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. 9.1 OPERATING SYSTEM

Approved by

J. R. Hilley, 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.~~

DECEMBER 1973

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

E. I. du Pont de Nemours & Co.
Savannah River Laboratory
Aiken, S. C. 29801

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 Operating System

Part D Functions

Part E Modules and Subprograms

JOSHUA DPSTM-500	Title Part D. Functions	Vol. IX	Page D.O.1
		Rev. 1	Date 6/73

Contents

D.1 Functions

JOSHUA DPSTM-500	Title Part D. Functions	Vol. IX	Page D.0.2
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Functions

Vol.
IX

Page
D.1.1

Rev.
1

Date
6/73

Work Done by

R. L. Boyce, Jr.
W. H. Brotherton, Jr.
R. C. Haywood
J. C. Jensen
J. E. Suich

Report Written By

W. H. Brotherton, Jr.
R. C. Haywood

Contents

	<u>Page</u>
1. Introduction	D.1.3
2. Functions	D.1.5
2.1 AB	D.1.5
2.2 C,Co	D.1.7
2.3 CD	D.1.9
2.4 CT	D.1.11
2.5 DB	D.1.15
2.6 DC	D.1.17
2.7 DD	D.1.19
2.8 DE	D.1.21
2.9 DT	D.1.23
2.10 DU	D.1.25
2.11 ED	D.1.27
2.12 EJ	D.1.29
2.13 EM	D.1.33
2.14 EN	D.1.35
2.15 EP	D.1.37
2.16 EX	D.1.39
2.17 FM	D.1.43
2.18 IP	D.1.45
2.19 L,LI	D.1.47
2.20 LB	D.1.49
2.21 LO	D.1.51
2.22 LS	D.1.53
2.23 MA	D.1.55

JOSHUA DPSTM-500	Title Functions	Vol. IX	Page D.1.2
		Rev. 1	Date 6/7

Contents (Cont'd)

	<u>Page</u>
2.24 MD	D.1.57
2.25 ME	D.1.59
2.26 MT	D.1.61
2.27 PD	D.1.65
2.28 PR	D.1.67
2.29 RA	D.1.69
2.30 RD	D.1.71
2.31 RE	D.1.73
2.32 RM	D.1.75
2.33 RP	D.1.77
2.34 SC	D.1.79
2.35 SE	D.1.81
2.36 SI	D.1.83
2.37 SJ	D.1.85
2.38 SK	D.1.87
2.39 SN	D.1.89
2.40 SP	D.1.91
2.41 ST	D.1.93
2.42 SU	D.1.95
2.43 SW	D.1.97
2.44 TE	D.1.99
2.45 TS	D.1.101
2.46 VA	D.1.103
2.47 VQ	D.1.105
2.48 VV	D.1.107
2.49 XB	D.1.109
2.50 XR	D.1.111

1. Introduction

Upon a function request, a module, whose name takes the form JTAA where AA is the function requested, sets up one or more modules for sequential execution. The module accomplishes this by moving the list of sub-functions with their respective control flags to the function control table (FCT) area. The monitor then goes through the list to carry out the particular function.

The following pages give the various basic functions contained within the JOSHUA system along with the modules that compose them. (See function list in the General Overview of the JOSHUA Monitor section for functions' usage.)

JOSHUA DPSTM-500	Title Introduction	Vol. IX	Page D.1.4
		Rev. 1	Date 6/7

(Blank)

Purpose: To follow up abnormal termination and set up appropriate diagnostics.

Subfunctions (JOS Modules):

JTAB
JTABEND
JTDSPLY
JTABLOGC
JTCLOSDS
JTDEALDS

Flowchart:

JOSHUA DPSTM-500	Title	Function: AB	Vol.	Page
			IX	D.1.6
			Rev.	Date
			1	6/73

(Blank)

Purpose: To copy a particular tree or sub-tree record or portions of it.

Subfunctions (JOS Modules):

JTC
JTBR14
COPY
JTDSPLY
JTALOGIC

JOSHUA DPSTM-500	Title	Function: C or CO	Vol. IX	Page D.1.8
			Rev. 1	Date 6/7

(Blank)

Purpose: To create an unformatted JOSHUA data record from data formatted according to an associated template.

Subfunctions (JOS Modules): JTCD
 CREANAME
 TDISPLAY
 JTPAGER
 NTERPRET
 JTRSTPGE
 DDISPLAY
 JTPAGER
 JTRLOGIC
 NTERPRET

Flowchart:

JOSHUA DPSTM-500	Title Function: CD	Vol.	Page
		IX	D.1.10
		Rev.	Date
		1	6/7

(Blank)

JOSHUA

DPSTM-500

Title

Function: CT

Vol.
IX

Page
D.I.11

Rev.
1

Date
6/73

Purpose: To create a template to be used to format data in
JOSHUA data sets for display.

Subfunctions (JOS Modules):

JTCT
TEMPINIT
JTDSPLY
TEMPLIST
TEMPNEST
JTDSPLY
TEMPPAGE
TEMPEXIT
JTDSPLY

Flowchart:

Flowchart: (Cont.)

