$YUD3	10994
			\$YUD5	109A8	\$YUDR	10980	\$YUDC	1098C
\$YUD9998*	108C8	170						
			\$YUD98	108C8	\$YUD99	108DA		
\$55 *	10D38	2						
ENTRY ADDRESS	3E0							
TOTAL LENGTH	10D40							

Figure 3. CØG Link Edit Map

JOSHUA DPSTM-500	Title An Example - Part III	Vol. II	Page C.2.8
		Rev. 1	Date 3/73

region size is specified at about 100K greater than the module size, GØSIZE=170K. A system dump, DUMP='SYSØUT=A', has been requested in case of abnormal termination.

A GØ.SYSIN DD * card is required in this example since a monitor input card, SCRATCH=TST12791, is used. The monitor input card causes the system to scratch the data set named TST12791 if it exists. If it does not exist, the card is ignored. Even though DISP=TMP is specified for this job, the jobname could exist in the catalog if system failure occurred during execution of a previous job specifying the same jobname.

3. FØRTJCLP Procedure

```
//STEP1 EXEC FØRTJCLP,PDS='SRL.JCJ72791',  
// MEMBER=CØG,SYSLIB='SYS1.FØRTL18'  
//FØRT.SYSIN DD *  
    MØDULE  
    .  
    .  
    .  
    STØP  
    END  
/*
```

Figure 4. Use of FØRTJCLP Procedure

Figure 4 shows the use of the FØRTJCLP procedure to compile the CØG module and link edit to a private PDS. This procedure is identical to the FØRTJCLG procedure without the GØ step. In this example the module named CØG, MEMBER=CØG, will be placed in a private library, PDS='SRL.JCJ72791'. Also, the release 18 version of the FORTRAN library, SYSLIB='SYS1.FØRTL18', will be searched ahead of all other libraries during the link edit step.

JOSHUA DPSTM-500	Title	An Example - Part III	Vol. II	Page C.2.10
			Rev. 1	Date 3/73

(Blank)

4. FØRTJCL Procedure

```
//STEP1 EXEC FØRTJCL
//FØRT.SYSIN DD *
    MODULE
    :
    :
    STØP
    END
//LKED.SYSIN DD *
    ENTRY MAIN
    NAME CØG(R)
/*
```

Figure 5. Use of the FØRTJCL Procedure

Figure 5 shows the use of the FØRTJCL procedure to compile the CØG module and link edit into the public PDS,JØS.TESTLIB. This public library should only be used when more than one module from the same path is being tested. In this example an ENTRY and NAME card are used as link editor input. This is the only procedure where these cards may be used. In addition, the PDS and MEMBER symbolic parameters may not be used.

JOSHUA DPSTM-500	Title An Example - Part III	Vol. II	Page C.2.12
		Rev. 1	Date 3/73

(Blank)

JOSHUA DPSTM-500	Title An Example - Part III	Vol. II	Page C.2.13
		Rev. 2	Date 9/73

5. JOSHUA Procedure

```
//STEP1 EXEC JØSHUA, MEMBER=CØG, PDS='SRL.JCJ72791',
// USER=2816, JØB=TST22791, SPACE=3, DISP=NEW,
// SSPACE=0, GØSIZE=170K
//GØ.SYSIN DD *
CATEGORY=1
SCRATCH=TST12791
SCRATCH=TST22791
DATASET=3208,2791,2816
/*
```

Figure 6. Use of JOSHUA Procedure

Figure 6 shows the use of the JOSHUA procedure to execute the CØG module, MEMBER=CØG, from a private library, PDS='SRL.JCJ72791'. This procedure is identical to the GØ step of the FØRTJCLG procedure. This example differs from the GØ step in example No. 1 in the following ways:

- a) The module has previously been link edited into the private library, SRL.JCJ72791, with either the FØRTJCLP or FØRTJCLG procedure.
- b) The User and Job data set names have been changed.
- c) The Job data set will be retained, DISP=NEW, when the job terminates. Also, all unused space in the data set will be released due to the default of RLSE=YES.
- d) Two SCRATCH= cards are used.
- e) The normal data set search sequence is overridden with a DATASET= card. In this case only the data sets 3208, 2791, and 2816 will be searched in the specified order.
- f) The Job data set will be allocated to a physical data set that has a file category of one. Also, any protected data for which the user has authorization may be accessed. The user is determined from the badge number in the Job card, not the User symbolic parameter. This type of job must be submitted to Operations as a special handling job.

JOSHUA
DPSTM-500

Title Part D - Utility Programs

Vol.
II

Page
D.O.1

Rev.
1

Date
3/73

Part D Deleted

JOSHUA DPSTM-500	Title Part D - Utility Programs	Vol. II	Page D.O.2
		Rev. 1	Date 3/73

(Blank)

JOSHUA DPSTM-500	Title Part E - Monitor System (JMS)	Vol. II	Page E.0.1
		Rev. 1	Date 2/73

Contents

E.1 - Monitor System (JMS)

E.2 - An Example - Part IV

JOSHUA
DPSTM-500

Title

Part E - Monitor System (JMS)

Vol.

II

Page

E.O.2

Rev.

1

Date

2/73

(Blank)

Work Done By

J. E. Suich
R. L. Boyce
J. H. Maddox
R. C. Haywood
J. C. Jensen

Report Written By

J. C. Jensen

Contents

	<u>Page</u>
1. Introduction	E.1.3
1.1 Definitions	E.1.4
2. Terminal Monitor Operation	E.1.6
2.1 The IBM 2260 Terminal	E.1.6
2.2 The "Sign-On" Frame	E.1.8
2.3 The "Select" Frame	E.1.8
2.3.1 Data Set Usage	E.1.12
2.3.2 Monitor Functions	E.1.12
2.3.3 Function Arguments	E.1.13
2.3.4 Data Record Names	E.1.13
2.4 Output Displays	E.1.15
2.4.1 The Communications Data Set	E.1.15
3. Formatting Named Data Records	E.1.16
3.1 The Create Data Function	E.1.16
3.2 The Display Data Function	E.1.20
3.3 The Modify Data Function	E.1.21
4. Copying Named Data Records	E.1.33
4.1 The COPY Function	E.1.33
5. Executing Computational Modules	E.1.39
5.1 The Set Job Function	E.1.39
5.2 The EXECUTE Function	E.1.39
5.3 The STATUS Function	E.1.45
5.4 The View Queue Function	E.1.49

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.2
		Rev. 1	Date 2/73

Contents

	<u>Page</u>
6. Utility Functions	E.1.51
6.1 Display Catalog	E.1.51
6.2 LIST	E.1.53
6.3 List Brothers	E.1.66
6.4 RENAME	E.1.66
6.5 Set User	E.1.66
6.6 Set Write	E.1.70
6.7 PRINT	E.1.70
6.8 END	E.1.70
7. Using Monitor Procedures	E.1.74
7.1 The Execute Procedure Function	E.1.74
7.2 The SKIP Function	E.1.74
7.3 The Interrupt Procedure Function	E.1.76
7.4 The Restart Procedure Function	E.1.76
7.5 The Stop Procedure Function	E.1.76
8. Using JMS As A Batch Monitor	E.1.81
8.1 The Punch Data Function	E.1.83
8.2 The Execute Member Function	E.1.83
9. Maintaining The JOSHUA Data Base	E.1.85
9.1 The DELETE Function	E.1.86
9.2 The Delete Brothers Function	E.1.87
9.3 The MAINTAIN Function	E.1.87
9.4 The DUMP Function	E.1.92
9.5 The LOAD Function	E.1.92
9.6 The MERGE Function	E.1.93
9.7 The SCRATCH Function	E.1.93
10. Templates	E.1.95
10.1 The Create Template Function	E.1.95
10.2 The Modify Template Function	E.1.101
10.3 The Display Template Function	E.1.101
11. Creating Monitor Procedures	E.1.102
12. Data Protection Facilities	E.1.109
12.1 Data Retrieval	E.1.109
12.2 Explicit Requests	E.1.110
12.3 Data Generation	E.1.110
12.4 Template Protection	E.1.110
12.5 Changing the File Category	E.1.111
12.6 Determining the File Category	E.1.111
12.7 The EX Function	E.1.111
12.8 The PR Function	E.1.111
12.9 Data Set Maintenance	E.1.111

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.3
		Rev. 1	Date 2/73

1. Introduction

The principal function of the JOSHUA system is to facilitate execution of problems by preservation of conveniently reusable data. The JOSHUA monitor system, (JMS), provides the necessary facilities to manipulate the data and to initiate job execution from a visual display terminal. A typical procedure for execution of a problem would be:

1. Find a related problem among stored data.
2. Modify data as required for current problem.
3. Execute appropriate programs.
4. Display results.

The intelligence essential to following this procedure is:

1. What programs are available?
2. What data are available?
3. How data are labelled.
4. How to execute appropriate steps from a terminal.

The above procedure is accomplished through a series of "Function Requests" that are entered on the "select frame" when it appears on the screen of the visual display terminal. A function request invokes a particular JMS program that will accomplish a specific task such as displaying the data in a JOSHUA named data record. Function requests may require "arguments" such as the name of a record to be displayed or the name of a job that is to be executed. The following example shows a function request on a select frame to display the data in the record named "ASSEMBLY.MARK16.SIZE". The name of the function is "DD" (Display Data).

```
DD      --FUNCTION
ASSEMBLY.MARK16.SIZE.....
.....
```

This function request has one argument, a record name, that is entered directly below the function name. A function request to copy a record to another record would require two arguments.

```
CO      --FUNCTION
ASSEMBLY.MARK16.SIZE.....
ASSEMBLY.MARK17.SIZE.....
```

JMS may also be used in a batch environment by using a card reader and printer to replace a visual display terminal. Functions that require non JOSHUA data sets, have large core requirements, or that take longer than a few seconds to accomplish their task are restricted to the batch environment. The primary use of JMS in the batch mode is to execute computational modules and to maintain JOSHUA data sets.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.4
		Rev. 1	Date 2/7

The remainder of this part of Volume II discusses each of the available monitor functions along with other general information such as the operation of the terminals.

1.1 Definitions

The following definitions should be helpful to the understanding of the remaining sections of this document.

2260	IBM Visual Display Terminal
CURSOR	A small vertical line that appears on the screen of a 2260 that indicates the position on the screen in which the next entered character will appear.
SUBFUNCTION	A program used by JMS to accomplish a specific task.
FUNCTION	A collection of subfunctions used to perform a specific job.
ARGUMENT	Input data required by a function.
FUNCTION REQUEST	A combination of a function name and required arguments that are needed to accomplish a specific job.
MONITOR PROCEDURE	A collection of predefined function requests that may be used to accomplish a series of steps that are required to run a problem.
FRAME	The screen of a 2260, usually containing indications of where to enter required data.
SIGN-ON	The act of entering the required information on the screen of a 2260 that enables the use of JMS.
SIGN-OFF	The act of entering the function request "EN" that will "disconnect" a user from JMS and enable the use of the terminal by another user.
SESSION	The time between sign-on and sign-off.
SIGN-ON FRAME	The frame that indicates where to enter the information required to activate JMS for a particular user.
SELECT FRAME	The frame that indicates where to enter the function name and arguments for a function request.
EXECUTE FRAME	The frame that indicates where to enter the required information for running a job.
COMMUNICATIONS DATA SET	A temporary data set that contains the output of most function requests.

JOSHUA

DPSTM-500

Title

Monitor System (JMS)

Vol.

II

Page

E.1.5

Rev.

1

Date

2/73

PAGE

Ten lines of data that are stored in a record on the communications data set.

PAGING

The act of displaying pages from the communications data set.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.6
		Rev. 1	Date 2/73

2. Terminal Monitor Operation

2.1 The IBM 2260 Terminal

Input messages generated at the keyboard are displayed on the CRT as they are composed. The data is held in a buffer in the 2848 control unit until the user requests the data to be transmitted to the computer by depressing both the shift and the enter keys. Data transfer starts in position 1 of the buffer (upper left hand corner of the CRT) and continues to the end of message character produced by the enter request. All data to the right of a new line character on each line and data to the right of and below the end of message character will be blanked. The cursor that appears on the screen determines the position of data in the buffer. Some of the keys contain two different characters or functions. The characters or functions on the upper half of the key are obtained by depressing the shift key at the same time. When a character is entered it replaces the character that appeared to the right of the cursor.

Output messages are transmitted to the terminal as soon as the monitor can service the current request. These messages may or may not require a specific user response. If a response is not necessary, the shift/enter keys must be depressed to cause the next output message to be displayed, e.g. the next page of data. In this case the cursor is normally in the top left hand corner of the CRT and only one character, the end of message, is transmitted. However, the same result will occur wherever the cursor may be positioned.

The special keys have the following function.

Tab Key --

The cursor will be moved to the right of the next colon found in the buffer. If no colon appears between the cursor and the end of the buffer the cursor will be placed at Position 1.

Shift Key --

Characters and functions on the upper half of the keys will be active.

Advance Key --

The cursor will be advanced one position if the key is depressed lightly. It will be continuously advanced if the key is depressed strongly. No data will be erased.

Space/Erase Key --

The cursor will be advanced as above. All data will be erased in positions where the cursor has traveled.

Down Key --

The cursor will be moved one line down.

New Line Key --

A new line character will be inserted to the right of the cursor. The cursor will be moved one line down and to the extreme left. Data to the right of and including the new line character will be blanked on the line containing the character.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.7
		Rev. 2	Date 9/73

Up Key --

The cursor will be moved one line up.

Start Key --

A start manual input character (Start MI) will be placed to the right of the cursor if the character does not already appear on the CRT. When the shift and enter keys are depressed data will be transmitted starting to the right of the start MI character and proceeding to the cursor. If the start MI character already appears on the CRT, the data between it and the cursor will be blanked and the cursor will be placed in position 1. All JOSHUA displays contain a Start MI character in position 960 of the buffer (lower right hand corner). The use of this key will erase all data from position 1 to the cursor. Be careful.

Print Key --

The current display will be written on a 1053 typewriter if the control unit is so equipped. We do not have one. Use of this key causes the keyboard to "lock up". The erase display key must be used to unlock the keyboard. Be careful.

Enter Key --

Transmission of data takes place between the control unit buffer and the computer.

BKSP Key --

The cursor is moved back one position in the buffer. Data is not erased.

ERAS EOL Button --

Data is erased from the cursor to the end of the line that contains the cursor.

ERAS EOS Button --

Data is erased from the cursor to the end of the screen. Note that the Start MI character will be erased and must be replaced to enable data transmission.

ERAS DPLY Button --

The entire screen is erased.

On/Off Knob --

This knob is located on the right side of the terminal as you face the CRT. Pull it out to turn on the terminal. Twist the knob to vary the light intensity on the CRT.

Note:

A check character (■) is displayed if an error occurs in data transmission from the computer or the keyboard.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.8
		Rev. 2	Date 9/73

2.2. The "Sign-On" Frame

Figure 1 shows a "sign-on" frame which contains the information displayed on the terminal CRT when JMS is active.

The proper data must be entered in the first four fields starting at the left hand edge of the CRT. The last two fields need only be entered if protected data is to be processed. If a file ID is entered, the file category must be supplied. If proper data is not entered, the original sign-on frame will reappear with all blank fields. The tube number appearing at the top of the CRT is solely for identification purposes. The following are definitions of "proper" data for each field.

- BADGE - Four numeric characters that are the users badge number. e.g. 2791
- COST CODE - Any valid SRP or SRL cost code. e.g. 8690-00, 8211-1G, 641C-01
- USER DATA SET - Four numeric characters that are a user data set name. e.g. 1299
- USER ID - Six alphanumeric characters that have been assigned to the user by COD.
- FILE ID - Described in Section 12.
- FILE CATEGORY - Described in Section 12.

When proper data has been entered on the sign-on frame, a message will appear indicating the status of the system, or that the specified user data set may not be updated. Since a reply is not requested, depressing the shift/enter keys will cause the next output message, the "select" frame, to be displayed.

2.3 The "Select" Frame

The "select" frame is the main point of communication between the monitor and the user. Figure 2 shows the select frame as it might appear at various times during a terminal session. The top line is blank. The second line contains the data set names that are in the data set search sequence and the name of the data set to be updated. All function requests are made in the first eight positions of the third line. The fourth and fifth lines are used to transmit data, such as record names, to the various functions. The seventh line may contain a message such as seen in Figure 2d. The tube number at the top of the screen is for identification purposes only.

..... BADGE

- .. COST CODE

..... USER DATA SET

... USER ID

... FILE ID

..... FILE CATEGORY

TUBE 035

Figure 1 Sign-On Frame

TEMP0000 - 0000 - STD ---- 0000 TUBE 035
 --FUNCTION

a

TEMP0000 - 0000 - STD ---- 0000 TUBE 035
 DD --FUNCTION
 ASSEMBLY.MARK16.SIZE.....

b

Figure 2 Select Frames

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol. II	Page E.1.11
			Rev. 2	Date 9/73

TEMP0000 - 0000 - STD ---- 0000
TUBE 035

DD
--FUNCTION

3208.ASSEMBLY.MARK16.SIZE.....

.....

c

TEMP0000 - 0000 - STD ---- 0000
TUBE 035

DD
--FUNCTION

STD.ASSEMBLY.MARK16.SIZE.....

.....

THE SYSTEM WILL BE DOWN AT 1:00 PM FOR ABOUT 5 MINUTES

d

Figure 2 Select Frames

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.12
		Rev. 2	Date 9/73

2.3.1 Data Set Usage

The first three data set names appearing on the second line represent the data set search sequence. The monitor will search the specified data sets from left to right when retrieving a record when a specific data set has not been specified. The first name is the active job data set, and is set by the monitor at sign-on time to be TEMP followed by the users badge number. This may be changed with the Set Job function which is discussed later. The second name is the active User data set and is initially obtained from the USER DATA SET field of the sign-on frame. This name may be changed with the Set User function which is discussed later. The next name will always be STD which indicates that the standard data base is to be searched last. This field may not be altered.

The last name appearing on the line is the "write" data set. Only this data set and its associated catalogs may be updated. This is initially set from the USER DATA SET field of the sign-on frame and may be changed with the Set Write function which is discussed later. In addition to setting the "write" data set, the monitor verifies that the user is authorized to update the data set. This is determined from a named data record called SYSTEM.ACCESS.nnnn, where nnnn is the specified User data set name. This record contains a list of badge numbers that are authorized to update the data set. If the record does not exist, the user is assumed to be authorized. The monitor also insures that the "write" data set may not be updated concurrently by the monitor system and a batch job or by more than one batch job. The "write" data set may be updated by more than one user from the monitor system.

2.3.2 Monitor Functions

The JOSHUA Monitor System is composed of a set of "FUNCTIONS" that are used to perform specific tasks such as displaying a named data record. Each function consists of two or more subfunctions. In the case of displaying a named data record, the first subfunction reads the binary record and writes a formatted record on a temporary data set. The second subfunction then reads the temporary data set and displays the records at the terminal. The combination of a function name and one or more input arguments is called a function request. The function name is entered on the third line of the select frame opposite the word FUNCTION. Input data arguments such as data set or data record names are entered on the following two lines. (See Figure 2)

Function names may be either two words or one word of eight characters or less. If the name is composed of two words the first character of each word is used in making a function request. e.g. Display Data - DD. If the function name is one word, either the first two characters of the word or the complete word may be used in making a request. e.g. COPY - CO or COPY.

Table I contains a complete list of function names for the monitor system. Some of the functions are only operational in one "mode", i.e. either the terminal system or the batch monitor. In general, this is because they require resources not allocated to the other mode of operation, e.g. tape units are not allocated to the terminal region. In other cases the functions are highly interactive and therefore not practical to operate in the batch monitor. The use of each JMS function is described in subsequent sections of this part of the Users guide. Table I contains page references for each function.

2.3.3 Function Arguments

Function arguments are composed of a series of alphanumeric qualifiers that are separated by periods. Qualifiers may not be more than eight characters in length and may not contain imbedded blanks. However, blank characters may be used on the right hand side of a qualifier if it is less than eight characters in length. Up to fifteen qualifiers may be entered on each of the two lines below the function name. See Figure 2 for examples of function arguments.