JOSHUA DPSTM-500	Title	Function: CT	Vol.	IX	Page
			Rev.	1	Date

(Blank)

JOSHUA DPSTM-500	Title Function: DB	Vol.	Page
		IX	D.1.15
		Rev.	Date
		1	6/73

Purpose: Given an incomplete record name, to delete the specified next-level modes.

Subfunctions (JOS Modules):

JTDB
GETBRTHR
JTSHIFTB
JTPAGER
DELBRTHR

Flowchart:

JOSHUA DPSTM-500	Title Function: DB	Vol. IX	Page D.1.16
		Rev. 1	Date 6/73

(Blank)

JOSHUA DPSTM-500	Title Function: DC	Vol. IX	Page D.1.17
		Rev. 1	Date 6/73

Purpose: To provide a display of the JOSHUA catalog.

Subfunctions (JOS Modules): JTDC
 DC
 JTPAGER

Flowchart:

JOSHUA DPSTM-500	Title Function: DC	Vol. IX	Page D.1.18
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Function: DD

Vol.

IX

Page

D.1.19

Rev.

1

Date

6/73

Purpose: To provide formatted display of unformatted JOSHUA data records.

Subfunctions (JOS Modules):

JTDD
JTRLOGIC
DDISPLAY
JTPAGER
JTRLOGIC
NTERPRET
JTRLOGIC

Flowchart:

JOSHUA DPSTM-500	Title	Function: DD	Vol.	Page
			IX	D.1.20
			Rev.	Date
			1	6/73

(Blank)

JOSHUA DPSTM-500	Title	Function: DE	Vol. IX	Page D.1.21
			Rev. 1	Date 6/73

Purpose: To delete requested name from catalog.

Subfunctions (JOS Modules):

JTDE
DELETE

Flowchart:

JOSHUA DPSTM-500	Title	Function: DE	Vol.	Page
			IX	D.1.22
			Rev. 1	Date 6/73

(Blank)

Purpose: To provide a display of the list, variable typings, and formatted page types of a specific template record.

Subfunctions (JOS Modules): JTDT
TDISPLAY
JTPAGER

Flowchart:

JOSHUA DPSTM-500	Title	Function: DT	Vol.	Page
			IX	D.1.24
			Rev.	Date
			1	6/73

(Blank)

JOSHUA

DPSTM-500

Title

Function: DU

Vol.

IX

Page

D.1.25

Rev.

1

Date

6/73

Purpose: To unload a JOSHUA data set to an external sequential file.

Subfunctions (JOS Modules): JTDU
VERIFY
DSDUMP

Flowchart:

JOSHUA DPSTM-500	Title	Function: DU	Vol.	Page
			IX	D.1.26
			Rev. 1	Date 6/7

(Blank)

JOSHUA DPSTM-500	Title Function: ED	Vol. IX	Page D.1.27
		Rev. 1	Date 6/73

Purpose: To perform text editing for a specified data set.

Subfunctions (JOS Modules):

JTED
 JTEDINIT
 EDIT
 EDIT?
 JTEDCNTL
 VERIFY
 JTUDQNQ
 JTEDLOGC
 JTINRDRX

Flowchart:

Purpose: To handle close out procedures at end of batch run.

Subfunctions (JOS Modules):

JTEJ
 JTJERRIT
 JTABEND
 JTIBEXIT
 JTBNAMCL
 JTWSTATS
 JTDQJOB
 JTDQSCR
 JTEJRLSE
 JTDCAT
 JTSCRTCH
 JTEJDEQ

Flowchart:

JOSHUA DPSTM-500	Title Function: EJ	Vol. IX	Page D.1.30
		Rev. 1	Date 6/73

JTBNAMCL

JTWSTATS

JTDQJOB

JTDQSCR

JTEJRLSE

JTDCAT

JOSHUA

DPSTM-500

Title

Function: EJ

Vol.
IX

Page
D.1.31

Rev.
1

Date
6/73

JTSCRTCH

Scratches scratch data set and job data set (if applicable).

JTEJDEQ

Deqs on scratch data set, job data set, BNAME path, and user's name.

JOSHUA DPSTM-500	Title Function: EJ	Vol. IX	Page D.1.32
		Rev. 1	Date 6/73

(Blank)

Purpose: To execute specified member name.

Subfunctions (JOS Modules):

JTEM
JTEMINIT
?MEMBER
JTBNAMCL
JTWSTATS
JTEMCLOS
JTDCA
JTDQJOB

Flowchart:

Purpose: To return to the sign-on frame.

Subfunctions (JOS Modules):

JTEN
JTPRNTRP
JTINRDRP
JTLOGOUT

Flowchart:

JOSHUA DPSTM-500	Title	Function: EN	Vol.	Page
			IX	D.1.36
			Rev. 1	Date 6/73

(Blank)

Purpose: To execute a series of functions as specified by a procedure record.

Subfunctions (JOS Modules):

JTEP
PROC
JTSELBUF
JTDSPLY
JTSHFTN
JTSELNIT
JTSELECT

Flowchart:

JOSHUA DPSTM-500	Title	Function: EP	Vol. IX	Page D.1.38
			Rev. 1	Date 6/73

Purpose: To execute a job.