The monitor terminates the scan of an argument line when it encounters two consecutive periods. However, it saves all the data entered on each argument line and restores the data when the select frame reappears. This feature in conjunction with the use of eight character left adjusted qualifiers can save a considerable amount of data entry time.

2.3.4 Data Record Names

Data record names are passed as arguments to many of the JMS functions. These names are composed of a series of qualifiers separated by periods as described in Part A of this volume. e.g. ASSEMBLY.MK5R.COLD.GEOMETRY.ANNULAR. The monitor system also permits partial names as arguments to most functions. A partial name indicates that the function request is for an entire subtree of a data set, i.e. if the name ASSEMBLY.MK5R were specified, it would indicate a request for all ASSEMBLY records for the MK5R assembly from the first data set in the search sequence that contains at least one record of the requested subtree.

Record names may also indicate the name of a specific User or Job data set, or a request may be made for Standard data only. If a User or Job data set name is appended to the front of the record name, only the specified data set will be searched. Figure 2 contains an example of an ordinary request for an ASSEMBLY record followed by a specific request for the record from a User data set. If the mnemonic "STD" is appended to the name, only the specified Standard data set will be searched. Note that in the example in Figure 2, a blank is inserted after STD to prevent reentering the entire name.

TABLE I

JMS FUNCTIONS

FUNCTION NAME	MNEMONIC	MODE **	PAGE
COPY	CØ	BT **	E.1.33
Create Data	CD	BT **	E.1.16
Create Freeform	CF	T	
Create Template	CT	T	E.1.95
Display Catalog	DC	BT	E.1.51
Display Data	DD	BT	E.1.20
Display Freeform	DF	T	
Delete Brothers	DB	T	E.1.87
DELETE	DE	BT **	E.1.86
Display Template	DT	T	E.1.101
DUMP	DU	B **	E.1.92
EDIT	ED	T	
Execute Member	EM	B **	E.1.83
END	EN	T	E.1.70
Execute Procedure	EP	T	E.1.74
EXECUTE	EX	T	E.1.39
Face Map	FM	T	F.1.1
Interrupt Procedure	IP	T	E.1.76
List Brothers	LB	BT	E.1.66
LIST	LI	BT	E.1.53
LOAD	LØ	B *	E.1.92
Least Squares	LS	BT	F.3.1
MAINTAIN	MA	B *	E.1.87
Modify Data	MD	BT *	E.1.21
MERGE	ME	B *	E.1.93
Modify Template	MT	T	E.1.101
Punch Data	PD	B	E.1.83
PRINT	PR	T	E.1.70
RENAME	RE	BT *	E.1.66
Reactor Map	RM	T	F.2.1
Restart Procedure	RP	T	E.1.76
SCRATCH	SC	BT *	E.1.93
Set Job	SJ	BT	E.1.39
Stop Procedure	SP	T	E.1.76
STATUS	ST	T	E.1.45
Set User	SU	BT	E.1.66
Set Write	SW	BT	E.1.70
View Queue	VQ	T	E.1.49

* Batch monitor terminates on error conditions
 ** B-Batch, T-Terminal

2.4 Output Displays

The output messages from a function may be placed in a display buffer for a direct display on the CRT or they may be written to a "communications" data set for subsequent display. Function output messages are always displayed on lines two thru eleven of the CRT leaving the top and bottom lines for monitor system use. The direct display method therefore may only be used when there are ten or fewer lines of output from a subfunction.

The top and bottom lines of the display are used for various purposes by different monitor subfunctions and their use is described with each function. However, there are two control characters, "S" and "P", that may be entered on any output display. The "S" causes the monitor system to return immediately to the select frame, skipping all other subfunctions in the current function request. The "P" causes the current display to be saved on a temporary data set that will subsequently be printed. These control characters may be entered in either the top or bottom left hand corners of the display. However, the cursor must be placed below or to the right of any data to be saved for printing.

2.4.1 The Communications Data Set

Subfunctions that produce more than ten lines of output write the output to the communications data set. Each record of this data set will contain ten lines of formatted data referred to as a page. After a subfunction that writes to the communications data set has completed, the first page of data is automatically displayed at the terminal. Each time the shift/enter keys are now depressed the next sequential page of the data set will be displayed and the page number will appear on the left side of the bottom line (this operation is referred to as paging). When the last page is reached, the word LASTPAGE will appear next to the page number on the bottom line. If the shift/enter keys are depressed at this point, the next subfunction will be executed, or, if there are no more subfunctions in the list, the select frame will reappear.

The first four characters on the top and bottom lines of the screen are used as control fields. Data that is entered in these fields must be left adjusted. The communications data set may be viewed in a random order by supplying a page number in either control field. If the requested page number is beyond the last page of data, the last page will be displayed with its true page number and the word LASTPAGE appearing on the bottom line. A control character, "E", may also be used in either control field. This will cause the paging process to be terminated and the next subfunction to be executed.

The communication data set may also be updated during the execution of certain functions. The details of updating are discussed in the section dealing with creating and modifying named data records.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.16
		Rev. 2	Date 9/73

3. Formatting Named Data Records

Data is stored in the JOSHUA data base in a binary form and is cataloged as named data records. The monitor system converts the data to and from an understandable terminal display by means of a "template". A template contains data that describes the structure of the data record and hollerith text that describes the meaning of the various data fields.

Templates are stored in the JOSHUA data base as named data records, the first qualifier being the word TEMPLATE. There is one template for each record type that may be displayed at a terminal. A record type is defined by the name and position of the fixed qualifiers in a record name. Variable qualifiers are represented by question marks in the template name. e.g.

Record name INPUT.GAUGE.CASE.TEST01
Template name TEMPLATE.INPUT.GAUGE.CASE.?

3.1 The Create Data Function (CD)

The CD function is used to create a new data record in the JOSHUA data base. The argument is the complete name of a data record. The function first creates a display of the template for the specified record type. Data fields are identified on the display with a colon followed by one or more field descriptors. The following descriptors are used.

I - Integer
E - Floating point
F - Fixed point
A - Alphanumeric
Z - Hexadecimal
n - Index field ($1 < n < 7$)

Field identifiers for floating and fixed point variables will also contain a decimal point. The numeric index field does not represent a data value in the record, but a value of an index used in an implied DO loop in the I/O list.

Figure 3 shows the initial step of a CD operation for a record named SAMPLE.CASE1. The template name is TEMPLATE.SAMPLE.?. The top frame shows the function request which consists of the function name and the name of the record to be created. Note that the record will be written to data set 0000 as indicated by the "write" data set field at the top of the select frame. (See Section 2.3.1, Data Set Usage.)

The bottom frame shows the template display that results from the function request. The top line contains the name of the template. Note that the User data set name, 0000, has been appended to the front of the name indicating that the template was retrieved from data set 0000 and not the

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.17
		Rev. 1	Date 2/73

```

TEMP0000 - 0000 - STD ---- 0000          TUBE 035
CD      --FUNCTION
SAMPLE.CASE1.....
.....

```

```

0000.TEMPLATE.SAMPLE.CASE1
:II INTEGER FIELD - ENTER NUMBER OF PAGES HERE
INDEX FIELD - PAGE :22
:E.EEEEEEE - FLOATING POINT FIELD
:FF.FFF      - FIXED POINT FIELD
:AAAA        - ALPHANUMERIC FIELD
:ZZZZZZZZ    - HEXADECIMAL FIELD

```

LASTPAGE
1

Figure 3

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.18
		Rev. 1	Date 2/73

Standard template data set. The question mark in the template name has also been replaced by the specified variable qualifier, CASE1. Data should be entered on the remaining portion of the frame by overlaying the indicated data fields with the desired values. Leading and trailing field descriptors need not be blanked. Data is not placed in the index field since this does not represent a data value.

When the shift/enter keys are depressed the communications data set will be updated with the entered values, and since this is the last page, the next subfunction will be executed. The data will be converted to a binary form, written to data set 0000, and cataloged as a named data record. The next subfunction then retrieves the data and uses the template to produce a formatted display of the record on the communications data set. The first page of the data set, and in this case the only page, will then be automatically displayed. Figure 4 shows data as entered at the top of the page and the resulting display at the bottom. Note that the record name now appears on the top line of the display.

The data fields appearing on this display may be modified at this point if the control character, "R", is placed in one of the control fields and the shift/enter keys are depressed. This causes the system to change from a "display" mode to an "update" mode. After modifications have been made and the shift/enter keys again depressed, the communications data set will be updated and the next subfunction will be executed as before. However, the record will not be reformatted for display and the system will return to the select frame. Further details concerning "update" mode operation may be found in Section 3.3, the Modify Data Function.

The previous discussion has been concerned with creating a data record containing just one page of data. There are two ways that a data record may consist of more than one page of data. First, the template may contain more than one page type. In this case there will be one page on the communications data set for each page type contained in the template. There also may be multiple pages of a page type if it describes a variable length list. In this case, the user must add page types where necessary to the communication data set by using the control character, "A", in the lower left control field of the screen. Each time the shift/enter keys are depressed, a page of the type currently displayed will be added immediately following the current page. After the required number of pages have been added, they may be updated in any order. Note that as soon as the last page is updated the paging process will be terminated and the record will be created. Until that time the pages may be updated or just viewed in any order.

Usually the number of pages required will be self evident from the explanations given in the template text as in the example shown in Figures 3 and 4. Also, the integer variable that controls the number of pages will, in this case, appear on each of the pages of the display. This variable must be entered on the first page and need not be entered on succeeding pages. If it is unclear from the template text how many pages will be required, fill in the specified fields and let the system create the record with blank data. The subsequent display will contain the proper number of pages and the correct data may be entered after resetting the paging mode to update with an "R" control character.

0000.TEMPLATE.SAMPLE.CASE1

:II INTEGER FIELD - ENTER NUMBER OF PAGES HERE

INDEX FIELD - PAGE :22

:1.5E-6EEEE - FLOATING POINT FIELD

:F1.5FF - FIXED POINT FIELD

:BC12 - ALPHANUMERIC FIELD

:C2C3F1F2 - HEXADECIMAL FIELD

LASTPAGE 1

0000.SAMPLE.CASE1

: 1 INTEGER FIELD - ENTER NUMBER OF PAGES HERE

INDEX FIELD - PAGE : 1

:1.5000E-06 - FLOATING POINT FIELD

: 1.500 - FIXED POINT FIELD

:BC12 - ALPHANUMERIC FIELD

:C2C3F1F2 - HEXADECIMAL FIELD

LASTPAGE 1

Figure 4

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.20
		Rev. 1	Date 2/7

3.2 The Display Data Function (DD)

The DD function is used to display and modify existing data records in the JOSHUA data base. The argument may be either a complete record name or the name of a subtree of a data set. e.g. the record described in the last section may be displayed with either of the following requests.

```
DD      -- FUNCTION                      (complete record name)
SAMPLE.CASE1.....
.....
```

or

```
DD      -- FUNCTION                      (name of a subtree)
SAMPLE.....
.....
```

In the latter case, if more than one record existed than the SAMPLE subtree, such as SAMPLE.CASE2, SAMPLE.CASE3, etc., all the records would be displayed from the first data set in the search sequence that contained a SAMPLE subtree. The complete name of each record that is displayed will appear on the top line of the screen. The name of the data set from which it was retrieved will be appended to the front of the name. e.g. 0000.SAMPLE.CASE1.

Each record type encountered in a subtree must have a corresponding template or the function request will be terminated. When a record is retrieved and a template exists for that record type, the formatted data is written to the communications data set and the first page of data is automatically displayed. The communications data set may then be viewed as described in Section 2.4, Output Displays. When the shift/enter keys are depressed on the last page of data without using a control character, the DD function will retrieve the next record of the subtree and repeat the above process. Also, an "E" control character may be used in either control field on any page to terminate the current display and cause the next record of the subtree to be displayed. When the end of the subtree is encountered, or an "S" control character is detected in either control field of any page, the select frame will return to the screen.

Since the format of a display depends entirely on the template and not on the data record, the format may be easily changed by modifying the existing template. (See Section 10, Templates.) The modified template would then reside in the users data set, thus only changing the format for one user. Format changes may be desirable in order to reduce the number of pages in a display by displaying more values per page, display the data values in hexadecimal for debugging purposes, or to provide more suitable text.

Any record of the subtree may be modified by using the "R" control character in either control field during the paging process. This causes the system to change from a "display" mode to an "update" mode for the named record shown at the top of the screen. After the communications data set has been updated and either an "E" control character or the last page is encountered, the record will be modified in the JOSHUA data base and the next record of the subtree will be displayed. Additional records will only be modified

Title	Monitor System (JMS)		Vol. II	Page E.1.21
			Rev. 1	Date 2/73

when another "R" control character is encountered. Further information concerning the "update" mode may be found in Section 3.3, the Modify Data Function.

The DD function may also be used with a blank argument to display the current contents of the communications data set. This is useful in cases where you may have accidentally caused the system to return to the select frame or when error conditions occur during a subfunction that has written some output to the data set before the error occurred. e.g. the message "Page Limit Exceeded" may appear following a DD request for a long record. In order to view the data already written to the communications data set you must return to the select frame and use the DD function with a blank argument.

3.3 The Modify Data Function (MD)

The MD function is used to modify existing records in the JOSHUA data base. Either one or two arguments may be used consisting of complete record names or subtree names. When two arguments are used the data is retrieved from the record or subtree specified by the first argument and written to the record or subtree specified by the second argument. e.g.

```
MD          -- FUNCTION
SAMPLE.CASE1.....(1st Argument)
SAMPLE.CASE2.....(2nd Argument)
```

The MD function is similar to the DD function except that it always operates in the "update" mode. When paging in the update mode, the system will rewrite the current page on the communications data set if no control information other than an "E" is supplied, and the cursor is positioned below the first line of the screen when the shift/enter keys are depressed. The next sequential page is then displayed. When the cursor is left on the top line of the screen the data pages may be randomly viewed with no updating occurring. Each time a modification is made, the cursor should be placed on the bottom line of the screen to avoid the inadvertent blanking of subsequent data fields. If the shift/enter keys are depressed on the last page, or an "E" is encountered in a control field, the page will be updated and the modified record will be written to the JOSHUA data base. The next record of the subtree will then be formatted and written to the communications data set so that modifications may be made. The select frame will return when the end of the subtree is encountered.

The previous discussion has considered only those cases where the length of the record did not have to be changed. Pages of data may be added to or deleted from the communications data set thru the use of the "A" or "D" control characters. These characters may be placed in either control field. When using the "A" character the cursor must be below the top line of the screen for it to be effective. The appearance of an "A" in a control field indicates that the current page is to be added to the communications data set immediately following the current page. The appearance of a "D" in a control field indicates that the current page is to be removed from the communications data set. In addition, a page number may be placed immediately following either character. e.g. "A4" would mean to add the current page immediately after page 4 on the communications data

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.22
		Rev. 1	Date 2/73

set; "D4" would mean to remove page 4 from the communications data set - the current page has no meaning in this case.

Figures 5 thru 11 illustrate the use of the MD function. The top half of each figure represents an action by the user and the bottom half is the system response to that request.

Figure 5 - this shows an MD function request indicating that the data in the record named MODIFY.SAMPLE.RECORD1 is to be modified, written to the JOSHUA data base, and cataloged by the name MODIFY.SAMPLE.RECORD1. The first page of the data on the communications data set is then displayed. The record name on the top line of the screen indicates from which data set the record was retrieved. In this example, the number of values in the integer array will be changed from 6 to 3 and all values will be modified.

Figure 6 - The first page of data is modified as indicated by overlaying the previous data fields. The field containing the integer 4 need not be blanked and will be ignored when the record is created. The next page of the communications data set is then displayed.

Figure 7 - This page of data is no longer appropriate since the number of integer values was decreased to three. The page is deleted by using the "D" control character in either control field. The resulting display is the next sequential page of the communications data set. It has been renumbered to account for the deleted page. In this example the number of values in the floating point array will be increased from three to five and all values will be modified.

Figure 8 - The template text indicates that five values will require two pages of this type. A page is added by placing the control character, "A", in the lower left hand corner of the screen and depressing the shift/enter keys. The "A" may be entered on the top line if the cursor is moved below the top line. Remember that all data to the right and below the cursor will be blanked, e.g. if the cursor is in the first position of the second line a blank page would be added. The result of the add operation is a display of the added page.

Figure 9 - Since page two could not be updated at the same time the third page was added, the next step is to request to see page two again. This is done by entering a 2 in either control field.

Figure 10 - The data fields on page two are overlayed with new values. When the data is entered, page two of the communications data set is updated and page three is displayed.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.23
		Rev. 1	Date 2/73

Figure 11 - The additional two data values are entered on page three by overlaying the next two sequential value fields for the floating point array. The number of values will be displayed on each page of this type but its true value will be obtained from the *first* page of this type when data is modified. Additional value fields, i.e. the number 3.0000E 00 in this case, may also be ignored. When the data is entered, page three of the communications data set will be updated and since this is the last page, the data record will be created in the JOSHUA data base and the resultant display is the original select frame.

Figures 12 and 13 show a Display Data (DD) function request and its resultant display for the newly created record named MODIFY.SAMPLE.RECORD2.

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol.	Page
			II	E.1.24
			Rev.	Date
			1	2/7

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

MD --FUNCTION

MODIFY.SAMPLE.RECORD1.....

MODIFY.SAMPLE.RECORD2.....

0000.MODIFY.SAMPLE.RECORD1

: 6 NUMBER OF INTEGER VALUES

THIS IS PAGE TYPE ONE FOR THIS RECORD. IT CONTAINS A VARIABLE
LENGTH INTEGER ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF
THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF FOUR VALUES.

: 1

: 2

: 3

: 4

PAGE 1

Figure 5

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.25
		Rev. 1	Date 2/73

```

0000.MODIFY.SAMPLE.RECORD1
: 3  NUMBER OF INTEGER VALUES

```

THIS IS PAGE TYPE ONE FOR THIS RECORD. IT CONTAINS A VARIABLE LENGTH INTEGER ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF FOUR VALUES.

```

: 11
: 22
: 33
: 4

```

PAGE 1

```

0000.MODIFY.SAMPLE.RECORD1
: 6  NUMBER OF INTEGER VALUES

```

THIS IS PAGE TYPE ONE FOR THIS RECORD. IT CONTAINS A VARIABLE LENGTH INTEGER ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF FOUR VALUES.

```

: 5
: 6

```

PAGE 2

Figure 6

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol.	Page
		II	E.1.26
		Rev.	Date
		1	2/73

D 0000.MODIFY.SAMPLE.RECORD1
: 6 NUMBER OF INTEGER VALUES

THIS IS PAGE ONE FOR THIS RECORD. IT CONTAINS A VARIABLE
LENGTH INTEGER ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF
THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF FOUR VALUES.

: 5
: 6

PAGE 2

0000.MODIFY.SAMPLE.RECORD1
: 3 NUMBER OF FLOATING POINT VALUES

THIS IS PAGE TYPE TWO FOR THIS RECORD. IT CONTAINS A VARIABLE
LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT. ANOTHER
PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE
VALUES.

:1.0000E 00
:2.0000E 00
:3.0000E 00

LASTPAGE 2

Figure 7

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.27
		Rev. 1	Date 2/73

0000.MODIFY.SAMPLE.RECORD1

: 3 NUMBER OF FLOATING POINT VALUES

THIS IS PAGE TYPE TWO FOR THIS RECORD. IT CONTAINS A VARIABLE LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE VALUES.

:1.0000E 00

:2.0000E 00

:3.0000E 00

A LASTPAGE 2

0000.MODIFY.SAMPLE.RECORD1

: 3 NUMBER OF FLOATING POINT VALUES

THIS IS PAGE TYPE TWO FOR THIS RECORD. IT CONTAINS A VARIABLE LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE VALUES.