Subfunctions (JOS Modules):

JTEX
JTEXBUF
JTDSPLY
JTEXINIT
JTEXTBUF
JTDSPLY
JTDINIT
JTPAGER
JTBLDQUE
JTSTATCK
JTINRDR
JTDSPLY
JTEXLOGC

Flowchart:

JOSHUA DPSTM-500	Title	Function: EX	Vol.	Page
			IX	D.1.42
			Rev.	Date
			1	6/73

(Blank)

Purpose: To interactively specify an SRP reactor facemap -
'INPUT'. 'GAUGE'. 'GEOMETRY'. XNAME.

Subfunctions (JOS Modules):

JTFM
FMSTART
FMFORM
JTDSPLY
FMBUILD
FMEND

Flowchart:

JOSHUA DPSTM-500	Title Function: FM	Vol. IX	Page D.1.44
		Rev.	Date

(Blank)

JOSHUA

DPSTM-500

Title

Function: IP

Vol.
IX

Page
D.1.45

Rev.
1

Date
6/73

Purpose: To interrupt a procedure.

Subfunctions (JOS Modules): JTIP
JTPRNTRP

Flowchart:

JOSHUA DPSTM-500	Title Function: IP	Vol. IX	Page D.1.46
		Rev. 1	Date 6/7

(Blank)

Purpose: To list particular tree records or portions of them.

Subfunctions (JOS Modules): JTLI
LIST
JTSHFTB
JTPAGER

Flowchart:

JOSHUA DPSTM-500	Title Function: L or LI	Vol. IX	Page D.1.48
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Function: LB

Vol.
IX

Page
D.1.49

Rev.
1

Date
6/73

Purpose: Given an incomplete name, to list all the next level modes.

Subfunctions (JOS Modules):

JTLB
LISTB
JTSHFTB
JTPAGER

Flowchart:

JOSHUA DPSTM-500	Title Function: LB	Vol. IX	Page D.1.50
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Function: LO

Vol.
IX

Page
D.1.51

Rev.
1

Date
6/73

Purpose: To load a JOSHUA data set from an external file containing a previously unloaded data set.

Subfunctions (JOS Modules):

JTLO
VERIFY
DSLOAD

Flowchart:

JOSHUA DPSTM-500	Title	Function: LO	Vol.	Page
			IX	D.1.52
			Rev.	Date
			1	6/73

(Blank)

Purpose: To fit a mathematical function to a set of data points.

Subfunctions (JOS Modules):

JTLS
LSTSQR
DDISPLAY
JTPAGER

Flowchart:

JOSHUA DPSTM-500	Title Function: LS	Vol. IX	Page D.1.54
		Rev. 1	Date 6/73

(Blank)

JOSHUA DPSTM-500	Title Function: MA	Vol. IX	Page D.1.55
		Rev. 1	Date 6/73

Purpose: To compress a JOSHUA data set in place and to copy the compressed data set to an external sequential file.

Subfunctions (JOS Modules):

JTMA
VERIFY
DSMAINT

Flowchart:

JOSHUA DPSTM-500	Title Function: MA	Vol. IX	Page D.1.56
		Rev. 1	Date 6/73

(Blank)

Purpose: To modify a JOSHUA data record.

Subfunctions (JOS Modules):

JTMD
 JTBR14
 DDISPLAY
 JTPAGER
 JTRLOGIC
 NTERPRET
 JTRLOGIC

Flowchart:

JOSHUA DPSTM-500	Title	Function: MD	Vol. IX	Page D.1.58
			Rev. 1	Date 6/73

JOSHUA

DPSTM-500

Title

Function: ME

Vol.
IX

Page
D.1.59

Rev.
1

Date
6/73

Purpose: To merge into a JOSHUA data set data from an external data set which resulted from a previous maintenance pass (see function MA).

Subfunctions (JOS Modules):

JTME
VERIFY
DSMERGE

Flowchart:

JOSHUA DPSTM-500	Title Function: ME	Vol. IX	Page D.I.60
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Function: MT

Vol.

IX

Page

D.1.61

Rev.

1

Date

6/73

Purpose:

To display an existing template record, accept modifications, and write out the modified template either as a replacement of the template being modified or as a differently named template at the user's option.

Subfunctions (JOS Modules):

JTMT
TDISPLAY
JTDSPLY
TEMPLIST
MTPAGE
JTDSPLY
MDIRECT
TEMPPAGE
TEMPEXIT
TEMPERRM
JTDSPLY

Flowchart:

Flowchart: (Cont.)

JOSHUA DPSTM-500	Title Function: MT	Vol. IX	Page D.1.64
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Function: PD

Vol.
IX

Page
D.1.65

Rev.

1

Date
6/73

Purpose: To provide a formatted punched deck of unformatted JOSHUA data records.

Subfunctions (JOS Modules):

JTPD
JTRLOGIC
DDISPLAY
JTPAGER
JTRLOGIC
NTERPRET

Flowchart:

JOSHUA DPSTM-500	Title Function: PD	Vol. IX	Page D.1.66
		Rev. 1	Date 6/73

(Blank)

Purpose: To provide a printed or punched copy of the display buffer.