:1.0000E 00

:2.0000E 00

:3.0000E 00

LASTPAGE 3

Figure 8

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.28
		Rev. 1	Date 2/73

```

2  0000.MODIFY.SAMPLE.RECORD1
:  3  NUMBER OF FLOATING POINT VALUES

```

THIS IS PAGE TYPE TWO FOR THIS RECORD. IT CONTAINS A VARIABLE LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE VALUES.

```

:1.0000E 00
:2.0000E 00
:3.0000E 00

```

LASTPAGE 3

```

0000.MODIFY.SAMPLE.RECORD1
:  3  NUMBER OF FLOATING POINT VALUES

```

THIS IS PAGE TYPE TWO FOR THIS RECORD. IT CONTAINS A VARIABLE LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE VALUES.

```

:1.0000E 00
:2.0000E 00
:3.0000E 00

```

PAGE 2

Figure 9

0000.MODIFY.SAMPLE.RECORD1

: 5 NUMBER OF FLOATING POINT VALUES

THIS IS PAGE TYPE TWO FOR THIS RECORD. IT CONTAINS A VARIABLE LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE VALUES.

:1.1000E 00

:2.2000E 00

:3.3000E 00

PAGE 2

0000.MODIFY.SAMPLE.RECORD1

: 3 NUMBER OF FLOATING POINT VALUES

THIS IS PAGE TYPE TWO FOR THIS RECORD. IT CONTAINS A VARIABLE LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE VALUES.

:1.0000E 00

:2.0000E 00

:3.0000E 00

LASTPAGE 3

Figure 10

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.30
		Rev. 1	Date 2/73

```

0000.MODIFY.SAMPLE.RECORD1
: 3  NUMBER OF FLOATING POINT VALUES

THIS IS PAGE TYPE TWO FOR THIS RECORD.  IT CONTAINS A VARIABLE
LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT.  ANOTHER
PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE
VALUES.

:4.4000E 00
:5.5000E 00
:3.0000E 00

LASTPAGE      3

```

```

TEMP0000 - 0000 - STD ---- 0000      TUBE 035
MD      --FUNCTION
MODIFY.SAMPLE.RECORD1.....
MODIFY.SAMPLE.RECORD2.....

```

Figure 11

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.31
		Rev. 1	Date 2/73

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

DD --FUNCTION

MODIFY.SAMPLE.RECORD2.....

.....

0000.MODIFY.SAMPLE.RECORD2

: 3 NUMBER OF INTEGER VALUES

THIS IS PAGE TYPE ONE FOR THIS RECORD. IT CONTAINS A VARIABLE
LENGTH INTEGER ARRAY IN ADDITION TO THIS TEXT. ANOTHER PAGE OF
THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF FOUR VALUES.

: 11

: 22

: 33

PAGE

1

Figure 12

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.32
		Rev. 1	Date 2/73

```

0000.MODIFY.SAMPLE.RECORD2
: 5  NUMBER OF FLOATING POINT VALUES

THIS IS PAGE TYPE TWO FOR THIS RECORD.  IT CONTAINS A VARIABLE
LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT.  ANOTHER
PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE
VALUES.

:1.1000E 00
:2.2000E 00
:3.3000E 00

PAGE          2

```

```

0000.MODIFY.SAMPLE.RECORD2
: 5  NUMBER OF FLOATING POINT VALUES

THIS IS PAGE TYPE TWO FOR THIS RECORD.  IT CONTAINS A VARIABLE
LENGTH FLOATING POINT ARRAY IN ADDITION TO THIS TEXT.  ANOTHER
PAGE OF THIS TYPE WILL BE REQUIRED FOR EACH MULTIPLE OF THREE
VALUES.

:4.4000E 00
:5.5000E 00

LASTPAGE     3

```

Figure 13

4. Copying Named Data Records

4.1 The COPY Function (CO)

The CO function is used to copy data from one named data record to another named data record. The data may reside in the same or different data sets. The function requires one or two arguments. The first argument is the name of the record to be copied and is referred to as the "old name". The second argument is optional and is the "new name" of the record. When the second argument is omitted the new name is set to the old name. This form of the CO function is used to transfer data from one data set to another data set while retaining the same name. All data that is copied will be written to the "write" data set as shown at the top of the select frame. If a data record of the same name already exists in the "write" data set, it will be overwritten.

The arguments may also be partial names, or subtree names. The additional qualifiers for each name of the "old subtree" will be added to the "new subtree" name to form the "new name" for each record. The "new subtree" name may also be omitted as in the case with complete record names. In the terminal region a message is issued after each record of a subtree has been copied indicating the complete names of each record. The shift/enter keys must then be depressed to copy the next record. The operation may be stopped at any time by entering an "S" in either control field. When the CO function is used in the batch region, the entire subtree is copied with no intervening messages. The batch monitor should always be used to copy large subtrees, such as entire Job data sets.

Figures 14 thru 17 illustrate the use of the CO function. The top half of each figure represents an action by the user and the bottom half is the system response to that request.

Figure 14 - This CO request uses one complete record name as an argument. The function locates the record in the Job data set, JOB10000, generates a new name identical to the old name, and writes the record to the "write" data set, 0000. Note that the name of the data set in which the record is located is appended to the front of the record name on the output message. If the record had been located in the Standard data set the name would appear as it does on the select frame.

Figure 15 - This CO request uses two complete names as arguments. The second name is necessary in this case in order to change the qualifier, MK16, to MK17 in the new name. The record was located in the Job data set and written to the "write" data set with a different name.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.34
		Rev. 1	Date 2/73

Figure 16 - This CO request uses two subtree names as arguments. The second subtree name is necessary in order to change the qualifier, MK16, to MK17 in the new name. The additional qualifiers, GEOMETRY and CASE1, that were located in the old subtree are added to the new subtree name to form the new record name. The data was located in the Job data set and written to the "write" data set with a different name. When the shift/enter keys are depressed in this case, the next record of the subtree will be copied. Note that once one record of a subtree is located in the Job data set, the User and Standard data sets will not be searched for additional records. Only the records from the first data set in the search sequence that contains a subtree of the specified name will be copied.

Figure 17 - This CO request uses one subtree name as an argument. This form of the CO request is normally used to copy a small subtree from a Job data set to a user data set. The same form may be used with the batch monitor to copy large subtrees such as entire Job data sets.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol.	Page
		II	E.1.35
		Rev.	Date
		1	2/73

JOB10000 - 0000 - STD ---- 0000 TUBE 035

CO --FUNCTION

LATTICE.MK16.GEOMETRY.CASE1.....

.....

THE RECORD NAMED

JOB10000.LATTICE.MK16.GEOMETRY.CASE1

HAS BEEN COPIED TO

0000.LATTICE.MK16.GEOMETRY.CASE1

REQUESTED RECORDS HAVE BEEN COPIED

Figure 14

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.36
		Rev. 1	Date 2/73

JOB10000 - 0000 - STD ----- 0000 TUBE 035

CO --FUNCTION

LATTICE.MK16.GEOMETRY.CASE1.....

LATTICE.MK17.GEOMETRY.CASE1.....

THE RECORD NAMED

JOB10000.LATTICE.MK16.GEOMETRY.CASE1

HAS BEEN COPIED TO

0000.LATTICE.MK17.GEOMETRY.CASE1

REQUESTED RECORDS HAVE BEEN COPIED

Figure 15

JOB10000 - 0000 - STD ---- 0000 TUBE 035

CO --FUNCTION

LATTICE.MK16.....

LATTICE.MK17.....

THE RECORD NAMED

JOB10000.LATTICE.MK16.GEOMETRY.CASE1

HAS BEEN COPIED TO

0000.LATTICE.MK17.GEOMETRY.CASE1

TO COPY NEXT RECORD HIT ENTER

Figure 16

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol. II	Page E.1.38
			Rev. 1	Date 2/73

JOB10000 - 0000 - STD ----- 0000 TUBE 035
CO --FUNCTION
LATTICE.....
.....

THE RECORD NAMED
JOB10000.LATTICE.MK16.GEOMETRY.CASE1
HAS BEEN COPIED TO
0000.LATTICE.MK16.GEOMETRY.CASE1

TO COPY NEXT RECORD HIT ENTER

Figure 17

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.39
		Rev. 2	Date 9/73

5. Executing Computational Modules

Computational modules are usually too large and run too long to enable execution in the terminal region. These jobs must therefore be run as batch "background" jobs. JMS provides the facilities to submit background jobs directly from a terminal. In addition, utilities are provided that enable the user to check for duplicate job names, determine the status of background jobs, and to view the JCL for jobs in the JOSHUA hold queue.

5.1 The Set Job Function (SJ)

Set Job is a multipurpose function that will do the following three things.

- 1) Establish the active job name. This name will be the first name in the data set search sequence and will appear at the top of the select frame on the left hand side of the screen. The job name is initially set to TEMPnnnn where nnnn is the users badge number.
- 2) Checks for a duplicate job name in the JOSHUA catalog. The resultant message of a SJ request indicates whether or not the specified job name could be located in the JOSHUA catalog.
- 3) If the job name is located in the JOSHUA catalog, the current catalog data is read from disk into core storage. This is necessary since a batch background job may update the catalog after it has been read into the terminal region.

The function requires one argument, a job name. Job names must begin with an alphabetic character, be eight characters in length, and the last four characters must be numeric, e.g. JOB10000. The four numeric characters should be set to the users badge number whenever possible. This will provide an easy way to determine all the job names for a given user via the DC function which is discussed later. Figure 18 shows a SJ function request and its resultant message.

5.2 The EXECUTE Function (EX)

The EX function is used to enter a background batch job that either executes a computational module or performs monitor functions in the batch mode. See Section 8, "Using JMS as a Batch Monitor", for further details. There are no arguments to the EX function unless it is contained within a JOSHUA procedure. See Section 11, "Creating Monitor Procedures", for more details. Figure 19 shows an EX function request. The resultant display is referred to as the "execute frame".

The parameters on the execute frame are used to build the job control language, (JCL), necessary to execute a batch job. The parameters are defined below. Part C, "Module Preparation and Execution", contains additional information concerning these parameters.

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol.	Page .
			II	E.1.40
			Rev.	Date
			2	9/77

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

SJ --FUNCTION

RUNA0000.....

.....

RUNA0000-IS NOW THE ACTIVE JOB NAME TUBE 035

JOB DATA SET WAS NOT LOCATED - ASSUME NEW JOB

Figure 18

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol.	Page
		II	E.1.41
		Rev.	Date
		I	2/73

RUNA0000 - 0000 - STD ---- 0000
TUBE 035

EX
--FUNCTION

:L000.....BOX
:001...EST TIME
:01...EST LINES
JOB PARAMETERS

:
.....MEMBER
:01....SPACE
:001..GTIME
EXEC PARAMETERS

:JOS.PGMSO1 ..PDS
:140K..GOSIZE
:NEW..DISP

:JOSHUAPROC
:YES...RLSE
:NO...DUMP

:OFF.....TIMER
:010...BUFNO
:001..SSPACE

:RUNA0000.....JOB

:NO.....TAPES
:NO....INPUT

:
...:
...:
...:
.. DATA SETS

:
.....SCRATCH

Figure 19

JOSHUA

DPSTM-500

Title Monitor System (JMS)

Vol. II	Page E.1.42
Rev. 1	Date 2/73

- BOX - Destination of printed output.
- EST TIME - Estimated elapsed time that the job will occupy core storage. (Minutes) The operator has the option to cancel the job if it runs longer than this time.
- EST LINES - Estimated number of thousands of lines of printed output.
- MEMBER - Name of the computational module to be executed.
- SPACE - Number of blocks of Job data set space that will be required. One block of space contains 100 physical data records of 100 words each in addition to 2 tree catalog records.
- GTIME - Estimated CPU time for the job step. The job will be automatically terminated when the elapsed CPU time reaches this value.
- PDS - Name of the library that contains the module to be executed.
- GOSIZE - Amount of core storage necessary for job execution.
- DISP - Disposition of the Job data set. The values may be NEW, TMP, or OLD.
- NEW - A Job data set is to be allocated for this job. It will be retained at job termination. The expiration date will appear in the catalog display at the end of the printed output.
- TMP - A Job data set is to be allocated for this job. It will not be retained at job termination.
- OLD - The specified Job data set already exists. The SPACE parameter will be ignored and the expiration date will be updated as if it were a NEW data set. If the data set cannot be located, the disposition will be changed to NEW.
- PROC - Name of the cataloged procedure to be executed.
- RLSE - Job data set space release parameter. The values may be YES or NO. YES means to release all *unused* space in the Job data set at job termination.
- DUMP - Indication of whether or not a core storage dump is desired if the job terminates abnormally; may be YES or NO.
- TIMER - Indication of whether or not program timings are desired; may be OFF or ON.

- BUFNO - Number of 784 byte buffers allowed for the data management routine, BNAM. The minimum value of this parameter is four.
- SSPACE - Number of blocks of Scratch data set space that will be required.
- JOB - Job name.
- TAPES - Indication of whether or not one or more tapes will be required for job execution; may be YES or NO.
- INPUT - Indications of whether or not input data will be entered with the job; may be YES or NO.
- DATA SETS - Data set search sequence. If one or more data set names are entered they will override the Standard data set search sequence.
- SCRATCH - Name of a Job data set that will be scratched prior to Job data set allocation.

The result of entering the data on the execute frame will vary depending on the values of some of the parameters that were entered. The following cases may be encountered.

- 1) The TAPES and INPUT parameters are specified as NO. The EST TIME, EST LINES, and GOSIZE all fall within the "express" category. (Since the express category criteria are subject to change, their current values should be obtained from the Computer Operations Division.) This job will be entered directly into the batch system with the resultant message, "HASP JOB NUMBER nnn AWAITING EXECUTION".
- 2) The TAPES and INPUT parameters are specified as NO. The job does not meet the express criteria. The card images for this job will be stored in the JOSHUA "hold queue" for subsequent execution. The resultant message is, "JOB IN HOLD QUEUE".
- 3) The TAPES parameter is YES and the INPUT parameter is NO. The resultant display is the "tapes" frame shown in the top half of Figure 20. The utility functions described on the right side of the frame are discussed in Section 9, "Maintaining the JOSHUA Data Base". Up to eight tape data sets may be defined on this frame. The FT NO. is the FORTRAN data set reference number, e.g. //FT10F001. VOL-SER is the volume serial number of the tape, e.g. VOL=SER=LAI234. DSNAME is the data set name, e.g. DSNAME or DSN=FILE1. FILE NO. is the file number on a tape volume and corresponds to the LABEL parameter on the JCL, e.g. LABEL=2. If FILE NO. is omitted, it assumes a value of one.

If an "E" is placed in either control position when the tape data is entered, the job will then be treated as in case 1 or 2 and be entered into the batch system or placed in the JOSHUA hold queue. If no control data is entered, the resultant display is shown at the bottom of Figure 20. This is the JCL that was generated from the data entered on the tapes frame. The system is now in the "update" paging mode so that anything on the page may be modified or more JCL may be added, DCB parameters may be changed, tape volumes changed to disk data sets, etc. When an "E" is encountered

FT NO.	VOL-SER	DSNAME	FILE NO	FT NO.'S FOR JOSHUA UTILITY FUNCTIONS
-----	-----	-----	-----	-----
:10	:LA1234	:FILE1	:1	
:11	:LA1234	:FILE2	:2	MAINTAIN - 10,11
:	:	:	:	MERGE - 12,13
:	:	:	:	LOAD - 14
:	:	:	:	DUMP - 15
:	:	:	:	
:	:	:	:	
:	:	:	:	
:	:	:	:	

```

//GO.FT10F001 DD UNIT=2400,VOL=SER=LA1234,DSN=FILE1,LABEL=1,
// DCB=(BUFNO=1,RECFM=VBS,BLKSIZE=8004,LRECL=800),DISP=(,KEEP)
//GO.FT11F001 DD UNIT=2400,VOL=SER=LA1234,DSN=FILE2,LABEL=2,
// DCB=(BUFNO=1,RECFM=VBS,BLKSIZE=8004,LRECL=800),DISP=(,KEEP)

```

PAGE 2

Figure 20

in either control position, the job will be either submitted to the batch system or placed in the JOSHUA hold queue based on the express job criteria.

- 4) The INPUT parameter is YES and the TAPES parameter is NO. The resultant display is the "input" frame shown in Figure 21. This feature of the EX function was designed primarily to enable the use of JMS as a batch monitor, hence the ENTER MONITOR INPUT text. This usage is described in detail in Section 8, "Using JMS as a Batch Monitor". If both the TAPES and INPUT parameters are set to YES, the input frame will follow the tapes frame.

An "R" control character may be used in either control field of the display containing the message indicating the disposition of the job, e.g. "JOB IS IN HOLD QUEUE". This will cause the first page of JCL that was just entered into the system to be displayed. All of the JCL and data may be modified and another job entered into the system. This may be repeated as many times as desired. However, it should only be done for jobs of the same category, either "express" or "hold queue".

5.3 The STATUS Function (ST)

The ST function is used to obtain the status of a batch background job. One argument is required. The first qualifier may be either a "HASP" job number, a batch job name or partial name, a user badge number, or the keyword "ALL". The keyword "ACCT" may be used as a second qualifier which will provide the status of jobs that have completed execution. When only one qualifier is used, the status of all jobs in the HASP queue is provided. Figure 22 shows the various messages that may result from a ST request when a job number is used as the argument. If the job is awaiting execution the message indicates the number of jobs ahead in priority for all jobs and for class M jobs only. All jobs submitted with the EX function are marked as class M. The message indicating that the job is on an initiator or in execution will occur from the time the job is selected by a HASP initiator until it has completed execution. The execution complete message will occur from that time until the job has finished printing and punching when the "not located" message will occur. When the job cannot be located the message also indicates the number of jobs in the HASP job queue.

Figure 23 shows the result of using the keywords "ALL" and "ACCT" as an argument. The display is identical when using a job name, partial job name, or badge number as the first qualifier. These other forms of the argument are used to select specified data from the entire set. In all cases, only jobs that have *completed* between 8:00 AM and 4:00 PM will be displayed.

Column one of the display, JOBNAME, is the HASP job name. The last three digits in the job name are the HASP job number. If the job has been submitted from a terminal, the first character of the name will be a "Q" followed by the users badge number.

Column two, P, is the HASP priority of the job. This is calculated from the estimated time and lines that appear in the job card, but may be altered by an operator.

Column three, C, is the class assigned to the job. All jobs submitted with the EX function will be assigned to class M.

Figure 21

```

TEMP0000 - 0000 - STD ----- 0000          TUBE 035
ST      --FUNCTION
110.....
.....

```

```

JOB 110 AWAITING EXECUTION
TOTAL JOBS AHEAD IN PRIORITY-   12
CLASS M JOBS AHEAD IN PRIORITY-   4

JOB 110 ON INITIATOR OR IN EXECUTION

JOB 110 EXECUTION COMPLETE

JOB 110 NOT LOCATED
JOBS IN HASP QUEUE-   34
CLASS M JOBS IN HASP QUEUE-   10

```

Figure 22

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page. E.1.48
		Rev. 2	Date 9/7

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

ST --FUNCTION

ALL.ACCT.....

.....

	JOBNAME	P	C	SYSIN	SYSOT	RDR---ON	STOP	CPU-TIME	EL--TIME	COMP	
*	*Q2791034	9	G	32	00	8:00:40	8:00:42	0:00:00	0:00:00	U000	1
*	*Q2628035	9	C	23	232	8:01:47	8:05:43	0:00:00	0:00:10	U000	2
*	*Q0778038	9	G	20	00	8:05:02	8:05:49	0:00:00	0:00:01	U000	3
*	*Q0778031	9	C	57	5011	7:56:29	8:05:49	0:00:01	0:00:18	S137	4
*	*WRH71036	9	G	15	12	8:03:35	8:05:51	0:00:00	0:00:08	U00C	5
*	*Q9037037	9	M	11	1099	8:03:54	8:06:35	0:00:03	0:00:46	U000	6
*	*Q2628039	9	C	23	477	8:06:56	8:07:11	0:00:01	0:00:09	U000	7
*	*TTTBM043	9	C	75	239	8:09:16	8:10:14	0:00:00	0:00:08	U008	8
*	*WRH71044	9	G	15	763	8:09:22	8:10:19	0:00:01	0:00:11	U000	9
*	*Q2628046	9	C	23	348	8:10:07	8:10:28	0:00:01	0:00:10	U000	10
	PAGE 1										

Figure 23

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.49
		Rev. 2	Date 9/73

Column four, SYSIN, is the number of input cards or card images associated with the job.