Subfunctions (JOS Modules):

JTPR
JTSTATCK
JTINRDRP
JTDSPLY

Flowchart:

JOSHUA DPSTM-500	Title	Function: PR	Vol.	Page
			IX	D.1.68
			Rev.	Date
			1	6/73

(Blank)

JOSHUA

DPSTM-500

Title

Function: RA

Vol.
IX

Page
D.1.69

Rev.
1

Date
6/73

Purpose: To provide system abending subfunction.

Subfunctions (JOS Modules):

JTRA
JTOC1

Flowchart:

JOSHUA DPSTM-500	Title Function: RA	Vol. IX	Page D.1.70
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Function: RD

Vol.

IX

Page

D.1.71

Rev.

1

Date

6/73

Purpose: To provide a monitor dump upon abnormal termination of a subfunction.

Subfunctions (JOS Modules):

JTRD

JTREQDMP

Flowchart:

JOSHUA
DPSTM-500

Title

Function: RD

Vol.

IX

Page

D.1.72

Rev.

1

Date

6/73

(Blank)

JOSHUA

DPSTM-500

Title

Function: RE

Vol.
IX

Page
D.1.73

Rev.
1

Date
6/73

Purpose: To rename a JOSHUA data record.

Subfunctions (JOS Modules): JTRE
RENAME

Flowchart:

JOSHUA DPSTM-500	Title	Function: RE	Vol.	Page
			IX	D.1.74
			Rev. 1	Date 6/73

(Blank)

Purpose: To interactively specify an SRP reactor facemap
'REACTOR'. RVSN. 'GEOMETRY'. 'HEX'. 'MESH'. ZLVL.

Subfunctions (JOS Modules):

JTRM
RMSTART
FMFORM
JTDSPLY
FMBUILD
RMEND

Flowchart:

JOSHUA DPSTM-500	Title Function: RM	Vol. IX	Page D.1.76
		Rev. 1	Date 6/7

(Blank)

JOSHUA

DPSTM-500

Title

Function: RP

Vol.
IX

Page
D.1.77

Rev.
1

Date
6/73

Purpose: To restart a procedure.

Subfunctions (JOS Modules):

JTRP

JTPRNTRP

Flowchart:

JOSHUA DPSTM-500	Title	Function: RP	Vol.	Page
			IX	D.1.78
			Rev.	Date
			1	6/73

(Blank)

JOSHUA DPSTM-500	Title	Function: SC	Vol. IX	Page D.1.79
			Rev. 1	Date 6/73

Purpose: To release, completely or in part, the space allocated to a JOSHUA data set.

Subfunctions (JOS Modules):

JTSC
VERIFY
SCRATCH

Flowchart:

JOSHUA DPSTM-500	Title Function: SC	Vol.	Page
		IX	D.1.80
		Rev.	Date
		1	6/73

(Blank)

Purpose: To allow selection of a particular function.

Subfunctions (JOS Modules):

JTSE
JTFTIME
JTSELBUF
JTDSPLY
JTSHFTN
JTSELNIT
JTSELECT

Flowchart:

JOSHUA DPSTM-500	Title	Function: SE	Vol.	Page
			IX Rev. 1	D.1.82 Date 6/7

JOSHUA

DPSTM-500

Title

Function: SI

Vol.

IX

Page

D.1.83

Rev.

1

Date

6/73

Purpose: To set up and test validity of sign-on parameters.

Subfunctions (JOS Modules):

JTSI
JTLOGBUF
JTUDQNN
JTDSPLY
JTPASWRD
JTLOGIN
VERIFY
JTUDQNN
SYSMSG
JTDSPLY

Flowchart:

JOSHUA DPSTM-500	Title Function: SI	Vol. IX	Page D.1.84
		Rev. 1	Date 6/73

JOSHUA

DPSTM-500

Title Function: SJ

Vol.
IX

Page
D.1.85

Rev.
1

Date
6/73

Purpose: To set job name for user's use.

Subfunctions (JOS Modules):

JTSJ
SETJOB
JTDSPLY

Flowchart:

JOSHUA
DPSTM-500

Title

Function: SJ

Vol.
IX

Page
D.1.86

Rev.
1

Date
6/73

(Blank)

JOSHUA DPSTM-500	Title Function: SK	Vol. IX	Page D.1.87
		Rev. 1	Date 6/73

Purpose: To skip a procedure step.

Subfunctions (JOS Modules): . . .

JTSK
JTBR14

JOSHUA
DPSTM-500

Title

Function: SK

Vol.

IX

Page

D.1.88

Rev.

1

Date

6/73

(Blank)

JOSHUA

DPSTM-500

Title Function: SN

Vol. IX

Page
D.1.89

Rev. 1

Date
6/73

Purpose: To initialize appropriate areas for batch job execution.

Subfunctions (JOS Modules):

JTSN
JTSNENQ
JTSNOPEN
JTAUTOEM

Flowchart:

JOSHUA DPSTM-500	Title Function: SN	Vol. IX	Page D.1.90
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Function: SP

Vol.
IX

Page
D.1.91

Rev.
1

Date
6/73

Purpose: To stop a procedure.

Subfunctions (JOS Modules):

JTSP
JTPRNTRP

Flowchart:

JOSHUA
DPSTM-500

Title

Function: SP

Vol.

IX

Page

D.1.92

Rev.

1

Date

6/73

(Blank)

JOSHUA

DPSTM-500

Title Function: ST

Vol. IX	Page D.1.93
Rev. 1	Date 6/73

Purpose: To display the status of a user's job(s).

Subfunctions (JOS Modules):

JTST
JTSTLOGC
JTSTATUS
JTDSPLY
ZONK1
JTPAGER

Flowchart:

JOSHUA

DPSTM-500

Title

Function: SU

Vol.
IX

Page
D.1.95

Rev.
1

Date
6/73

Purpose: To establish desired user data set in JOSHUA search sequence.

Subfunctions (JOS Modules):

JTSU
JTSETUSR

Flowchart:

JOSHUA DPSTM-500	Title	Function: SU	Vol.	Page
			IX	D.1.96
			Rev.	Date
			1	6/73

(Blank)

JOSHUA

DPSTM-500

Title Function: SW

Vol.
IX

Page
D.1.97

Rev.
1

Date
6/73

Purpose: To establish the output JOSHUA data set.

Subfunctions (JOS Modules):

JTSW
VERIFY
JTUDQNQ

Flowchart:

JOSHUA DPSTM-500	Title	Function: SW	Vol. IX	Page D.1.98
			Rev. 1	Date 6/7

(Blank)

JOSHUA

DPSTM-500

Title Function: TE

Vol.
IX

Page
D.1.99

Rev.
1

Date
6/73

Purpose: To test a module or a function.

Subfunctions (JOS Modules):

JTTE
(User Supplied Modules)

Flowchart:

JOSHUA DPSTM-500	Title Function: TE	Vol. IX	Page D.1.100
		Rev. 1	Date 6

(Blank)

JOSHUA DPSTM-500	Title Function: TS	Vol. IX	Page D.1.101
		Rev. 1	Date 6/73

Purpose: To test a module or a function.

Subfunctions (JOS Modules):

JTTS

(User Supplied Modules)

Flowchart:

JOSHUA DPSTM-500	Title Function: TS	Vol. IX	Page D.1.102
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Function: VA

Vol.
IX

Page
D.1.103

Rev.

1

Date
6/73

Purpose: To activate or deactivate a terminal unit; to deallocate or allocate a terminal unit; to provide for intra-terminal region communication.

Subfunctions (JOS Modules):

JTVA
JTSETLNK
JTVARY

Flowchart:

JOSHUA DPSTM-500	Title	Function: VA	Vol.	Page
			IX	D.1.104
			Rev.	Date
			1	6/73

(Blank)

Purpose: To display the JCL for all jobs in the hold queue.

Subfunctions (JOS Modules):

JTVQ
JTVIEWQ
JTPAGER

Flowchart:

JOSHUA DPSTM-500	Title	Function: VQ	Vol.	Page
			IX	D.1.106
			Rev.	Date
			1	6/73

(Blank)

JOSHUA DPSTM-500	Title Function: VV	Vol. IX	Page D.1.107
		Rev. 1	Date 6/73

Purpose: To view protection violations.

Subfunctions (JOS Modules):

JTVV
 JTVQACCT
 JTPAGER

Flowchart:

JOSHUA DPSTM-500	Title Function: VV	Vol. IX	Page D.1.108
		Rev. 1	Date 6/73

(Blank)

JOSHUA DPSTM-500	Title Function: XB	Vol. IX	Page D.1.109
		Rev. 1	Date 6/73

Purpose: To present jobs in hold queue data set to HASP.

Subfunctions (JOS Modules):

JTXB
 JTXBCHEK
 JTINRDRB

Flowchart:

JOSHUA DPSTM-500	Title Function: XB	Vol. IX	Page D.1.110
		Rev. 1	Date 6/73

(Blank)

JOSHUA DPSTM-500	Title Function: XR	Vol.	Page
		IX	D.1.111
		Rev.	Date
		1	6/73

Purpose: To link to a given module for its execution within the JOSHUA Terminal Region.

Subfunctions (JOS Modules): JTXR
JTEXREGN

Flowchart:

JOSHUA

DPSTM-500

Title

Function: XR

Vol.

IX

Page

D.1.112

Rev.

1

Date

6/73

(Blank)

JOSHUA DPSTM-500	Title Part E. Modules and Subprograms	Vol. IX	Page E.O.1
		Rev. 1	Date 6/73

Contents

E.1 Modules and Subprograms

JOSHUA DPSTM-500	Title Part E. Modules and Subprograms	Vol. IX	Page E.0.2
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Modules