Column five, SYSOT, is the number of lines of printed output excluding job control language and system messages.

Column six, RDR---ON, is the time of day that the job was recieved by the HASP system, i.e. the time the last input card has been read by HASP.

Column seven, STOP, is the time of day that the job completed execution. The job will be printed after this time based on a new priority that is determined by the actual number of output lines.

Column eight, CPU-TIME, is the CPU time used by the job.

Column nine, EL--TIME, is the total time that the job occupied core storage. This is usually referred to as the elapsed or execute time.

Column ten, COMP, is the job completion code. The letter U is appended for a user code and the letter S for a system code. In a multistep job, a system code will always override a previous user code. If a system code does not occur, the highest user code encountered will be displayed.

Figure 23a shows the result of using the keyword "ALL" as an argument in order to examine the HASP queue which contains the status of jobs that have not completed all HASP functions. The same argument conventions, partial job names, etc, apply when using this form of the argument except that a badge number may not be used.

Column one of the display is the HASP job name and is identical in form to the JOBNAME in Figure 23. Column two indicates which function HASP is currently processing for each job or which function the job is currently awaiting. Column three contains the class of jobs that are executing or awaiting execution or the number of the printer if the job is awaiting print. Column four contains the HASP priority of the job as previously described. Column five will contain the keyword "HOLD" for all jobs that are in a "hold" status. All setup jobs and those jobs that do not meet the express job criteria are automatically placed in the "hold" status.

5.4 The View Queue Function (VQ)

The VQ function is used to display the JCL in the JOSHUA hold queue. No arguments are required. This function was provided primarily for operations personnel to determine the backlog of jobs in the queue. However, it may be used to verify questionable input data, and to obtain the queue job number. This job number appears on the right side of the line that contains the job card. Figure 24 shows a VQ request and a sample page of the resultant display.

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

ST --FUNCTION

ALL.....
.....

<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>
WHB42138	CN PRINTER2		PRI0	9
MVGG3137	AWAITING PRINT	2	PRI0	9
Q4137131	AWAITING PRINT	2	PRI0	9
SJJ61148	CN PRINTER1		PRI0	9
Q1991134	AWAITING PRINT	2	PRI0	9
CES23156	AWAITING PRINT	1	PRI0	9
Q9049157	AWAITING PRINT	2	PRI0	9
SJJ63150	EXECUTING	M	PRI0	8
L2102151	AWAITING EXEC	E	PRI0	8 HOLD
CAM8A123	EXECUTING	G	PRI0	7
	PAGE	1		

Figure 23a

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.50.1
		Rev. 2	Date 9/73

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

VQ --FUNCTION

.....

.....

```
//Q3868    JOB (3868,L033,060,20,0000,,,,,8242-1G,L5365-01,T,71),            010
// 'JOSHUA 3868            ',MSGLEVEL=1,CLASS=M
/*SETUP            M,400K,3868
//*OUTE   PRINT PRINTER2
//   EXEC JOSHUA,DISP=TMP,RLSE=YE,ACT1=L4120071,
//   USER=0666,MEMBER=MONTE2,SPACE=10,GTIME=060,BUFNO=010,
//   GOSIZE=400K,PDS='JOS.PGMS01',TIMER=OF,JOB=JRS20666,SSPACE=010
//GO.SYSIN DD *
SCRATCH=JRS20666
/*
```

PAGE 3

Figure 24

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.50.2
		Rev. 2	Date 9/77

6. Utility Functions

6.1 Display Catalog (DC)

The DC function is used to display the data contained in the JOSHUA data set catalog and the free space elements associated with each physical data set. Several options are provided that enable the selective retrieval of specified data. These options are discussed below.

- 1) Figure 25 - No argument is supplied - The resultant display is the catalog data corresponding to the "write" data set. The data fields are defined as follows.

DATA SET	- Data set name
IS	- Status of data set - either ONLINE or OFFLINE
FILE CATEGORY	- Protection category of the data set
RECORD NUMBER	- Position of data in the catalog
CREATED	- Creation date
EXPIRES	- Expiration date
NUMBER	- Data set reference number of the tree catalog or data storage areas
BASE	- Starting record minus one of the tree catalog or data storage areas
RECORDS USED	- Number of tree catalog or data storage records currently in use
MAX RECORDS	- Number of tree catalog or data storage records allocated to this data set
LAST NODE	- The last eight byte positions used in the last used tree record. There are forty eight "nodes" in each record.
EXTENT	- Catalog extent of data storage area

If the data set is offline, the line below the dashed line will contain the DD card that is necessary to be able to access the data. When a data set contains more than one extent, the data for each extent will be displayed on a separate page. The order in which they appear will depend on their position in the catalog. The data for the tree catalog will only appear in the display of the first extent.

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol. II	Page E.1.52
			Rev. 1	Date 2/73

TEMP0000 - 0000 - STD ----- 0000 TUBE 035

DC --FUNCTION

.....

.....

CATALOG DISPLAY

DATA SET 0000 IS ONLINE -FILE CATEGORY 00000000

RECORD NUMBER 38 - CREATED 021572 - EXPIRES 123199

TREE NUMBER	23	DATA NUMBER	26
BASE	3013	BASE	17901
RECORDS USED	2	RECORDS USED	66
LAST NODE	45	EXTENT	1
MAX RECORDS	8	MAX RECORDS	300

LASTPAGE 1

Figure 25

Title	Monitor System (JMS)	Vol.	Page
		II	E.1.53
Rev.	1	Date	2/73

- 2) Figure 26 - The argument is a data set name - The resultant display is identical to case 1 above for the specified data set.
- 3) Figure 27 - The argument is the keyword "USER" - The resultant display is the same as case 1 or 2 above for the User data set that has the same name as the users badge number.
- 4) Figure 28 - The argument is the keyword "ALL" - The resultant display contains the names of all JOSHUA data sets. This figure contains just the first page of these names.
- 5) Figure 29 - The argument is the keyword "STD.ALL" - The resultant display contains the names of all JOSHUA Standard data sets.
- 6) Figure 30 - The argument is the keyword "USER.ALL" - The resultant display contains the names of all JOSHUA User data sets. This figure contains the first page of these names.
- 7) Figure 31 - The argument is the keyword "JOB.ALL" - The resultant display contains the names of all JOSHUA Job data sets. This figure contains the first page of these names.
- 8) Figure 32 - The argument is the keyword "JOB" - The resultant display contains all the JOSHUA job names that contain the users badge number as the last four characters.
- 9) Figure 33 - The argument is the keyword "JOB.nnnn" - The value of nnnn in this case is a user badge number. The resultant display contains all of the JOSHUA job names that contain the specified badge number as the last four characters.
- 10) Figure 34 - The argument is the keyword "SPACE" - The resultant display contains the free space elements for each of the online physical data sets used by the JOSHUA system. This figure contains just the first page of the display. The top line of each page will contain the data set reference number and whether or not the data set is used for tree catalogs or data storage.

6.2 List (LI)

The List function is used to obtain a display of the record names in a given data set or a subtree of the data set. The display is in a tree structured form. If qualifiers on a line match the qualifiers on the previous line they will appear as blank fields. The function requires one argument; the data set name, the name of a group of records (subtree), or a complete record name. The output also contains the starting location of the associated data record to the left of the record name.

Figure 35 shows an LI request for the record names in the data set INPUT. Note that deleted record names and subtrees will appear on the list with the deleted qualifier being replaced by the word "DELETED". Only the first page of the resultant display is shown in this figure. Figure 36 shows a request for a subtree of the data set named INPUT.GAUGE. Figure 37 shows a request for a complete record name. A request of this type is used only

```

TEMP0000 - 0000 - STD ---- 0000          TUBE 035
DC      --FUNCTION
JCJ42791.....
.....

```

```

                        CATALOG DISPLAY
DATA SET JCJ42791  IS ONLINE  -FILE CATEGORY  00000000
RECORD NUMBER 196 - CREATED 011773 - EXPIRES 012573
TREE NUMBER      23          DATA NUMBER      31
BASE             4612        BASE             5772
RECORDS USED     4          RECORDS USED      641
LAST NODE        48          EXTENT           1
MAX RECORDS      18          MAX RECORDS      900
-----
LASTPAGE      1

```

Figure 26

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol.	Page
		II	E.1.55
		Rev.	Date
		1	2/73

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

DC --FUNCTION

USER.....

.....

CATALOG DISPLAY

DATA SET 0000 IS ONLINE -FILE CATEGORY 00000000

RECORD NUMBER 38 - CREATED 021572 - EXPIRES 123199

TREE NUMBER 23 DATA NUMBER 26

BASE 3013 BASE 17901

RECORDS USED 2 RECORDS USED 66

LAST NODE 45 EXTENT 1

MAX RECORDS 8 MAX RECORDS 300

LASTPAGE 1

Figure 27

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.56
		Rev. 1	Date 2/73

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

DC --FUNCTION

ALL.....

.....

LIST OF ALL DATA SET NAMES -

MULTIGRP	FEWGRP	ASSEMBLY	LATTICE	REACTOR	NUCPARAM	SYSTEM
TEMP0778	INPUT	3618	FINEGRP	ENDFB	3618	3401
2733	3897	1111	3174	0666	2425	3326
3362	3782	0777	0888	2967	3736	3809
1058	TEMP0829	3206	3712	TEMP3208	3944	TEMP9037
1299	9037	0000	3736	0978	1578	1718
TEMP3554	3853	TEMPLATE	1718	2485	TEMP0523	3801
4137	CELL	TEMP3867	TEMP3006	BRA09037	SAVEDATA	3907

PAGE 1

Figure 28

TEMP0000 - 0000 - STD ---- 0000
TUBE 035

DC
--FUNCTION

STD.ALL.....

.....

LIST OF STD
DATA SET NAMES -

MULTIGRP FEWGRP
ASSEMBLY LATTICE REACTOR
NUCPARAM SYSTEM

INPUT FINEGRP ENDFB
TEMPLATE CELL
SAVEDATA INPUT

TEMPLATE PROPERTY SECRET
INPUT ASSEMBLY

LASTPAGE
1

Figure 29

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.58
		Rev. 1	Date 2/73

TEMP0000 - 0000 - STD ---- 0000
TUBE 035

DC
--FUNCTION

USER.ALL.....
.....

LIST OF USER
DATA SET NAMES -

3618	3618	3401	2733	3897	1111	3174
0666	2425	3326	3362	3782	0777	0888
2967	3736	3809	1058	3206	3712	3944
1299	9037	0000	3736	0978	1578	1718
3853	1718	2485	3801	4137	3267	3907
3934	1275	2973	2485	3354	3331	3865
3939	2733	3554	3947	3922	3762	2973
1149	3939	2772	3055	1275	4106	0978

PAGE
1

Figure 30

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.59
		Rev. 1	Date 2/73

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

DC --FUNCTION

JOB.ALL.....
.....

LIST OF JOB DATA SET NAMES -

TEMP0778 TEMP0829 TEMP9037 TEMP3589 TEMP3554 TEMP0523 TEMP3867
TEMP3006 HKCK1149 TEMP3559 TEMP2628 HKCE1149 FEW19037 CLEN2628
TEMP1790 JAS91878 CEPB3662 TRI53907 TEMP0606 BRA29037 TEMP3782
FDLT3736 HV5R3267 TEMP3170 MMAF3947 M16D3809 DH5B1299 HRV53144
TEMP3138 T5CR3907 BRAY3331 U2364137 DH151299 MMA23947 RJPA3897
IMMC3903 MMAE3947 CEPC3662 BAV13331 TD5R3907 DEH91299 QEST3809
TEMP0000 MVG34148 MVG24148 FLOW1275 D2013453 LATT9037 STWT3554
AT403055 TEMP3208 FEW29037 MAC33947 MAPT3782 SAMP3865 TES13809

PAGE 1

Figure 31

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol.	Page
		II	E.1.60
		Rev.	Date
		1	2/73

TEMP0000 - 0000 - STD ---- 0000

TUBE 035

DC --FUNCTION

JOB.....

.....

LIST OF JOB DATA SET NAMES -

TEST0000 JCJ40000 TEMP0000

LASTPAGE 1

Figure 32

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.61
		Rev. 1	Date 2/73

TEMP0000 - 0000 - STD ---- 0000
TUBE 035
DC --FUNCTION
JOB.9037.....
.....

LIST OF JOB DATA SET NAMES -

TEMP9037 BRA09037 BRA39037 FEW19037 BRA29037 ASSY9037 LATT9037
FEW29037 BENT9037 TPC09037 BRA19037 LOAD9037

LASTPAGE 1

Figure 33

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

DC --FUNCTION

SPACE.....
.....

FREE SPACE FOR JOB DATA -DSRN 28- (DATA)

59	FREE RECORDS AT	2 -	35	FREE RECORDS AT	2161 -
3	FREE RECORDS AT	3996 -	1	FREE RECORDS AT	4132 -
67	FREE RECORDS AT	4433 -	2	FREE RECORDS AT	4589 -
15	FREE RECORDS AT	4702 -	2	FREE RECORDS AT	6467 -
281	FREE RECORDS AT	9488 -	18	FREE RECORDS AT	10570 -
168	FREE RECORDS AT	10843 -	18	FREE RECORDS AT	11511 -
59	FREE RECORDS AT	12929 -	1	FREE RECORDS AT	13005 -
98	FREE RECORDS AT	13024 -	82	FREE RECORDS AT	18740 -

PAGE 1

Figure 34

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol. II	Page E.1.63
			Rev. 1	Date 2/73

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

LI --FUNCTION

INPUT.GAUGE.....
.....

INPUT.GAUGE

13601 JOB .SAMPLE .
13602 CASE .SAMPLE .
13605 .STD14130.
13607 .STD16130.
13609 .STD18151.
13610 GEOMETRY.SAMPLE .
13621 .TESTFM .
13632 .STD14130.
13643 .STD16130.
13657 .STD18151.

LASTPAGE 1

Figure 35

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

LI --FUNCTION

INPUT.....
.....

INPUT

13600 DELETED..CASE1

13601 GAUGE .JOB .SAMPLE .

13602 .CASE .SAMPLE .

13605 . .STD14130.

13607 . .STD16130.

13609 . .STD18151.

13610 .GEOMETRY.SAMPLE .

13621 . .TESTFM .

13632 . .STD14130.

13643 . .STD16130.

PAGE 1

Figure 36

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

LI --FUNCTION

INPUT.GAUGE.CASE.SAMPLE.....

.....

INPUT.GAUGE.CASE.SAMPLE

13602 NNNNNNNN.

LASTPAGE

1

Figure 37

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.66
		Rev. 1	Date 2/73

to determine if the record exists. Since there is no tree or subtree involved, the qualifier adjacent to the record number is meaningless.

When a specific data set name is not appended to the argument, such as above, the names will be retrieved from the first data set in the search sequence that contains any member of the requested subtree. In the above cases, INPUT records did not exist in either the Job or User data set and the names were retrieved from the Standard data set as indicated by the data set name at the top of the display.

6.3 List Brothers (LB)

The LB function is used to obtain a display of the qualifiers at a given level in a subtree. One argument is required; the name of a subtree. Figure 38 shows an LB request that is used to determine all of the stage names in the JERICHO input records in data set 3865. The top line of the resultant display contains the name of the specified subtree with the special symbol, \$?\$, attached. This symbol indicates the level of the subtree that contains the brother qualifiers listed below.

6.4 RENAME (RE)

The RE function is used to change a specified qualifier in a data set tree catalog. Two arguments are required. The first argument may be either a record name or the name of a subtree. The specified argument must exist in the "write" data set as defined on the top line of the select frame. The second argument must be a single name qualifier. The qualifier specified in the second argument will replace the last qualifier entered in the first argument.

Figure 39 shows two different rename requests. There is no resultant message associated with a rename request; the select frame will reappear. The top request would result in the subtree named 0000.ASSEMBLY.MARK6C, changed to 0000.ASSEMBLY.MARK6B. Deleted qualifiers in a subtree may also be renamed as shown in the bottom request. (The DELETE function is discussed in Section 9.1, "The DELETE Function".) This request would result in the *first* deleted son qualifier of the subtree named 0000.ASSEMBLY, changed to MK5R. The next deleted brother qualifier could then be renamed by repeating the request with a different second argument.

6.5 Set User (SU)

The SU function is used to change the User data set name in the data set search sequence and in the JCL generated by the EX function. One argument is required; a User data set name. Figure 40 shows an SU request for User data set "1234". The resultant display is the select frame with the User data set name in the data set search sequence changed on the top line from "0000" to "1234".

TEMP0000 - 0000 - STD ---- 0000 TUBE 035
 LB --FUNCTION
 3865.INPUT.JERICO.STAGE.....

3865.INPUT.JERICO.STAGE.\$?\$
 P1411 P1511 P1611 STD1 STD2 STD3 STD161
 STD162 STD163 P1121112 P11212 P11213 K1611

LASTPAGE 1

Figure 38

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol.	Page
			II	E.1.68
			Rev.	Date
			1	2/73

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

RE --FUNCTION

ASSEMBLY.MARK6C.....

MARK6B.....

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

RE --FUNCTION

ASSEMBLY.<D>.....

MK5R.....

Figure 39

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.69
		Rev. 2	Date 9/73

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

SU --FUNCTION

1234.....
.....

TEMP0000 - 1234 - STD ---- 0000 TUBE 035

SU --FUNCTION

1234.....
.....

Figure 40

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.70
		Rev. 2	Date 9/73

6.6 Set Write (SW)

The SW function is used to change the name of the "write" data set. One argument is required; a JOSHUA data set name. The function does two things before changing the name. 1) A record named dsn.SYSTEM.ACCESS.dsn is read where dsn is the function argument. This record contains a list of badge numbers of users that may update the specified data set. The "write" data set name will only be changed when the requestors badge number is on this list or the record cannot be located. 2) It is determined if the data set is being updated in another region of core storage, e.g. a batch job may be writing to the specified Job data set. The "write" data set name will only be changed if another job is not updating the same data set. If the name is changed, other jobs may not update the data set, e.g. if the "write" data set is a job name, a job specifying that job name will be immediately terminated. Figure 41 shows an SW request for the data set named "1234". The resultant display has the "write" data set name at the top of the screen changed from "0000" to "1234".

6.7 PRINT (PR)

The PR function is used to produce a hard copy printout or a punched deck of cards of previously saved CRT images. Up to ten images may be saved by placing the character P in either control field on the screen. Either one or two arguments are used. The first argument is the users output box number, e.g. L021. If the second argument is not entered the CRT images will be printed. If the keyword, PUNCH, is entered as a second argument, the CRT images will be punched. Figure 42 shows two PR requests. The first will cause the images to be printed, the second to be punched. The resultant display in each case will be, "HASP JOB NUMBER nnn AWAITING EXECUTION".

6.8 END (EN)

The EN function is used to terminate a terminal session, i.e. sign-off. No arguments are required. Figure 43 shows an EN request. The resultant display is the sign-on frame.

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol. II	Page E.1.71
			Rev. 1	Date 2/73

TEMP0000 - 1234 - STD ---- 0000
TUBE 035

SW
--FUNCTION

1234.....
.....

TEMP0000 - 1234 - STD ---- 0000
TUBE 035

SW
--FUNCTION

1234.....
.....

Figure 41

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.72
		Rev. 1	Date 2/7

```

TEMP0000 - 0000 - STD ---- 0000          TUBE 035
PR      --FUNCTION
L021.....
.....

```

```

TEMP0000 - 0000 - STD ---- 0000          TUBE 035
PR      --FUNCTION
L021.....
PUNCH.....

```

Figure 42

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol. II	Page E.1.73
			Rev. 1	Date 2/73

TEMP0000 - 1234 - STD ---- 1234
TUBE 035

EN
--FUNCTION

.....

.....

.....BADGE
TUBE 035

- ..COST CODE

.....USER DATA SET

...USER ID

...FILE ID

.....FILE CATEGORY

Figure 43

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.74
		Rev. 1	Date 2/73

7. Using Monitor Procedures

A monitor procedure is a series of predefined function requests and control parameters that are stored in a JOSHUA data record named SYSTEM.PROC.q1.q2. The variable qualifiers q1, and q2 represent the procedure name, e.g. the procedure named CREATE.GAUGE would be stored in the record named SYSTEM.PROC.CREATE.GAUGE. Name qualifiers in the procedure function requests are defined as literals or variables. The variable procedure qualifiers are obtained from previously created data records, the active Job data set name, or are carried forward from a previous step in the procedure. A procedure is utilized with the Execute Procedure (EP) function, specifying the procedure name as an argument. Procedure steps may be skipped with the SKIP function, execution may be interrupted with the Interrupt Procedure function, the procedure restarted with the Restart Procedure function, and execution terminated with the Stop Procedure function.

7.1 The Execute Procedure Function (EP)

The EP function is used to perform a series of predefined function requests. One argument is required; a procedure name. Figure 44 shows an EP request for the procedure named CREATE.GAUGE. The resultant display for this procedure is a select frame containing a CD function request for the record named INPUT.GAUGE.JOB.TEMP0000. The procedure name and step number will always appear at the top of the select frame. The qualifier TEMP0000 is a variable qualifier in this record name. The Job data set name is used in this case based on control data in the procedure record, i.e. whenever the keyword ?\$JOB is encountered it is replaced by the Job data set name. The named data record may be created by depressing the shift/enter keys and overlaying the proper fields in the template display. When the record has been created and displayed, the next function request in the procedure will appear with the step number indicated at the top of the screen. Since "loops" within the procedure allow steps to be executed up to *nine* different times, the step number could be the same or less than the previous step number.

The function request may be changed in any step of the procedure, i.e. function name, arguments, or both. However, records that contain variable procedure qualifiers must be created to enable the procedure to function properly. If a data retrieval error occurs during the execution of a procedure, the procedure execution will be terminated.

More detailed information concerning the capabilities and limitations of monitor procedures may be found in Section 11, "Creating Monitor Procedures".

7.2 The SKIP Function (SK)

The SK function is used to skip a monitor procedure step. No arguments are required. Care must be taken not to skip a step that will create a record that will be used by a following step. SK could be used in the following situations.

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol. II	Page E.1.75
			Rev. 1	Date 2/73

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

EP --FUNCTION

CREATE.GAUGE.....

.....

PROCEDURE (CREATE .GAUGE) STEP 1-

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

CD --FUNCTION

INPUT.GAUGE.JOB.TEMP0000

.....

Figure 44

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.76
		Rev. 1	Date 2/73

- 1) The function request is a CD and the record specified in the argument has been previously created.
- 2) The procedure contains four steps. The first three steps are CD requests that prepare the required input data for a specific computational module. The fourth step is an EX request for the computational module. Also, the procedure loops thru steps two and three so as to set up data for several problems. In this case it may be desirable to skip the EX request because of current trouble in the computational module. The problems may be executed after the module is fixed.

Figure 45 illustrates the use of the SK function for the function request shown in Figure 44. The CD request is overlayed with SK and the resultant display is the next function request in the procedure. In this case the record named INPUT.GAUGE.JOB.TEMP0000 had been previously created so that the variable procedure qualifier in the next function argument could be located.

7.3 The Interrupt Procedure Function (IP)

The IP function is used to interrupt a monitor procedure. No arguments are required. This may be necessary to determine additional information such as a list of a data set subtree or the status of a job. No messages are generated by the IP function. However, the procedure name and step number will not appear on the resultant select frame. (See Figure 46)

7.4 The Restart Procedure Function (RP)

The RP function is used to continue the execution of a monitor procedure after it has been interrupted with the IP function. No arguments are required. Figure 47 shows the RP request necessary to continue the procedure that was interrupted as shown in Figure 46. Note that a procedure may not be restarted after the EN function has been used, i.e. the user has signed off.

7.5 The Stop Procedure Function (SP)

The SP function is used to terminate the execution of a procedure. No arguments are required. The resultant display is a normal select frame, see Figure 48.

PROCEDURE (CREATE .GAUGE) STEP 1-
 TEMP0000 - 0000 - STD ---- 0000 TUBE 035
 SK --FUNCTION
 INPUT.GAUGE.JOB.TEMP0000

PROCEDURE (CREATE .GAUGE) STEP 2-
 TEMP0000 - 0000 - STD ---- 0000 TUBE 035
 CD --FUNCTION
 INPUT.GAUGE.CASE.SAMPLE

Figure 45

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.78
		Rev. 1	Date 2/73

```

PROCEDURE (CREATE .GAUGE ) STEP 2-
TEMP0000 - 0000 - STD ---- 0000      TUBE 035
IP      --FUNCTION
INPUT.GAUGE.CASE.SAMPLE .....
.....

```

```

TEMP0000 - 0000 - STD ---- 0000      TUBE 035
IP      --FUNCTION
INPUT.GAUGE.CASE.SAMPLE .....
.....

```

Figure 46

TEMP0000 - 0000 - STD ---- 0000 TUBE 035
RP --FUNCTION

.....
.....

PROCEDURE (CREATE .GAUGE) STEP 2-

TEMP0000 - 0000 - STD ---- 0000 TUBE 035
CD --FUNCTION

INPUT.GAUGE.CASE.SAMPLE
.....

Figure 47

PROCEDURE (CREATE .GAUGE) STEP 2-

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

SP --FUNCTION

INPUT.GAUGE.CASE.SAMPLE

.....

TEMP0000 - 0000 - STD ---- 0000 TUBE 035

SP --FUNCTION

.....

.....

Figure 48

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.81
		Rev. 2	Date 9/73

8. Using JMS as a Batch Monitor

The JOSHUA Monitor System may be run as a batch monitor thru the use of the JOSHUA cataloged procedure. Function requests are entered on input data cards and output displays are printed. The same procedure is used to execute computational modules and is described in PART C - Section 4, "Module Execution". It is not necessary to supply any symbolic JCL parameters. However, if a Job data set is not to be allocated, the SPACE parameter should be set to zero to avoid the possibility of a duplicate job name. The keyword input cards, SCRATCH=, CATEGORY=, and DATASET=, may be used if desired. Table I, "JMS Functions", indicates which monitor functions may be used in the batch mode.

A card containing the characters, S/E, in columns one thru three is used to signal the start of batch monitor operation. Function requests are punched starting in column one of each data card. Each request will be composed of one to four cards depending on the number of required arguments. The last card of each request must contain the characters S/E in columns one thru three.

A batch monitor job will be terminated if an error condition occurs during the execution of any function that is updating a data set. These functions are indicated in Table I with an asterisk.

Figure 49 shows the input cards for a sample batch monitor job. (When keyword cards are used they precede the first S/E card.) The first function request is to list data set 0000. The output tree listing will be printed with fifty lines/page rather than the ten line pages generated at a terminal. The second request is to display the subtree named "SYSTEM.PROC". The keyword "STD" is appended to the name to make sure the standard procedures are displayed. The output is printed in groups of ten lines with a blank line between each group.

The next request establishes a "write" data set for the subsequent copy request. This is necessary since the monitor does not automatically establish the User data set as the "write" data set when operating in the batch mode. Jobs containing SW requests must be submitted by Operations as "special handling" in order to validate the user. The last request copies a subtree from a Job data set to the User data set 0000. The name of the subtree is also changed during the copy operation. During this request and the previous DD request that also had a subtree name as an argument, the entire subtree is processed without user intervention. In the terminal region these requests are processed one record at a time.

Data records may be created from input cards with the CD function. The data cards should immediately follow the function request and must be followed by an S/E card. If the number of data cards is not a multiple of ten, blank cards should be added to satisfy this condition. Note that an SW request must be issued prior to the CD request.

Batch monitor jobs may be conveniently submitted from a terminal by specifying YES in the INPUT field of the execute frame. (See Section 5.2, "The EXECUTE Function".) The input data should overlay the /* on each line of the input frame (see Figure 21). The //GO.SYSIN DD* and the first S/E card are generated by the system and should be omitted from the input stream. The comments, "ENTER MONITOR INPUT", may be left as is, overlayed, or blanked. All jobs submitted from a terminal that contain input data will be marked as "special handling".

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.82
		Rev. 2	Date 9/77

```

//JCJ22791 JOB (2791,L021,2,2,,,,,8690-00,,T,00),JCJ,CLASS=M
//STEP1 EXEC JOSHUA,SPACE=0
//GO.SYSIN DD *
S/E
LI
0000
S/E
DD
STD.SYSTEM.PROC
S/E
SW
0000
S/E
CO
TEMP0000.ONEGRP.STANDARD
ONEGRP.VERSN1
S/E
/*

```

Figure 49

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.83
		Rev. 2	Date 9/73

8.1 The Punch Data Function

Formatted input data cards may be obtained for existing named data records thru the use of the Punch Data function (PD). This function is identical to the DD function except that the output is punched on cards instead of printed. A multiple of ten cards will always be punched. Also, the job card must be modified to include the estimated number of cards since this field does not appear on the execute frame.

8.2 The Execute Member Function (EM)

The EM function may be used to execute computational modules along with other monitor system requests. If no argument is supplied, the module specified by the MEMBER parameter on the EXEC card will be executed. A module name may be supplied as an argument and will override the MEMBER parameter. Figure 50 shows the JCL and input cards necessary to execute a computational module and also obtain a list of the record names in the Job data set.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page - E.1.84
		Rev. 2	Date 9/77

```

//JCJ22791 JOB (2791,L021,2,2,,,,,8690-00,,T,00),JCJ,CLASS=M
//STEP1 EXEC JOSHUA,MEMBER=COG,USER=0000,JOB=TST12791,SPACE=2,
// GOSIZE=240K,SSPACE=0
//GO.SYSIN DD *
SCRATCH=TST12791
S/E
EM
S/E
LI
TST12791
S/E
/*

```

Figure 50

Title	Monitor System (JMS)	Vol.	Page
		II	E.1.85
Rev.	1	Date	2/73

9. Maintaining the JOSHUA Data Base

The JOSHUA data base is composed of a pool of preallocated physical data sets that reside on direct access storage devices. All types of JOSHUA data sets are allocated within this pool. Standard and User data sets are allocated by CAD personnel with a special utility program whereas all others are allocated by JMS based on data supplied by the user, e.g. the SPACE parameter on the EXEC card. Since the amount of space in the pool is necessarily limited because of its high cost, data must continually be deleted or moved to less expensive storage such as tape volumes. Some of this maintenance is accomplished by automatic system actions. However, most data set maintenance must be done by the user with the utility functions described later in this section. The following paragraphs describe the maintenance procedures for each type of JOSHUA data set. Note that data sets may be maintained only by those persons authorized to update the data set.

Standard and User Data Sets - Records or subtrees are deleted with the DELETE function. The space occupied by the deleted records is recovered by using the MAINTAIN function. Data records may be saved on tape by first copying them to the Scratch data set and using the DUMP function to move the Scratch data set to tape. These operations "free up" space within a specified JOSHUA data set but do not return space to the data pool. Records stored on tape may be added to a data set with the MERGE function or replace the data set with the LOAD function. Either the MAINTAIN or DUMP functions may be used to obtain a backup tape copy of a data set.

Job Data Sets - Job data set space is maintained in several ways. First, the symbolic parameters on the EXEC card provide several options. The DISP parameter may be set to a value of TMP. This will cause the data set to be scratched at the end of the job and all space is returned to the pool. The RLSE parameter, when set to a value of YES, will cause all unused space in the data set to be released to the pool at the end of the job.

Job data sets that are retained should be scratched as soon as possible. Data records may be copied to a User data set with the COPY function either at a terminal or in a batch monitor run. The batch monitor should be used when more than a few records are involved since only one record at a time may be copied at a terminal. If the data in a Job data set is needed only as input for another job that will be run in the very near future, the DISP parameter may be set to a value of OLD to avoid copying the data to a User data set. If additional records are to be written, the necessary space must have been specified when the data set was created and the RLSE parameter set to a value of NO.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.86
		Rev. 1	Date 2/73

Job data sets may be scratched by a user in two ways. The keyword data card, SCRATCH=, may be used any time a batch job is run. This type card is generated from the name entered in the SCRATCH field of the execute frame. The SCRATCH function may also be used from either a terminal or batch run. Job data sets will also be scratched by a system utility program based on their expiration date and also if they contain no entries. This feature is included to purge the system of forgotten data sets and is not intended to relieve the user of the responsibility of scratching unneeded data sets.

A large Job data set that may be needed for future examination or input usage should be moved to tape with the DUMP function at the end of the job. It may then be scratched with the SCRATCH function. The data set may be restored to the data base at any time with the LOAD function. If only a portion of the records are required for future use, they may be copied to the Scratch data set after it has been deleted and then the Scratch data set may be moved to tape with the DUMP function. The tape data set may then be restored as a smaller Job data set in the future.

Scratch Data Sets - Space allocated to a Scratch data set is always returned to the pool at the end of a job. The DELETE function may be used, with the keyword SCRATCH as an argument, to recover all space within the data set during a run. This technique might be used after a module execution that has used the Scratch data set and before selectively copying records to be moved to tape with the DUMP function.

All of the utility functions that require the use of tapes may only be run in the batch mode. However, these batch mode jobs may be conveniently submitted from a terminal with the EX function. See Section 5.2, "The EXECUTE Function".

9.1 The DELETE Function (DE)

The DE function is used to delete a record or subtree name in a data set tree catalog. The argument is a record or subtree name followed by the special qualifier, <D>. The specified argument must exist in the "write" data set as defined on the top line of the select frame.

The qualifier preceeding the special qualifier will be replaced in the tree catalog with special characters, thus causing the specified record or subtree to no longer exist by its given name. Since the record still does exist, no space is recovered in the data set. Space recovery is accomplished with the MAINTAIN function described later in this section.

When the argument consists of a data set name only, i.e. TEMP0000, the DE function operates in a different manner since the data set name is not stored in the tree catalog. In this case, the values in the data set control block that indicate the amount of used space in the data set are re-set to indicate that no space has been used. This effectively deletes the entire data set and also recovers all space. When this form of the DE function is used, two special qualifiers must be added to the data set name.

Figure 51 shows two types of DE requests. The top request causes the qualifier, RECORD3, to be overlaid, thus making it inaccessible. The bottom request causes all records in the data set named TEMP0000 to be deleted and all space to be recovered. There is no resultant display for a DE request; the select frame will be returned.

9.2 The Delete Brothers Function (DB)

The DB function is used to delete more than one record or subtree name in a data set tree catalog with a single function request. The argument is a record or subtree name followed by the special qualifier, <D>. Figure 52 shows a DB request for the subtree illustrated in Section 6.3, "The List Brothers Function". In this case the "write" data set was first established to be 3865 since all delete requests are directed only to this data set. The resultant display is a list of brother qualifiers for the subtree listed on the top line at the level specified by the special qualifier, \$?\$. These qualifiers will always be retrieved from the "write" data set catalog; the search sequence is ignored.

All qualifiers that are preceded with a non blank character when the data is entered will be deleted. The tab key should be used to position the cursor for each qualifier. Figure 53 shows the resultant display frame as it could be entered to delete seven of the thirteen brother qualifiers in the subtree. The character, X, was arbitrarily used in this example. Each of the so marked qualifiers will be overlaid in the tree catalog with the special deleted qualifier. The resultant display is the select frame. The DB request can be aborted during the time the top display is on the screen by placing an S in either control field.

9.3 The MAINTAIN Function (MA)

The MA function is used to recover wasted space in JOSHUA data sets that occurs from delete operations, and to reorder the data according to the logical structure of the record names. It also produces a listing of all record names and a back-up sequential data set. One argument is required; the data set name. In addition, a data definition (DD) card must be supplied for the sequential data set.

Figure 54 shows the JCL and input data cards necessary to use the MA function. The SETUP card is used to specify the tape volume at the operators console. Only two symbolic parameters need to be specified on the EXEC card. The SSPACE parameter should be set equal to the amount of used

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol. II	Page E.1.88
			Rev. 1	Date 2/75

```

TEMP0000 - 0000 - STD ---- 0000          TUBE 035
DE      --FUNCTION
MODIFY.SAMPLE.RECORD3.<D>.....
.....

```

```

TEMP0000 - 0000 - STD ---- TEMP0000      TUBE 035
DE      --FUNCTION
TEMP0000.<D>.<D>.....
.....

```

Figure 51

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol. II	Page E.1.89
			Rev. 1	Date 2/73

```

TEMP0000 - 0000 - STD ---- 3865          TUBE 035
DB      --FUNCTION
INPUT.JERICO.STAGE.<D>.....
.....

```

```

3865.INPUT.JERICO.STAGE.$?$
PLACE A NON BLANK CHARACTER IN (FRONT OF EACH QUALIFIER TO BE
DELETED.
: P1411   : P1511   : P1611   : STD1    : STD2    : STD3
: STD161  : STD162  : STD163  : P1121112: P11212  : P11213
: K1611

```

LASTPAGE
1

Figure 52

JOSHUA DPSTM-500	Title	Monitor System (JMS)	Vol.	Page
			II	E.1.90
			Rev.	Date
			1	2/7

3865.INPUT.JERICO.STAGE.\$?\$

PLACE A NON BLANK CHARACTER IN FRONT OF EACH QUALIFIER TO BE DELETED.

:XP1411 :XP1511 :XP1611 : STD1 : STD2 : STD3

: STD161 : STD162 : STD163 :XP1121112:XP11212 :XP11213

:XK1611

LASTPAGE 1

TEMP0000 - 0000 - STD ---- 3865 TUBE 035

DB --FUNCTION

INPUT.JERICO.STAGE.<D>.....

.....

Figure 53

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.91
		Rev. 1	Date 2/73

```
//JDD12791 JOB (2791,L021,2,2,,,,,8690-00,,T,00),JDD,CLASS=M
/*SETUP          LA0001
//STEP1 EXEC JOSHUA,SSPACE=14,SPACE=0
//GO.FT10F001 UNIT=2400,VOL=SER=LA0001,DSN=MA0000,
// DCB=(BUFNO=1,RECFM=VBS,LRECL=800,BLKSIZE=8004),DISP=(,KEEP)
//GO.SYSIN DD *
S/E
MA
0000
S/E
/*
```

Figure 54

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.92
		Rev. 1	Date 2/

space in the data set that is being maintained. The DC function can be used to determine this. The SPACE parameter should be set to zero to avoid any duplicate job name problems. The DD name of the sequential data set is FT10F001. A personal tape volume should normally be used so that it may be kept as a back-up copy of the data set. Any data set name, DSN, may be used. The DCB parameters shown in the example may be changed if the GOSIZE parameter is adjusted accordingly. The first S/E data card signals the start of monitor operation. The last three data cards represent a MAINTAIN function request for the User data set 0000. The input stream is terminated with a /* delimiter card. All jobs containing MA requests must be submitted by Operations as "special handling".

The MA function is actually a combination of several function operations. First, a Set Write is issued to determine if the user is authorized to update the data set and whether the data set is being updated in another task, e.g. the terminal system. The COPY function is then used to copy the records in the data set to the Scratch data set. During this process all deleted records are ignored and all wasted space is recovered. The LIST function is then used to obtain a listing of the record names in the Scratch data set. The Scratch data set is then moved to tape with the DUMP function and the data set is deleted.

At this point a record named DSMAINT is written to the data set. This is a protection feature that prevents the job from being rerun if it is abnormally terminated. The record will be deleted at the end of a successful run. This record should only be deleted by a user when it is determined that there is a valid back-up copy of the data set. The final step of the function is to copy the Scratch data set back to the data set and to delete the DSMAINT record.