Vol. IX

Page E.1.1

Rev. 1

Date 6/73

Work Done By

R. L. Boyce, Jr.
W. H. Brotherton, Jr.
R. C. Haywood
J. C. Jensen
F. D. Knight
J. E. Suich
R. S. Wood

Report Written By

R. L. Boyce, Jr.
W. H. Brotherton, Jr.
R. C. Haywood
R. S. Wood

Contents

	<u>Page</u>
1. Introduction	E.1.3
2. General Name Handling Conventions	E.1.3
2.1 Procedure for Read-type Operations	E.1.3
2.2 Procedure for Write-type Operations	E.1.4
3. Template Interpretation and Structure	E.1.4
4. Modules	E.1.9
4.1 \$YUDLI	E.1.9
4.2 \$YUDPR	E.1.11
4.3 COPY	E.1.13
4.4 CREANAME	E.1.17
4.5 DALLOC	E.1.19
4.6 DC	E.1.27
4.7 DDISPLAY	E.1.31
4.7.1 BTFP	E.1.37
4.7.2 CONVER	E.1.41
4.7.3 LOADPG	E.1.45
4.8 DELBRTHR	E.1.49
4.9 DELETE	E.1.51
4.10 DSDUMP	E.1.53
4.11 DSLOAD	E.1.55
4.12 DSMAINT	E.1.59
4.13 DSMERGE	E.1.63
4.14 DUMP61	E.1.67
4.15 EDIT	E.1.71
4.16 EDIT\$	E.1.73
4.17 EDITA	E.1.75
4.18 EDITC	E.1.77

JOSHUA DPSTM-500	Title Modules	Vol. IX	Page E.1.2
		Rev. 1	Date 6/73

Contents (Cont.)

	<u>Page</u>
4.19 EDITD	E.1.79
4.20 EDITG	E.1.81
4.21 EDITL	E.1.83
4.22 EDITO	E.1.85
4.23 EDITQ	E.1.87
4.24 EDITR	E.1.89
4.25 EDITS	E.1.91
4.26 EDITW	E.1.93
4.27 EDITX	E.1.95
4.28 EDITY	E.1.97
4.29 EDITZ	E.1.99
4.30 EDITO	E.1.101
4.31 EDIT1	E.1.103
4.32 EDIT8	E.1.105
4.33 EDUTIL	E.1.107
4.34 FMBUILD	E.1.109
4.34.1 FMSCAN	E.1.112
4.34.2 FMWCH	E.1.114
4.34.3 MNUREV	E.1.116
4.35 FMEND	E.1.119
4.36 FMFORM	E.1.123
4.36.1 MNUFIL	E.1.125
4.36.2 SETMAP	E.1.127
4.37 FMSTART	E.1.131
4.37.1 FMREAD	E.1.134
4.38 FRECHECK	E.1.137
4.38.1 TRESRT	E.1.140
4.38.2 SIFTUP, SIFTU2	E.1.142
4.39 GALLOC	E.1.145
4.40 GETBRTHR	E.1.151

1. Introduction

This section will describe the modules which compose the JOSHUA Operating System. The module description is normally followed by a flow chart. In general, if a subroutine is used exclusively in a particular module, its description follows immediately that module's description. Subprograms used in more than one module are described in the Subprogram Section.

The following statements concerning FORTRAN-coded modules pertains to all such modules unless specifically stated otherwise in the module description.

- The main program is defined to the compiler as a subroutine. As a result, normal termination is generally through a RETURN statement.
- Included under the description of an argument is the name of the system-defined area to which the argument is generally although not necessarily considered to refer.
- Lists of subprograms will not include references developed by the FORTRAN compiler nor the JOSHUA precompiler.
- For the modules, all subprogram references (including entry points and control sections) contained within the module are listed; for a subprogram reference in a subprogram, only the entry points referenced by the subprogram are listed in the subprogram description.
- Subprogram-names followed by an asterisk (occurs in module write-ups only) denote subprograms pertaining only to the particular module in which it is listed. The subprogram description is included with the module description rather than placed in the Subprogram Section.
- Data set references listed are those referenced in the source coding of a module and do not include data sets accessed by the data manager (BNAM) or by any other module which may be executed.

2. General Name Handling Conventions

The input and output names are submitted in two eighteen-doubleword arrays - usually assumed to be WCBARG1 and WCBARG2, respectively. The number of qualifiers contained in WCBARG1 is either specified by argument - usually WXTNTGR - or determined by locating a doubleword of hexadecimal 55's following the last valid qualifier. For WCBARG2, it is determined by locating the hexadecimal 55's.

2.1 Procedure for Read-type Operations

- Submit the contents of WCBARG1 to BNAM as entered.
- If an error occurs, the first qualifier in WCBARG1 is assumed to be an overriding data set designation. Adjustments are made to the name-string and the search sequence list and BNAM is entered again.

JOSHUA DPSTM-500	Title Modules	Vol. IX	Page E.1.4
		Rev. 1	Date 6/

- If both attempts fail, the record is considered nonexistent.

The first qualifier in the namestring following a successful search defines the data set in which the record or subtree is located.

2.2 Procedure for Write-type Operations

- If WCBARG2 does not define the output, the contents of WCBARG1 are used.
- A name in WCBARG2 is submitted to BNAM as found; if WCBARG1 is used, the initial qualifier is stripped if it is of the form of a job or user data set name.

3. Template Interpretation and Structure

The structure and interpretation of a template is an integral factor in those modules concerned with the creation, modification and display of both JOSHUA template and data records.

A template describes a logical record type; it also contains coded instructions for presenting a formatted display of the data in a record of the type.

The template record is developed by parsing a Fortran-like list and enclosing page display formats.