9.4 The DUMP Function (DU)

The DU function is used to move a JOSHUA data set to a sequential data set, normally a tape. One argument is required; the data set name. In addition, a data definition (DD) card must be supplied for the sequential data set. The JCL and input data required to use the DU function are the same as for the MA function shown in Figure 54 with the following exceptions.

- 1) SSPACE should be set to zero on the EXEC card.
- 2) The DD name for the sequential data set is FT15F001.
- 3) The function name is DU

9.5 The LOAD Function (LO)

The LO function is used to move a sequential copy of a JOSHUA data set, normally residing on a tape, to the JOSHUA data base. One argument is required; the data set name. The data set name is the name of the data base data set and need not be the same as the name of the sequential copy of the data set. This allows Job data sets to be loaded into Scratch data sets and vice versa. In addition, a data definition (DD) card is required for the sequential data set. The JCL and input data required to use the LO function are the same as for the MA function shown in Figure 54 with

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.93
		Rev. 2	Date 9/73

the following exceptions.

- 1) SSPACE should be set to zero on the EXEC card if a Scratch data set is not loaded.
- 2) A Job data set may be allocated by specifying a job name and space on the EXEC card.
- 3) The DD name for the sequential data set is FT14F001.
- 4) The function name is LO.

9.6 The MERGE Function (ME)

The ME function is used to merge a sequential copy of a JOSHUA data set, normally residing on a tape, with an existing data set in the data base. One argument is required; the data set name. The data set name is the name of the data base data set and need not be the same as the name of the sequential copy of the data set. In addition, a data definition (DD) card is required for the sequential data set. The JCL and input required to use the ME function are the same as for the MA function shown in Figure 54 with the following exceptions.

- 1) The DD name for the sequential data set is FT12F001.
- 2) The function name is ME.

9.7 The SCRATCH Function (SC)

The SC function is used to remove JOSHUA data sets from the data base and to release unused space in Job data sets. The first qualifiers of the one required argument is the data set name and must be followed by the special qualifier <S>. The keyword RLSE is an optional third qualifier and is used to release space in a data set rather than removing it from the data base. The data set name may also be replaced with the keyword ALL when used from a terminal. This option causes the function to scratch all Job data sets that have the users badge number as the last four characters of the name.

Figure 55 illustrates the use of the SC function. The top request removes the data set named TST12791 from the data base and the bottom request removes all Job data sets that contain the users badge number as the last four characters of the name. There is no resultant display for an SC request; the select frame is returned.

TEMP0000 - 0000 - STD ---- 0000 TUBE 035
 SC --FUNCTION
 TST12791.<S>.....

TEMP0000 - 0000 - STD ---- 0000 TUBE 035
 SC --FUNCTION
 ALL.<S>.....

Figure 55

10. Templates

10.1 The Create Template Function (CT)

The CT function is used to create a "template" for a given record type. A template contains the following data:

- (1) The I/O list.
- (2) The type of each variable in the list (e.g. REAL*8).
- (3) The format used to display the record.

Templates are used by JOSHUA Monitor System functions (DD, CD, MD) to format binary named data records for displays, and to unformat EBCDIC displays and write binary data records.

Records that contain the same fixed qualifiers and have variable qualifiers in the same positions are of the same type. Variable qualifiers in a template name must begin with a question mark (they may consist of a "?" only). Examples:

- (1) Record type with no variable qualifiers.
Record name - LATTICE.CONTENTS
Template name - LATTICE.CONTENTS
- (2) Record type with variable qualifiers.
Record name - LATTICE.MK16.GEOMETRY.RUN1
Template name - LATTICE.?TYPE.GEOMETRY.?CASE
or - LATTICE.?.GEOMETRY.?

All records of the same type must have the same I/O list and will be displayed with the same format.

The CT function requires one argument; the template name. Figure 56 shows a CT request. The resultant display is referred to as the "list" frame. The input/output list and variable types are entered on this frame. The list is the same as a FORTRAN I/O list with the following additions and restrictions.

- (1) A variable may not appear after the first equal sign in each nest. e.g. ((X(I,J),I=1,3),Y(J),J=1,5) is illegal.
- (2) Indices may be omitted. e.g. M,(X,=1,M) is the same as M,(X(I),I=1,M).
- (3) The upper limits of implied DO loops may be singly indexed variables (vectors). In this case the index must be omitted. e.g. N,(M,(X,I=1,M),J=1,N). This is the same as: N,(M(J),(X(I),I=1,M(J)),J=1,N). The above list is shown for example only. The index of the vector M must be omitted.

Figure 56

Title	Monitor System (JMS)	Vol.	Page
		II	E.1.97
Rev.	1	Date	2/73

- (4) No more than one implied DO loop in any nest may have triangular indexing (i.e., refer to an outer loop index as an indexing parameter). e.g., the following is invalid
(((X,I=1,J),J=1,K),K=1,10).
- (5) Indexing must be explicit since dimension statements are not used.
- (6) The list may have no more than 5 nests, 12 implied loops, or 20 variable names.
- (7) Variable names in the list may be any length. e.g. the following list is valid.
REGIONNUMBER,(COMPOSITION(I),I=1,REGIONNUMBER).

The list should be entered in the first 248 positions of the frame that are denoted with question marks. Variable type data may be entered starting on line 6 in the 320 positions denoted with question marks. The right hand side of the screen contains pointers to the proper starting positions for both the list and the type information. The list limits and the permissible variable types are also given.

The variable types assume the same default values as in FORTRAN unless otherwise specified. A character variable may be used if the I/O list in the computational module treats it as a logical*1 array. The length of a character variable must be less than 64. The type data is specified by using the first character of the given type followed by an asterisk, followed by the length in bytes. e.g. R*8 X,Y,Z C*27 A,B,C. Variables of the same type are separated by comma's. A blank may be used to delimit a new type specification from the previous variable name. Figure 57 shows an I/O list and type data entered on a list frame. The question marks may be blanked if desired but it is not necessary. The resultant display is a "nest" frame as shown in the bottom of the figure.

A nest frame will appear for each nest of the list. Note that a single variable or a string of non-indexed variables is considered to be a nest with one loop. However, they may not be enclosed in parentheses unless the indexing parameter is explicitly given. e.g., (A,B,C,D,I=1,1).

Hollerith text and data fields are entered on these frames. The area allowed for display purposes is ten lines of 64 positions each. The following notation is used on the nest frames.

- (1) A colon normally denotes the starting position of a data field. However, if a colon appears in position 64 of a line, it denotes a page "segment", not a data field. Segments are used to display the data in a given loop only when the loop index changes. They should be used for records that are to be displayed only, not for records that must be created at a terminal. When this type of display format is used all loops of a nest must appear as segments.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.98
		Rev. 1	Date 2/73

```

0000.TEMPLATE.LATTICE.?.?.KEFF                                <- NAME
(NSOX,(LSOPT,KEFF,BSQD,MSQD,ALPHA,=1,NSOX),=1,1)??????<- LIST
???????????????????????????????????????????????????????? 5 NESTS
???????????????????????????????????????????????????????? 12 LOOPS
???????????????????????????????????????????????????????? 20 VAR. NAMES
R*4 KEFF ?????????????????????????????????????????????????<- TYPE
???????????????????????????????????????????????????????? I(NTEGER)*2,4
???????????????????????????????????????????????????????? R(EAL)*4,8
???????????????????????????????????????????????????????? C(HARACTER)*N
???????????????????????????????????????????????????????? N < 64

```

```

0000.TEMPLATE.LATTICE.?.?.KEFF                                <- NAME
???????????????????????????????????????????????????????? ---NOTATION---
???????????????????????????????????????????????????????? : STARTFIELD
???????????????????????????????????????????????????????? A EBCDIC STRING
???????????????????????????????????????????????????????? I INTEGER
???????????????????????????????????????????????????????? E FLOATING PT.
???????????????????????????????????????????????????????? F FIXED PT.
???????????????????????????????????????????????????????? Z HEXADECEMAL
???????????????????????????????????????????????????????? 1,2..7 INDEX
???????????????????????????????????????????????????????? :: REPEAT LINE
???????????????????????????????????????????????????????? : IN 64-P1 ONLY
(NSOX,(LSOPT,KEFF,BSQD,MSQD,ALPHA,=1,NSOX),=1,1)

```

Figure 57

- (2) Field identifiers are used to indicate the type and length of data fields. The following is a list of identifiers and their definitions.

A - EBCDIC string (FORTRAN A format)
I - Integer (FORTRAN I format)
E - Floating point (FORTRAN E format)
F - Fixed point (FORTRAN F format)
Z - HEXADECIMAL (FORTRAN Z format)
1 thru 7 - Value of a list index

The number of identifiers and associated decimal points define the length of each field as it will appear on the data display.

- (3) A double colon, ":", may be used in positions 1 and 2 of a line. This indicates that the previous line is to be repeated for the rest of the page.

A "page" of text and field identifiers is required for each nest of the list, (page type). All variables in a nest must be defined on one page type. There should be a direct correspondence between the types of the variables in the list to the field identifiers. Index fields must immediately precede their associated data field. (Hollerith text may appear between the fields.)

When a given nest is displayed, a new page of the same type is generated under the following conditions.

- (1) More data exists for the nest.
(2) The upper limit of the inner-most loop that has a variable upper limit is reached.

A new page type will be generated after the data in the current nest is exhausted.

Most errors that are made while creating a template will be diagnosed and a message displayed. The frame on which the error occurred will be re-displayed. If you cannot determine the cause of an error, enter an "S" in the lower left hand corner of the screen, sign off, and seek help.

Figure 58 shows a completed nest frame for the I/O list shown in Figure 57. The list is composed of one nest and two loops and appears at the bottom of the nest frame for quick reference. The variable NSOX was included in the nest so that all data could be displayed with one page type. Line two of the frame contains a description of the variable, NSOX, and an integer data field of length three. The following four lines contain text only, that describes the permissible values of the variable, LSOPT. The colon at the end of line six is used to "segment" the page. When the value of NSOX is greater than three, thus requiring additional pages to be displayed, the data to the left and above the

```

0000.TEMPLATE.LATTICE.?.?.KEFF
NUMBER OF SEARCHES = :III
      LSOPT = 1      - KEFF CALCULATION
                2      - BUCKLING SEARCH
                3      - TIME EIGENVALUE (ALPHA) SEARCH
                4      - CONCENTRATION SEARCH
      LSOPT  KEFF      BUCKLING      M**2      ALPHA
      ----  ----      -
:222  :I  :E.EEEEEEEEE :E.EEEEEEEEE :E.EEEEEEEEE :E.EEEEEEEEE
::
(NSOX,(LSOPT,KEFF,BSQD,MSQD,ALPHA,=1,NSOX),=1,1)

```

Figure 58

colon will not appear on the additional pages.

Lines seven and eight contain column headings for the remaining variables in the list. The first field on line nine, :222, is an index field and does not have a corresponding data value. In this case, the current value of the inner-most index will be displayed. (The outer-most index always assumes the value of 1.) This is followed by an integer data field and four floating point data fields. Note that four positions must be allowed in a floating point field for the exponent notation. The double colon on line ten will cause line nine to be repeated for the rest of the page when the template is displayed. The resultant display is the original select frame.

10.2 The Modify Template Function (MT)

The MT function is used to create a template by modifying a previously created template. It differs from the CT function in the following ways.

- (1) Two arguments may be used. The second argument is the name of the template to be created. If the second argument is omitted the previously created template will be overwritten.
- (2) The notation on the right side of the nest frame does not appear. It is replaced by the following menu.

```
THIS PAGE IS
TO BE -
:_ DELETED
:_ INSERTED
:_ PROCESSED
```

A non-blank character must be placed to the right of one of the colons to indicate the required action.

10.3 The Display Template Function (DT)

The DT function is used to display the template for a specified record type. One argument is required; a record name. (Subtree names may not be used.) The display will differ from the original template input in the following ways.

- (1) Blanks will not appear in the I/O list.
- (2) The variable typing data may appear in a slightly different form.
- (3) A line beginning with a double colon will cause the previous line to be repeated to the end of the page.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.102
		Rev. 1	Date 2/7

11. Creating Monitor Procedures

Monitor procedures are created with the CD function. The record type is SYSTEM.PROC.?.?, where the two variable qualifiers are the procedure name. The text contained in the template for this record type is designed to assist the user as much as possible when creating a procedure record.

A procedure is composed of a series of function requests called steps. The arguments in each step may contain variable qualifiers that are "filled in" for the user when the procedure is executed. These qualifiers start with a question mark and must not exceed eight characters in length, e.g. ?CASE. The values of the variable qualifiers will be obtained from previously defined data records, the active job name, or previously entered qualifiers. The procedure may also contain DO loops where the upper limit of the loop is obtained from a previously defined data record and a special form of an argument for the EX function. The following paragraphs describe each of these features.

- (1) The Special Qualifier, ?\$JOB. Wherever this qualifier appears in a function argument, it will be replaced with the active job name.
- (2) Replacing Qualifiers with Back Referenced Data. All variable qualifiers, except ?\$JOB, have associated control data that indicates where the value of the qualifier is located. These values must be within the first 400 bytes of a data record that has been previously defined in the procedure, and must exist on full word boundaries. The control data consists of the step number that contains the record name, the location of the first value of the qualifier in units of full words, and a full word index to be used if the current step is in a DO loop. (When the value of the index is 2, the next 8 character value will be used.) If the index is set to zero, this implies that the back referenced data record is also in a DO loop and that the values of the qualifier are to be obtained from the same location in different records.
- (3) Double Argument Functions. The MD and CO functions will always be presented with two arguments. The variable qualifiers in the first argument will appear just as defined in the procedure, e.g. ?CASE. The variable qualifiers in the second argument will be replaced by the specified values. When the same variable qualifiers appear in successive steps that are double argument functions, the value entered by the user in the first argument will be carried forward to the next step.
- (4) DO Loops. DO loops are defined by specifying the starting and ending step numbers, the previous step number that contains the name of the record that contains the upper limit of the loop, and the location of the upper limit in units of full words. The loops may not be nested.
- (5) The EX Argument. A series of qualifiers may be used as an argument to the EX function. They will "fill in" most of the fields on the execute frame. They must be entered in the following order. BOX, EST TIME, EST LINES, MEMBER, SPACE, GTIME, GOSIZE, DISP, PROC, RLSE, DUMP, TIMER, SCRATCH. e.g. L014.04.02.RAID.05.02.420.NEW.JOSHUA.YES.NO.OFF.JOB12791.

JOSHUA
DPSTM-500

Title

Monitor System (JMS)

Vol.

II

Page

E.1.103

Rev.

1

Date

2/73

- (6) The SK Function. When the SK function is used in a procedure the select frame for the step will not be shown.

The template for the procedure record contains four page types as shown in Figures 59 and 60. Figures 61 thru 63 show a procedure that is used to create the necessary data records and execute the computational module, GAUGE.

0000.TEMPLATE.SYSTEM.PROC.?.?

THE FOLLOWING LIMITS MUST BE OBSERVED.

NUMBER OF STEPS - 25

NUMBER OF VARIABLE QUALIFIERS - 20

NUMBER OF DO LOOPS - 12

THE VALUE OF THE UPPER LIMIT OF A DO LOOP - 9

:II NUMBER OF PROCEDURE STEPS

:II NUMBER OF UNIQUE VARIABLE QUALIFIERS(DO NOT INCLUDE ?\$JOB)

:II NUMBER OF DO LOOPS IN THE PROCEDURE

PAGE 2

0000.TEMPLATE.SYSTEM.PROC.?.?

:AAAAAAA FUNCTION (PROCEDURE STEP :11)

:AA

:AAAAAAA FUNCTION (PROCEDURE STEP :11)

:AA

:AAAAAAA FUNCTION (PROCEDURE STEP :11)

:AA

:AAAAAAA FUNCTION (PROCEDURE STEP :11)

:AA

:AAAAAAA FUNCTION (PROCEDURE STEP :11)

:AA

PAGE 3

Figure 59

0000.TEMPLATE.SYSTEM.PROC.?.?

ENTER ALL VARIABLE QUALIFIERS THAT WERE USED IN DEFINING
RECORD NAMES WITH THE EXCEPTION OF ?\$JOB.

:AAAAAAA VARIABLE QUALIFIER :11

:II PROCEDURE STEP THAT DEFINES THE NAME OF THE RECORD THAT
CONTAINS THE VALUE(S) OF THE VARIABLE QUALIFIER.

:II LOCATION OF THE FIRST VALUE OF THE VARIABLE QUALIFIER IN
THE DATA RECORD IN UNITS OF FULL WORDS(4 BYTES/WORD).

:II INCREMENT(IN FULL WORDS) TO FIND THE NEXT VALUE OF THE
VARIABLE QUALIFIER IF IT APPEARS IN A DO LOOP.

PAGE 4

0000.TEMPLATE.SYSTEM.PROC.?.?

ENTER CONTROL DATA FOR EACH DO LOOP. IF THE PROCEDURE HAS NO
LOOPS, IGNORE THIS PAGE.

-----DO STEP M THRU STEP N, WHERE I=1,L-----

:II M (FIRST PROCEDURE STEP OF LOOP:11)

:II N (LAST PROCEDURE STEP OF LOOP:11)

:II PROCEDURE STEP THAT DEFINES THE NAME OF THE RECORD THAT
CONTAINS THE VALUE OF THE UPPER LIMIT, L.

:II LOCATION OF L IN THE DATA RECORD IN UNITS OF FULL WORDS.

LASTPAGE 5

Figure 60

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.106
		Rev. 1	Date 2/73

SYSTEM.PROC.CREATE.GAUGE
THE FOLLOWING LIMITS MUST BE OBSERVED.
NUMBER OF STEPS - 25
NUMBER OF VARIABLE QUALIFIERS - 20
NUMBER OF DO LOOPS - 12
THE VALUE OF THE UPPER LIMIT OF A DO LOOP - 9

: 4 NUMBER OF PROCEDURE STEPS
: 2 NUMBER OF UNIQUE VARIABLE QUALIFIERS(DO NOT INCLUDE ?\$JOB)
: 1 NUMBER OF DO LOOPS IN THE PROCEDURE

PAGE 1

SYSTEM.PROC.CREATE.GAUGE
:CD FUNCTION (PROCEDURE STEP : 1)
:INPUT.GAUGE.JOB.?\$JOB
:CD FUNCTION (PROCEDURE STEP : 2)
:INPUT.GAUGE.CASE.?CASE
:CD FUNCTION (PROCEDURE STEP : 3)
:INPUT.GAUGE.?GEOM
:EX FUNCTION (PROCEDURE STEP : 4)
:L000.002.02.GAUGE.01.002.240.NEW.JOSHUA.YES.NO.OFF.?\$JOB

PAGE 2

Figure 61

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.107
		Rev. 1	Date 2/73

SYSTEM.PROC.CREATE.GAUGE

ENTER ALL VARIABLE QUALIFIERS THAT WERE USED IN DEFINING
RECORD NAMES WITH THE EXCEPTION OF ?\$JOB.

- ```

: ?CASE VARIABLE QUALIFIER : 1
: 1 PROCEDURE STEP THAT DEFINES THE NAME OF THE RECORD THAT
 CONTAINS THE VALUE(S) OF THE VARIABLE QUALIFIER.
: 2 LOCATION OF THE FIRST VALUE OF THE VARIABLE QUALIFIER IN
 THE DATA RECORD IN UNITS OF FULL WORDS(4 BYTES/WORD).
: 2 INCREMENT(IN FULL WORDS) TO FIND THE NEXT VALUE OF THE
 VARIABLE QUALIFIER IF IT APPEARS IN A DO LOOP.

```

PAGE 3

SYSTEM.PROC.CREATE.GAUGE

ENTER ALL VARIABLE QUALIFIERS THAT WERE USED IN DEFINING  
RECORD NAMES WITH THE EXCEPTION OF ?\$JOB.

- ```

: ?GEOM    VARIABLE QUALIFIER    : 2
: 2 PROCEDURE STEP THAT DEFINES THE NAME OF THE RECORD THAT
    CONTAINS THE VALUE(S) OF THE VARIABLE QUALIFIER.
: 18 LOCATION OF THE FIRST VALUE OF THE VARIABLE QUALIFIER IN
    THE DATA RECORD IN UNITS OF FULL WORDS(4 BYTES/WORD).
: 0 INCREMENT(IN FULL WORDS) TO FIND THE NEXT VALUE OF THE
    VARIABLE QUALIFIER IF IT APPEARS IN A DO LOOP.

```

PAGE 4

Figure 62

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.108
		Rev. 1	Date 2/73

SYSTEM.PROC.CREATE.GAUGE

ENTER CONTROL DATA FOR EACH DO LOOP. IF THE PROCEDURE HAS NO
LOOPS, IGNORE THIS PAGE.

-----DO STEP M THRU STEP N, WHERE I=1,L-----

: 2 M (FIRST PROCEDURE STEP OF LOOP: 1)

: 3 N (LAST PROCEDURE STEP OF LOOP: 1)

: 1 PROCEDURE STEP THAT DEFINES THE NAME OF THE RECORD THAT
CONTAINS THE VALUE OF THE UPPER LIMIT, L.

: 1 LOCATION OF L IN THE DATA RECORD IN UNITS OF FULL WORDS.

LASTPAGE 5

Figure 63

12. Data Protection Facilities

Data may be protected from unauthorized access by assigning it a "file category" other than zero. Each category has an associated password that must be entered on the sign-on frame opposite the words, FILE ID. (See Figure 1) When a file ID is specified, a value must also be entered in the field labeled FILE CATEGORY. This value is used to determine the category of newly created data records when no other information is supplied. Contact the Computer Operations Division for information concerning authorization procedures.

The data is protected on a "record level" basis, i.e., a check is made each time a record is retrieved or generated. This feature allows JOSHUA data sets to be composed of combinations of protected categories, the most common being one unprotected category in conjunction with one protected category.

12.1 Data Retrieval

Protected data records may be viewed only when a proper File ID is supplied and the requestor is authorized to access the retrieved data category. Each time a record is retrieved and the above criteria are not met, the system treats the request as if the requested record did not exist. When a protected record is displayed, each frame of the display contains the protection category on the top and bottom lines of the CRT.

Example:

2600.GLOSSARY.AEC.FIS.COST

1

NUMBER OF GLOSSARY NAMES : 23

NUMBER	GLOSSARY NAME	FIELD LENGTH	COLUMN HEADING
: 1	:OFFICE	: 2	:OFC
: 2	:RPTGUNIT	: 1	:RU
: 3	:SITE	: 1	:S
: 4	:BSC	: 4	:BSC
: 5	:STATUS	: 2	:SC
: 6	:ASSET	: 3	:AT
: 7	:FUND	: 2	:FT
: 8	:SUMCLASS	: 2	:SCC

1

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.110
		Rev. 1	Date 2/73

12.2 Explicit Requests

Records of a specific category may be retrieved or generated with an explicit request. This is accomplished by adding a special qualifier to the argument of the request. The qualifier contains two asterisks followed by the desired file category, e.g. **1. The following request would display only the category one records from the INPUT data set.

```
DD      -- FUNCTION
INPUT.**1.....
.....
```

12.3 Data Generation

Protected data records may be generated only when a proper File ID is supplied, the user is authorized to access the requested data category, and the referenced data set contains an extent corresponding to the requested data category. If the above criteria are not met, the system displays a message indicating that an improper write request was received, or that the requested data set extent does not exist.

The category of the generated data is determined by the following hierarchy of rules:

1. If the request is explicit, that category is used.
2. If the request is not explicit, the output record will be the same category as the input record.
3. If the request is not explicit and an input record was not used, the output category will be that specified on the "Sign-On" frame.

Example: Assume that the record A.B exists in category 0 and that the category on the "Sign-On" frame is 0. The following request would create a record in category 1.

```
CD      -- FUNCTION
A.B.**1.....
.....
```

12.4 Template Protection

The capability has been provided to protect templates as well as data. However, the normal mode of operation is to protect only the data. If it is necessary to protect a template, it should be emphasized that a protected template can only be used to display unprotected data or data of the same category as the template.

Example: TEMPLATE.A.? exists in category 1 and the data A.B exists in category 0 or 1. This data, A.B, can be displayed. However, if data A.B existed in category 2, an error message would result when the display is requested.

12.5 Changing the File Category

The category of a record or subtree may be changed by using the COPY function and supplying the new category as the first qualifier of the second argument on the "Select" frame.

Example: CO -- FUNCTION
INPUT.**1.....
**0.....

This function request will change all category 1 INPUT records to category zero. Note that the category of a record cannot be changed without using an explicit request.

12.6 Determining the File Category

The LIST function is used to determine the file category of data records. Two asterisks followed by the category number will appear to the left of each record name for those records not in category zero.

12.7 The EX Function

When a file ID has been specified at sign-on time, the execute frame will contain an additional field labeled FILE CATEGORY. A value must always be entered in this field. If protected data will not be accessed by the batch job a value of zero should be entered. The default value in the BOX field will be HOLD rather than L000, indicating that the printed output is to be held by operations rather than delivered to the users output box. Also, if YES is specified in the DUMP field, the printed output will be held even if the HOLD value is overridden on the execute frame.

12.8 The PR Function

All output from the PR function will be held by operations unless an explicit request is used. The following example illustrates how to direct the output to a users output box.

PR -- FUNCTION
L001.**0.....
.....

12.9 Data Set Maintenance

All monitor jobs that maintain data sets containing protected data must be submitted by operations as "special handling". The MAINTAIN function may be used to maintain data sets that contain protected data if no more than one non-zero category exists. When a data set contains more than one non-zero category, each category must be copied to a Scratch data set and dumped to tape in a separate job step. These sequential data sets

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.112
		Rev. 1	Date 2/73

may then each be merged with the JOSHUA data set in separate job steps, after the data set has been deleted.

When a data set contains both protected and unprotected data, two sequential files are created. Each file will be a complete data set of a single category and contain no extents. The protected data set will be written to unit 10 and the unprotected data to unit 11. An additional DD card must be supplied for unit 11, (DD name = FT11F001). The file category of the protected data must be specified on a CATEGORY= card. Figure 64 shows sample JCL and input data necessary to maintain a protected data set. In this sample, the two sequential files are put on one tape volume.

The output contains a list of both files, produced by the LIST function before the data is moved to the sequential data sets. The record names in both listings will be labeled as protected since the same Scratch data set is used for both files.

The MERGE function must be used to move sequential data sets into the data base if the JOSHUA data set contains both protected and unprotected data. The protected data will be read from unit 12 and the unprotected data from unit 13. An additional DD card must be supplied for unit 13, (DD name = FT13F001). The file category of the protected file must be specified on a CATEGORY= card. The JCL and input data required to move the two sequential files produced in Figure 64 back to the JOSHUA data base is the same as shown in Figure 64 with the MA function replaced by the ME function.

JOSHUA DPSTM-500	Title Monitor System (JMS)	Vol. II	Page E.1.113
		Rev. 1	Date 2/73

```
//JDD10000 JOB (0000,L021,2,2,,,,,8690-00,,T,00),JDD,CLASS=M
/*SETUP          LC0001
//STEP1 EXEC JOSHUA,SSPACE=14,SPACE=0
//GO.FT10F001 DD UNIT=2400,VOL=SER=LC0001,DISP=(,KEEP),LABEL=1,
// DCB=(RECFM=VBS,LRECL=800,BLKSIZE=8004,BUFNO=1),DSN=P0000
//GO.FT11F001 DD UNIT=2400,VOL=SER=LC0001,DISP=(,KEEP),LABEL=2,
// DCB=(RECFM=VBS,LRECL=800,BLKSIZE=8004,BUFNO=1),DSN=U0000
//GO.SYSIN DD *
CATEGORY=1
S/E
MA
0000
S/E
/*
```

Figure 64

JOSHUA

DPSTM-500

Title

An Example - Part IV

Vol.

II

Page

E.2.1

Rev.

Date

2/73

Work Done By

F. D. Knight

H. C. Honeck

Report Written By

F. D. Knight

H. C. Honeck

Contents

	<u>Page</u>
1. Introduction	E.2.3
2. Sign-On Procedure	E.2.5
3. Input Templates	E.2.9
4. Output Templates	E.2.21
5. Input Preparation	E.2.31
6. Job Execution	E.2.41
7. Display of Output	E.2.49
8. Sign-Off Procedure	E.2.65

JOSHUA DPSTM-500	Title	An Example - Part IV	Vol. II	Page E.2.2
			Rev.	Date 2/73

(Blank)

1. Introduction

In previous sections, the CØG module has been defined, written, precompiled, compiled, and linked into a JOSHUA program library. This section will guide the user through the necessary sequence of steps to gain useful results from CØG while at a terminal. In this sample sequence, it will be assumed that no previous input data records or templates exist. The data record names are as defined in Section A.2.

The preparation of templates is generally not necessary for a JOSHUA user; however, Sections 3 and 4 of Part IV may be useful for gaining a better understanding of the JOSHUA terminal system operation.

JOSHUA DPSTM-500	Title An Example - Part IV	Vol.	Page
		II	E.2.4
		Rev.	Date
			2/77

.....BADGE	TUBE 035
- ..COST CODE	
.....USER DATA SET	
...USER ID	
...FILE ID	
.....FILE CATEGORY	

2791.....BADGE	TUBE 035
8690-00..COST CODE	
3871.....USER DATA SET	
ABCDEF...USER ID	
...FILE ID	
.....FILE CATEGORY	

2. Sign-On Procedure

The top figure on the opposite page is a "Sign-On" frame and will appear on the terminal screen when JTS is ready for use. When the requested information is entered, the shift and enter keys should be depressed to cause transmission of the image to the computer. Note that these keys must always be depressed to proceed with any operation.

The bottom figure shows the sign-on frame after the user has entered the proper information.

JOSHUA TERMINAL SYSTEM STATUS

- 1 - JOB DATA SET SPACE IS AT A PREMIUM. PLEASE DELETE JOBS
AS SOON AS POSSIBLE.
- 2 - THE 'CPUT' ROUTINE HAS BEEN FIXED. YOU MAY NOW USE THE
TIMER OPTION.

TEMP2791 - 3871 - STD ---- 3871

TUBE 035

--FUNCTION

.....
.....

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.7
		Rev.	Date 2/73

The top figure on the opposite page will appear immediately after a successful sign-on procedure. It will contain information that is pertinent to the operation of JTS at that given time.

The bottom figure on the opposite page contains the next frame that will appear. This is the "Select Frame" and will be used to enter all function requests.

TEMP2791 - 3871 - STD ---- 3871 TUBE 035
CT --FUNCTION
INPUT.COG.?.....
.....

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.9
		Rev.	Date 2/73

3. Input Templates

The first step is to create an input template for CØG. The name of this template and the associated list have been specified when structuring the module back in Sections A and B. The request for creating this template (CT) is shown on the opposite page. The first line has the function request and the second line gives the template name, INPUT.CØG.? (Note that this name defines a data record type.) The variable qualifier (?) allows this template to be used for many input records.

3871.TEMPLATE.INPUT.COG.?

??

??

??

????????????????????Y??

??

??

??

??

??

3871.TEMPLATE.INPUT.COG.?

VERSN,NLX,(DLAT,VLAT,N=1,NLX)

R*8 VERSN,DLAT,VLAT

The top figure on the opposite page is the first frame that appears after requesting the CT function. It contains question marks in positions that may be used for data input, i.e., the I/O list and non-default variable types associated with this template. Although not shown here, a legend appears to the right of all template frames during a CT operation to guide the user in filling out the frame. At the top of the frame is the full data record name of this template. Note that the user data set (3871) and the qualifier TEMPLATE have been appended to the three user supplied qualifiers.

The bottom figure shows a completed template frame with proper list and type information. The list contains two nests: VERSN, NLX; and (DLAT,VLAT,N=1,NLX) with VERSN,DLAT, and VLAT typed as REAL*8.

3871.TEMPLATE.INPUT.COG.?

??
??
??
??
??
??
??
??
??
??
??
??
??
??
??
VERS,NLX

3871.TEMPLATE.INPUT.COG.?

:AAAAAAA MULTIGRP VERSION NAME
:II NUMBER OF LATTICES

VERS,NLX

JOSHUA**DPSTM-500****Title**

An Example - Part IV

Vol.

II

Page

E.2.13

Rev.**Date**

2/73

For each nest specified in the list, a frame is displayed for the purpose of creating the format for that nest. The nest to which the frame applies is shown on the bottom line of the frame. This is illustrated by the top figure on the opposite page.

The bottom figure illustrates the frame as filled in by the user in which the MULTIGRP version name and the number of lattices will serve as partial input to CØG. The colons in this case are field identifiers for the beginning of an 8-character EBCDIC value and a 2-digit integer value. The fields are terminated when a blank is encountered. In this case, the remainder of each line is Hollerith text.

3871.TEMPLATE.INPUT.COG.?

??
??
??
??
??
??
??
??
??
??
??
??
??
??

(DLAT,VLAT,N=1,NLX)

3871.TEMPLATE.INPUT.COG.?

LATTICE

NAME VERSION

:AAAAAAAA :AAAAAAAA

::

(DLAT,VLAT,N=1,NLX)

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.15
		Rev.	Date 2/73

The opposite page shows the format frame for the second nest before and after it has been completed. The input data will be the name and version of the lattice records desired up to the number specified in the first nest by NLX. The double colon indicates that the previous line is to be repeated for the rest of this frame. This completes the data necessary to generate this template.

JOSHUA
DPSTM-500

Title

An Example - Part IV

Vol.

II

Page

E.2.16

Rev.

Date

2/73

TEMP2791 - 3871 - STD ---- 3871

TUBE 035

DT --FUNCTION

INPUT.COG.?.....

.....

3871.TEMPLATE.INPUT.COG.?

VERSN,NLX,(DLAT,VLAT,N=1,NLX)

R*8 VERSN R*8 DLAT R*8 VLAT

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.17
		Rev.	Date 2/73

Now that the input template has been created, it should be displayed to verify its correctness. This is accomplished with the DT function as shown on the opposite page. The bottom figure shows the first page of the input template, which differs from that entered by the user in that the type statement has been changed in format and the page of the template is given.

3871.TEMPLATE.INPUT.COG.?

:AAAAAAA MULTIGRP VERSION NAME

:II NUMBER OF LATTICES

PAGE 2

3871.TEMPLATE.INPUT.COG.?

LATTICE

NAME	VERSION
------	---------

:AAAAAAA	:AAAAAAA
----------	----------

:AAAAAAA	:AAAAAAA
----------	----------

:AAAAAAA	:AAAAAAA
----------	----------

:AAAAAAA	:AAAAAAA
----------	----------

:AAAAAAA	:AAAAAAA
----------	----------

:AAAAAAA	:AAAAAAA
----------	----------

:AAAAAAA	:AAAAAAA
----------	----------

:AAAAAAA	:AAAAAAA
----------	----------

PAGE 3

JOSHUA

DPSTM-500

Title

An Example - Part IV

Vol.

II

Page

E.2.19

Rev.

Date

2/73

The formats for the first and second nests of the list are shown on the opposite page. They differ from those entered by the user in that the nests of the list are not shown and page numbers are included. In the second nest format, the double colon which was entered by the user has been expanded to repeat the previous line for the rest of the page.

3871.TEMPLATE.ONEGRP.?.SPECTRA
NCSX, (DLAT, VLAT, BSQD, N=1, NCSX)

R*8 DLAT R*8 VLAT

PAGE 1

3871.TEMPLATE.ONEGRP.?.SPECTRA

:III - NUMBER OF CASES

A CASE IS A PARTICULAR COMBINATION OF LATTICE NAME,
LATTICE VERSION, AND BUCKLING VALUE.

PAGE 2

4. Output Templates

There are three templates necessary to display the results generated by CØG. These are created similarly to the input template and will be shown as they exist after creation.

The first is shown on the opposite page as 3871.TEMPLATE.ØNEGRP.?.SPECTRA. The variable qualifier is replaced when displaying data by the MULTIGRP version name specified in the input record. The list consists of two nests with DLAT and VLAT typed as REAL*8 variables. The bottom figure specifies the number of cases and includes tutorial information as Hollerith text.

3871.TEMPLATE.ONEGRP.?.SPECTRA

LAT.NAME	LAT.VERS	BUCKLING(CM-2)
-----	-----	-----
:AAAAAAAA	:AAAAAAAA	:E.EEEEEEEEE
:AAAAAAAA	:AAAAAAAA	:E.EEEEEEEEE
:AAAAAAAA	:AAAAAAAA	:E.EEEEEEEEE
:AAAAAAAA	:AAAAAAAA	:E.EEEEEEEEE
:AAAAAAAA	:AAAAAAAA	:E.EEEEEEEEE
:AAAAAAAA	:AAAAAAAA	:E.EEEEEEEEE
:AAAAAAAA	:AAAAAAAA	:E.EEEEEEEEE
:AAAAAAAA	:AAAAAAAA	:E.EEEEEEEEE

PAGE 3

3871.TEMPLATE.ONEGRP.?.ISOTOPES

NIRX, (DISO, ITDC, REAC, N=1, NIRX)

R*8 DISO R*8 REAC

PAGE 1

JOSHUA

DPSTM-500

Title

An Example - Part IV

Vol.

II

Page

E.2.23

Rev.

Date

2/73

The third page of the template has heading information and three columns of data. Two columns will contain the appropriate lattice name and version from the input record, while the third column will contain Buckling results from the KEFF record for these lattices.

The bottom frame contains the first page of the template 3871.
TEMPLATE.ØNEGRP.?.ISØTØPES. The variable qualifier is treated as before. The list consists of two nests and has two variables typed as REAL*8.

3871.TEMPLATE.ONEGRP.?.ISOTOPEs

:III - NUMBER OF CASEs

A CASE IS A PARTICULAR COMBINATION OF ISOTOPE NAME,
TEMPERATURE, AND REACTION TYPE.

PAGE 2

3871.TEMPLATE.ONEGRP.?.ISOTOPEs

ISOTOPE	TEMP	REACTION
---------	------	----------

:AAAAAAA	:IIII	:AAAAAAA
:AAAAAAA	:IIII	:AAAAAAA
:AAAAAAA	:IIII	:AAAAAAA
:AAAAAAA	:IIII	:AAAAAAA
:AAAAAAA	:IIII	:AAAAAAA
:AAAAAAA	:IIII	:AAAAAAA
:AAAAAAA	:IIII	:AAAAAAA
:AAAAAAA	:IIII	:AAAAAAA

PAGE 3

JOSHUA**DPSTM-500****Title**

An Example - Part IV

Vol.

II

Page

E.2.25

Rev.**Date**

2/73

Page two of the template contains the number of cases and tutorial information as Hollerith text. Page three of the template contains heading information and three result columns for isotope, temperature, and reaction combinations.

3871.TEMPLATE.ONEGRP.?.XSECTION.?.?.?
NCSX, (SIG,N=1,NCSX)

PAGE 1

3871.TEMPLATE.ONEGRP.?.XSECTION.?.?.?
:III - NUMBER OF CASES
A CASE IS A PARTICULAR LATTICE-BUCKLING COMBINATION

PAGE 2

JOSHUA DPSTM-500	Title	An Example - Part IV	Vol. II	Page E.2.27
			Rev.	Date 2/73

Page one of the next template 3871.TEMPLATE.ØNEGRP.?.XSECTION.???.? is shown in the top figure on the opposite page. This template has two nests and all variables are typed by default. The first variable qualifier is the same as previously defined. The second, third, and fourth variable qualifiers represent isotopes, temperatures, and reaction types, respectively. Page two of the template contains the number of cases considered and Hollerith text.

3871.TEMPLATE.ONEGRP.?.XSECTION.?.?.?

AVERAGE ONE GROUP CROSS SECTION BY CASE NUMBER

:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE
:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE
:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE
:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE
:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE
:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE
:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE
:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE
:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE	:III	:E.EEEEEEEEE

PAGE 3

Page three of the template contains the average one-group cross sections by case number. There are three columns of data and the current value of the index will be explicitly displayed. Numeric values following a field identifier are used to display the current value of an index.