The list, which is retained in its original form as part of the template but is not used during actual display interpretation, is unlike a true Fortran list in that it is used here simply to define the structure of a logical record and not the transfer of data to variables. During creation of a template the list is parsed. All individual data fields or variables are located. Discrete groups of variables are segregated into loops based upon the relative positions of the variables and the implicit or explicit indexing of their occurrences. When loops are found one within another, the major loop in the series marks the bounds of a nest. Loop limits may be fixed (implicit limits are included in this group) or variable. A variable limit will be either a previously defined data variable or an index to a major loop within the nest.

The storage requirement for any variable defaults to four bytes; this may be overridden by a T*S declaration where T designates type (I, R, or C for integer, real, or character, respectively) and S the storage space or length. Type is used only in controlling length and storage specifications.

Each nest has an associated page type. During creation of the template, each page type display is examined and converted to an encoded format. The scan is made line by line and encodes repetitive occurrences of a character, repeat requests, roll-off indicators and data field length and conversion specifications.

FIGURE 1. TEMPLATE STRUCTURE

<u>Offset</u>	<u>Bytes</u>	<u>Contents</u>
0	2	Number of nests in template.
2	4	Number of halfwords in list.
4	var.	LIST.

After the above fields, the following subblock is defined for each nest in the template with each subblock occurring in the sequence in which the nest is defined in the list.

+0	2	Number of loops in the nest.
+2	2	Number of page types defined (always one).
+4	2	Length of the coded format in halfword (two-byte) units.
+6	2	Number of variables in the nest.
+8	2	The byte, relative one, in this subblock, in which the coded format for the nest begins (always eleven).
+10	var.	Coded page format.

Within the subblock for a nest the following set of controls is established for each loop in the nest and occurs in the sequence in which the loop was defined.

+0	2	Number of variables in the loop.
+2	2	Lower limit of the loop.
+4	2	Upper limit of the loop.
+6	2	The byte, relative one, in the subblock for the nest, in which the coded format for this loop begins. (Always eleven for the first nest in the loop.)
+8	2	The byte location (relative one) on the page at which the formatting for this loop initially begins.
+10	2	The page type (always one).

Following the definition of the controls for all the loops within the nest but still within the subblock for the nest, the following field occurs for each variable within the nest.

+0	2	Type and length code of a variable.
----	---	-------------------------------------

JOSHUA DPSTM-500	Title Modules	Vol. IX	Page E.1.6
		Rev. 1	Date 6/73

FIGURE 2. LOOP LIMIT CODES

<u>Type</u>	<u>Algorithm</u>
Fixed	Actual Value
Variable - data	- (variable number +100)
- index	The number of the highest ranking loop in the nest, minus the number of the loop in which the limit appears as an index, minus one.

For the list

S,I, ((X,J = 1,K), K = 1,2), L = 1,I)

the upper limits would be coded

J -2
 K 2
 L -102

JOSHUA DPSTM-500	Title Modules	Vol. IX	Page E.1.7
		Rev. 1	Date 6/73

TABLE I

Current implementation imposes the following restrictions on templates:

Number of nests	5
Total number of loops	12
Number of data variables	20
Total length of a coded nest	680 bytes

JOSHUA DPSTM-500	Title Modules	Vol. IX	Page E.1.8
		Rev. 1	Date 6/73

FIGURE 3. TEMPLATE FORMAT CODES

All bytes in a template format have the following interpretation except when completing a definition begun by an immediately preceding byte which was itself one of the following.

<u>Byte Value (Decimal)</u>	<u>Definition</u>
0	Repeat indicator. The next byte indicates the number of format bytes backward at which decoding should be resumed in order to continue formatting the display.
3 - 63	The number of consecutive bytes which the data in the next byte is to be propagated. Single and double occurrences of a character are not preceded by a count field.
64 - 249	A character to be propagated.
250 - 255	Data field format descriptions. ¹
	250 Loop index field
	251 Hexadecimal representation
	252 Character field - no conversion
	253 Fixed point conversion (I field)
	254 Floating point conversion (E field)
	255 Floating point conversion (F field)

-
1. All data fields are preceded by a colon. Thus the actual number of positions taken for representation is the length value +1. The byte following a floating-point indicator defines the number of decimal positions as well as the length. The algorithm is decimal positions x 16 + length for F type formats and (decimal positions -4) * 16 + length for E type formats.

JOSHUA

DPSTM-500

Title

Module: \$YUDLI

Vol.

IX

Page

E.1.9

Rev.

1

Date

6/73

Purpose:

In the batch, to serve as an interface between a module requesting a printed list of the fully-qualified names in a subtree and the system module LIST.

Attribute:

Nonreusable

Entry:

\$YUDLI or \$YUDLY

Arguments:

General purpose register one should point to the following list of addressess:

1. Either the true CCB or the user's local CCB (\$JCB).
2. A fullword containing the number of qualifiers in the name to be passed to LIST;
3. For \$YUDLI a sequence of addresses of doublewords; for \$YUDLY the address of a set of contiguous doublewords. In either case the doublewords contain the qualifiers in the name being passed to LIST. In general, the qualifiers will not represent a fully-qualified name.