TEMP2791 - 3871 - STD ----- 3871
TUBE 035
LI --FUNCTION
LATTICE.....
.....

3871.LATTICE
11604 PATTERN .HOMO .
11605 .BI-SEP .
11606 .TRI-SEPO.
11607 .ANYTHING.
11608 .TRI-SEP .
11609 MK-14 .CYL .FISSION .1 .
11610 . . .2 .
11611 . .FLUX .MK-14 .TEST .1 .
11613 . .INPUT .
11614 . .CHANGES .

PAGE 1

5. Input Preparation

The templates for input and output have been created. The next step is to prepare an input record. The input record requires the number of lattice name and version combinations for which the program CØG is to form the one-group library, and then the respective lattice name and version combinations. To get these input values we need to know what data are available on the LATTICE data set and select from these the ones we want to use. We first ask what data is available, i.e., what lattices have both KEFF and LFLUX records?

This is done with the LI function as shown in the top figure on the opposite page.

The bottom figure shows page one of this catalog and it is evident that no KEFF record exists on the page. Note that blank qualifiers appearing in a record name indicate that it is identical to the last non-blank qualifier at that level.

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.32
		Rev.	Date 2/73

3871.LATTICE

```

11617 MK-14 .CYL .
11618 . .LFLUX .1 .
11623 . .GEOMETRY. MK-1.4 .
11625 MARK5RBU.BURNUP1 .INPUT .
11626 . .CHANGES .
11629 . .KEFF .
11630 . .GEOMETRY. MARK.5R .
12392 ED14-30A.DTEST .INPUT .
12393 . .CHANGES .
12396 . .KEFF .

```

PAGE 2

3871.LATTICE

```

12397 ED14-30A.DTEST .LFLUX .1 .
12402 . . .2 .
12407 . . .3 .
12412 . . .4 .
12417 . . .5 .
12422 . .GEOMETRY.SEP-NROD.TEST .
12423 . .MK-14 .TEST .
12424 . .MK-30A .TEST .
12425 . .FISSION .1 .
12426 . . .2 .

```

PAGE 3

JOSHUA**DPSTM-500****Title**

An Example - Part IV

Vol.

II

Page

E.2.33

Rev.**Date**

2/73

Pages 2 and 3 of the list show that lattices MK-14.CYL and ED14-30A.DTEST have KEFF and LFLUX records (lines 11617-8 and 12396-12417). We will choose these two lattices and note that since the latter lattice has 5 LFLUX records (one for each of 5 separate Bucklings), we have selected a total of 6 lattice/Buckling combinations.

The MARK5RBU.BURNUP1 records (lines 11625 - 11630) are not used since there are no LFLUX records.

JOSHUA DPSTM-500	Title	An Example - Part IV	Vol. II	Page E.2.34
			Rev.	Date 2/73

TEMP2791 - 3871 - STD ---- 3871
TUBE 035

CD
--FUNCTION

INPUT.COG.RUN12791.....
.....

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.35
		Rev.	Date 2/73

In order to create the input record for CØG, the CD function is executed with a data record name of INPUT.CØG.RUN12791, where RUN12791 will be the JOSHUA job name for job execution. If this record does not exist, the template, 3871.TEMPLATE.INPUT.CØG.?, will be displayed. If the record does exist, the actual data will be formatted using the above template. In the latter case the user need enter only the values he wishes to change.

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.36
		Rev.	Date 2/73

3871.TEMPLATE.INPUT.COG.RUN12791

:AAAAAAA MULTIGRP VERSION NAME

:II NUMBER OF LATTICES

PAGE 1

3871.INPUT.COG.RUN12791

:STANDARD MULTIGRP VERSION NAME

: 2 NUMBER OF LATTICES

PAGE 1

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.37
		Rev.	Date 2/73

In the top figure, the CD function displays the frame corresponding to the first nest of the template. The user enters specific values in all places indicated by a field identifier, i.e., a colon. The bottom figure shows that the user wants to use the STANDARD version of the MULTIGRP data set and desires to run this problem for two lattices.

3871.TEMPLATE.INPUT.COG.RUN12791

LATTICE

NAME	VERSION
:AAAAAAAA	:AAAAAAAA
:AAAAAAAA	:AAAAAAAA
:AAAAAAAA	:AAAAAAAA
:AAAAAAAA	:AAAAAAAA
:AAAAAAAA	:AAAAAAAA
:AAAAAAAA	:AAAAAAAA
:AAAAAAAA	:AAAAAAAA
:AAAAAAAA	:AAAAAAAA

PAGE 2

3871.INPUT.COG.RUN12791

LATTICE

NAME	VERSION
:MK-14	:CYL
:ED14-30A	:DTEST

PAGE 2

JOSHUA DPSTM-500	Title	An Example - Part IV	Vol.	Page
			II	E.2.39
			Rev.	Date
				2/73

The top figure is the second nest of the template. It provides space for the user to enter name and version combinations. The bottom figure shows the two lattice name and version combinations requested by the user to be used as input. These are the same lattice names and versions found by listing the LATTICE catalog.

JOSHUA
DPSTM-500

Title

An Example - Part IV

Vol.

II

Page

E.2.40

Rev.

Date

2/73

TEMP2791 - 3871 - STD ---- 3871

TUBE 035

SJ --FUNCTION

RUN12791.....

.....

RUN12791-IS NOW THE ACTIVE JOB NAME TUBE 035

JOB DATA SET WAS NOT LOCATED - ASSUME NEW JOB

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.41
		Rev.	Date 2/73

6. Job Execution

The input for CØG is now complete. In order to execute the module, it is necessary to specify the JOSHUA job name to be used in the batch run. This is done with the set job or SJ function. In the top figure on the opposite page, the SJ function is used to request that the job name be set to RUN12791. The bottom frame illustrates the messages generated by the SJ request. The message says that RUN12791 is now the active job name. It will therefore be used in the batch as the name of the JØB data set where new records will be written by the module. The message also specifies that no job by that name currently exists in the JOSHUA system. With the job set to RUN12791, this active job name will appear on line one of each select frame. Prior to this time, the active job name had been set to TEMP2791 by the system.

TEMP2791 is a combination of the word TEMP and the users badge number.

JOSHUA
DPSTM-500

Title

An Example - Part IV

Vol.
II

Page
E.2.42

Rev.

Date
2/73

RUN12791 - 3871 - STD ---- 3871

TUBE 035

EX --FUNCTION

.....
.....

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.43
		Rev.	Date 2/73

The figure on the opposite page shows the request for a problem execution, i.e., the function EX.


```
:L000.....BOX      :001...EST TIME    :01...EST LINES  JOB PARAMETERS
:
:      .....MEMBER  :01....SPACE      :001..GTIME      EXEC PARAMETERS
:JOS.PGMS01 ..PDS    :140K..GOSIZE      :NEW..DISP
:JOSHUA .....PROC   :YES...RLSE        :NO...DUMP
:OFF.....TIMER     :010...BUFNO       :001..SSPACE
:RUN12791.....JOB
:NO.....TAPES       :NO....INPUT
:
:      ..:         ..:         ..:         .. DATA SETS
:
:      .....SCRATCH
```

```
:L001.....BOX      :001...EST TIME    :01...EST LINES  JOB PARAMETERS
:
:COG      ....MEMBER :02....SPACE      :001..GTIME      EXEC PARAMETERS
:JOS.PGMS01 ..PDS    :240K..GOSIZE      :NEW..DISP
:JOSHUA .....PROC   :YES...RLSE        :NO...DUMP
:OFF.....TIMER     :010...BUFNO       :000..SSPACE
:RUN12791.....JOB
:NO.....TAPES       :NO....INPUT
:
:      ..:         ..:         ..:         .. DATA SETS
:
:      .....SCRATCH
```

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.45
		Rev.	Date 2/73

The top figure shows the frame as it appears after initiating the EX function. This frame is used to enter the information necessary for JOSHUA to prepare the JCL cards necessary for a batch job. As can be seen, most of the parameters have default values and need not be entered unless an override is desired.

The bottom figure shows the same frame after the user has entered the necessary values and overridden the undesirable defaults.

JOSHUA DPSTM-500	Title	An Example - Part IV	Vol.	Page
			II	E.2.46
			Rev.	Date
				2/73

HASP JOB NUMBER 89 AWAITING EXECUTION

PAGE 2

RUN12791 - 3871 - STD ---- 3871 TUBE 035

ST --FUNCTION

89.....

.....

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.47
		Rev.	Date 2/73

The top figure indicates that the program to execute the CØG module has been assigned job number 89 by HASP and is awaiting execution.

The bottom figure indicates the use of the status of ST function. It is used to determine when job 89 has completed. The result of the ST function can be one of the following messages:

- (1) JOB 89 AWAITING EXECUTION
- (2) JOB 89 EXECUTION COMPLETE
- (3) JOB 89 NOT LOCATED

Messages 2 and 3 both indicate job completion. Message 2 indicates that the job has not yet been printed. Message 3 indicates the job is no longer in the system.

RUN12791 - 3871 - STD ----- 3871 TUBE 035

LI --FUNCTION

ONEGRP.....
.....

RUN12791.ONEGRP

5221	STANDARD.SPECTRA .		
5222	.XSECTION.H-H20	.20	.TOTAL .
5223	. . .	.	.CAPTURE .
5224	. . .	.	.TRANSPRT.
5225	. . .	.60	.TOTAL .
5226	. . .	.	.CAPTURE .
5227	. . .	.	.TRANSPRT.
5228	. . .	.90	.TOTAL .
5229	. . .	.	.CAPTURE .
5230	. . .	.	.TRANSPRT.

JOSHUA DPSTM-500	Title An Example - Part	Vol. II	Page E.2.49
		Rev.	Date 2/73

7. Output

When either the second or third messages are displayed for job 89, the job results are ready to be viewed. If the user knows which named data records are to be displayed, he can use the data display or DD function as shown a few pages from here. However, if the data record names are not known, then a list or LI of the JOB data set catalog for RUN12791.ØNEGRP is necessary. The execution of the LI function for ØNEGRP in the RUN12791 data set is shown in the top figure with page one of the list shown at the bottom.

The three output data records that will be displayed in this example are:

RUN12791.ØNEGRP.STANDARD.SPECTRA

RUN12791.ØNEGRP.STANDARD.ISOTOPES

RUN12791.ØNEGRP.STANDARD.XSECTION.H-H2Ø.20

It is not necessary in these display and list requests to enter the job name as part of the record name since it will automatically be appended by the system. These three display requests and output displays follow.

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.50
		Rev.	Date 2/73

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.50
		Rev.	Date 2/73

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.50
		Rev.	Date 2/73

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.50
		Rev.	Date 2/73

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.50
		Rev.	Date 2/73

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.50
		Rev.	Date 2/73

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.50
		Rev.	Date 2/73

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.50
		Rev.	Date 2/73

```

RUN12791 - 3871 - STD ---- 3871          TUBE 035
DD      --FUNCTION
ONEGRP.STANDARD.SPECTRA.....
.....

```

```

RUN12791 - 3871 - STD ---- 3871          TUBE 035
DD      --FUNCTION
ONEGRP.STANDARD.SPECTRA.....
.....

```

```

RUN12791 - 3871 - STD ---- 3871          TUBE 035
DD      --FUNCTION
ONEGRP.STANDARD.SPECTRA.....
.....

```

```

RUN12791 - 3871 - STD ---- 3871          TUBE 035
DD      --FUNCTION
ONEGRP.STANDARD.SPECTRA.....
.....

```

```

RUN12791 - 3871 - STD ---- 3871          TUBE 035
DD      --FUNCTION
ONEGRP.STANDARD.SPECTRA.....
.....

```

JOSHUA DPSTM-500	Title	An Example - Part IV	Vol. II	Page E.2.51
			Rev.	Date 2/73

This figure shows the request for a display of the ØNEGRP.STANDARD.SPECTRA data record in the RUN12791 data set. The results for this request are shown on the following page in the form indicated by the previously defined template.

RUN12791.ONEGRP.STANDARD.SPECTRA

: 6 - NUMBER OF CASES

A CASE IS A PARTICULAR COMBINATION OF LATTICE NAME,
LATTICE VERSION, AND BUCKLING VALUE.

PAGE 1

RUN12791.ONEGRP.STANDARD.SPECTRA

LAT.NAME	LAT.VERS	BUCKLING(CM-2)
----------	----------	----------------

-----	-----	-----
:MK-14	:CYL	: .00000E 00

:ED14-30A	:DTEST	: -.25000E-03
-----------	--------	---------------

:ED14-30A	:DTEST	: .00000E 00
-----------	--------	--------------

:ED14-30A	:DTEST	: .25000E 01
-----------	--------	--------------

:ED14-30A	:DTEST	: .50000E 01
-----------	--------	--------------

:ED14-30A	:DTEST	: .26820E-03
-----------	--------	--------------

PAGE 2

JOSHUA
DPSTM-500

Title

An Example - Part IV

Vol.

II

Page

E.2.53

Rev.

Date

2/73

(Blank)

JOSHUA DPSTM-500	Title	An Example - Part IV	Vol.	Page
			II	E.2.54
			Rev.	Date
				2/73

RUN12791 - 3871 - STD ---- 3871 TUBE 035

DD --FUNCTION

ONEGRP.STANDARD.ISOTOPES.....

.....

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.55
		Rev.	Date 2/73

This figure requests a display of the ØNEGRP.STANDARD.ISØTØPES data record in the RUN12791 data set. The results of this request are shown on the following pages in the form indicated by the previously defined template.

PAGE 1

PAGE 2

RUN12791.ONEGRP.STANDARD.ISOTOPES

ISOTOPE	TEMP	REACTION
-----	-----	-----
:H-H2O	: 90	:TRANSPRT
:D-D2O	: 20	:TOTAL
:D-D2O	: 20	:CAPTURE
:D-D2O	: 20	:TRANSPRT
:D-D2O	: 20	:N2N
:D-D2O	: 60	:TOTAL
:D-D2O	: 60	:CAPTURE
:D-D2O	: 60	:TRANSPRT

PAGE 3

RUN12791.ONEGRP.STANDARD.ISOTOPES

ISOTOPE	TEMP	REACTION
-----	-----	-----
:D-D2O	: 60	:N2N
:D-D2O	: 90	:TOTAL
:D-D2O	: 90	:CAPTURE
:D-D2O	: 90	:TRANSPRT
:D-D2O	: 90	:N2N
:LI6	: 0	:TOTAL
:LI6	: 0	:CAPTURE
:LI6	: 0	:TRANSPRT

PAGE 4

RUN12791 - 3871 - STD ---- 3871 TUBE 035
DD --FUNCTION
ONEGRP.STANDARD.XSECTION.H-H2O.20.....
.....

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.59
		Rev.	Date 2/73

This figure shows a request for a display of all data records in the subtree defined by ØNEGRP.STANDARD.XSECTION.H-H2Ø.20 in the RUN12791 data set. There are three data record names in the subtree and the display for each follows in the form specified by the previously defined template.

RUN12791.ONEGRP.STANDARD.XSECTION.H-H2O.20.TOTAL

: 6 - NUMBER OF CASES

A CASE IS A PARTICULAR LATTICE-BUCKLING COMBINATION

PAGE 1

RUN12791.ONEGRP.STANDARD.XSECTION.H-H2O.20.TOTAL

AVERAGE ONE GROUP CROSS SECTION BY CASE NUMBER

: 1 : .26141E 02 : 2 : .27362E 02 : 3 : .27110E 02

: 4 : .45088E 01 : 5 : .43340E 01 : 6 : .26842E 02

PAGE 2

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.61
		Rev.	Date 2/73

RUN12791.ONEGRP.STANDARD.XSECTION.H-H2O.20.CAPTURE
: 6 - NUMBER OF CASES
A CASE IS A PARTICULAR LATTICE-BUCKLING COMBINATION

RUN12791.ONEGRP.STANDARD.XSECTION.H-H2O.20.CAPTURE
AVERAGE ONE GROUP CROSS SECTION BY CASE NUMBER
: 1 : .10257E 00 : 2 : .11561E 00 : 3 : .11330E 00
: 4 : .58919E-04 : 5 : .56102E-04 : 6 : .11086E 00

RUN12791.ONEGRP.STANDARD.XSECTION.H-H2O.20.TRANSPRT

: 6 - NUMBER OF CASES

A CASE IS A PARTICULAR LATTICE-BUCKLING COMBINATION

PAGE 1

RUN12791.ONEGRP.STANDARD.XSECTION.H-H2O.20.TRANSPRT

AVERAGE ONE GROUP CROSS SECTION BY CASE NUMBER

:	1	:	.15561E 02	:	2	:	.16939E 02	:	3	:	.16691E 02
:	4	:	.15030E 01	:	5	:	.14447E 01	:	6	:	.16429E 02

PAGE 2

JOSHUA DPSTM-500	Title An Example - Part IV	Vol.	Page
		II	E.2.63
		Rev.	Date
			2/73

(Blank)

Title	An Example - Part IV	Vol. II	Page E.2.64
		Rev.	Date 2/73

JOSHUA DPSTM-500	Title An Example - Part IV	Vol. II	Page E.2.65
		Rev.	Date 2/73

8. Sign-Off Procedure

Whenever a terminal session is completed the user is required to "SIGN-OFF". This is done with the function request EN as shown on the opposite page. The next frame to appear after this operation will be the "SIGN-ON" frame as discussed in Section 2.

DISTRIBUTION

1. F. P. Baranowski	50. P. B. Parks
2. J. Lewellyn	51. 777-M File Point
3. C. M. Brown	52. J. D. Spencer
4. R. J. Brown	53. M. B. Stroud
5. T. B. Niland	54. Helen P. Thompson
6. D. N. Bridges	55. W. H. Brotherton
7. L. A. Heinrich	56. R. E. Cooper
8. C. W. Farmer, Jr.	57. F. D. Knight
9. C. A. McCharen	58. F. D. King
10. J. W. Croach	59. W. V. Baxter
11. J. D. Ellett	60. R. J. Pryor
12. Prof. Kent F. Hansen	61. M. R. Buckner
13. R. F. Gambill, Jr.	62. C. K. Paulson
14. Mrs Marget Butler	63. B. J. Toppel
15. W. B. Daspit	64. Richard Ehrlich
16. C. E. Parker	65. D. Vondy
17. O. A. Towler	66. C. W. Craven, Jr.
18. D. A. Ward	67. R. L. Howell
19. C. J. Temple	68. P. Michael
20. R. L. Carson	69. Neal Carter
21. J. N. Wilson	70. G. H. Best
22. C. W. Thorngate	71. R. A. Dannels
23. J. E. Suich	72. D. Roy
24. J. W. Wade	73. P. G. Aline
25. H. C. Honeck	74. B. Hutchins
26. J. H. Hightower	75. Alleen Thompson
27. W. R. Hartshorn	76. H. Alter
28. J. C. Jensen	77. R. F. DeAngelis
29. R. S. Wood	78. Martin Becker
30. R. L. Fronthroth	79. F. H. Smith
31. D. R. Finch	80. G. F. O'Neill
32. W. H. Hightower	81. H. T. Toffer
33. J. H. Maddox	82. George Minton
34. J. H. Carswell	83. G. C. Ridgely
35. J. W. Stewart	84. M. M. Anderson
36. B. P. Compton	85. R. C. Haywood
37. W. E. Graves	86. R. J. Mullin
38. D. A. Sharp	87. Bruce M. Beardsley
39. F. J. McCrosson	88. J. M. Sicilian
40. P. L. Ames	89. R. L. Karel
41. D. E. Hostetler	90. D. S. Craig
42. J. E. Beach	91. D. S. Craig
43. R. M. Wallace	92. R. L. Hellens
44. M. L. Hyder	93. W. E. Vann
45. F. R. Field	94. R. H. Finley
46. D. H. Knowbel	95. A. N. Mallen
47. I. M. Macafee	97. B. G. Kingery
48. M. W. Lewis	98. David S. Selengut
49. J. R. Taylor	100. Vital Records File
	101-110. P. L. Ames

TIS File:

96, *99

* Master Copy