Program Logic:

- The parameter list to LIST is constructed.
- Contiguous doublewords of WCBARG1 are loaded with the designated qualifiers and the doubleword following the last to receive a qualifier is filled with hexadecimal 55's.
- The number of qualifiers is placed in WXTNTGR.
- An IBCOM initialization is effected by:
 1. Loading JIBCOM;
 2. Branching to JIBCOM to obtain the address of IBCOM;
 3. Branching to IBCOM + 64, the entry point for initialization;
 4. Deleting JIBCOM.
- A link is made to LIST.
- Registers are reset and control returned to the point of invocation.

Language:

Assembler

Author:

R. C. Haywood

JOSHUA DPSTM-500	Title Module: \$YUDLI	Vol. IX	Page E.1.10
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Module: \$YUDPR

Vol.

Page

IX

1.1.11

Rev.

1

Date

6/73

Purpose:

In the batch, to serve as an interface between a user module and the system module DDISPLAY.

Attribute:

Nonreusable

Entry:

\$YUDPR

Arguments:

General purpose register points to the following list of addresses:

1. Either the true CCB or the user's local CCB (\$JCB).
2. A fullword containing the number of qualifiers in the name to be passed to DDISPLAY.
3. For the number of qualifiers so specified, a sequence of addresses of doublewords containing the qualifiers.

Program Logic:

- o The parameter list to DDISPLAY is constructed.
- o Contiguous doublewords of WCBARG1 are loaded with the designated qualifiers. The doubleword following the last containing a qualifier is filled with hexadecimal 55's to mark the end of the qualifiers.
- o The first doubleword of WCBARG2 is also filled with hexadecimal 55's.
- o The number of qualifiers is placed in WXTNTGR.
- o The third word of WXTWORK1 is set to zero to indicate the initial call to DDISPLAY.
- o \$CBNERR is set to zero.
- o An IBCOM initialization is effected by:
 1. Loading JIBCOM;
 2. Branching to JIBCOM to obtain the address of IBCOM;
 3. Branching to IBCOM + 64, the entry point for initialization;
 4. Deleting JIBCOM.
- o A loop is established for successive links to DDISPLAY. The loop terminates when the word at WXTWORK + 8 returns a zero.
- o Registers are reset and control returned to the point of invocation.

Language:

Assembler

Author:

R. C. Haywood

JOSHUA

DPSTM-500

Title

Module: COPY

Vol.

IX

Page

E.1.13

Rev.

1

Date

6/73

Purpose:

To copy a named data record to another named data record.

Attribute:

Nonreusable

JOS Main Program:

Main

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
A1	Real*8	CCB
A2	Real*8	WCBARG1 - specifies the record or subtree of records to be copied; must contain at least one qualifier.
A8	Integer*4	WXTNTGR - specifies the number of qualifiers in A2 at entry.
A3	Real*8	WCBARG2 - specifies the output record or subtree; when null, the contents of A2 are placed here; those qualifiers added to A2 to complete a record name are also added to A3.
A10	Integer*4	WXTWORK1 - used as the COMTEM array to BNAM which must be maintained between executions; A10(3) must be zero on the initial entry.
A11	Integer*2	WTWORK2 - used as the STORE array to BNAM which must be maintained between executions.
A4	Real*8	Area pointed to by MCBBUFFG - as the terminal buffer it is used to submit messages to the user; also used as a buffer area during record copying.

Subprograms:

Control Section

Entry Point

\$TERR
LINK
MOVEIT
SHIFTB
TRNSLT

\$TERR
BNAM, JTREE, WPAGE
MOVEIT
SHIFTB
TRNSLT

Data Sets:

JOSHUA Tree and Data Files
Communication Data Set (CDS)

JOSHUA DPSTM-500	Title	Module: COPY	Vol.	Page
			IX	E.1.14
			Rev.	Date
			1	6/73

- Initialize A10 for BNAM on initial execution.
- Call BNAM (functional request 6) for the next record in the subtree. If deleted, call again for the next record.
- If A3 contains no output specification, move A2 to A3 to the extent of A2's initial definition and including any prefixion by BNAM. If the first qualifier of A2 indicates a job or user data set, shift the contents of A3 so as to delete the first node.
- Add to A3 any low level qualifiers added to A2 by BNAM.
- Call BNAM (functional request 2) to catalog the output record and obtain the starting address for the output record.
- Remove any low level qualifiers added to A3.
- Copy the data.
- In the batch or when user is 9999, copy each record in the subtree in a single execution of COPY; otherwise, copy one record per execution.
- Place a message in A4. If in the batch and functional call WPAGE.

Terminating Conditions:

1. No qualifiers in A2.
2. Output record, when fully defined, contained more than sixteen (16) qualifiers.
3. Input or output data set could not be opened.
4. BNAM errors.

Language:

FORTRAN IV

Author:

R. C. Haywood - J. C. Jensen

JOSHUA DPSTM-500	Title Module: COPY	Vol. IX	Page E.1.16
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Module: CREANAME

Vol.

IX

Page

E.1.17

Rev.

1

Date

6/73

Purpose:

In the CD function, to determine the appropriateness of a data record or template record display and to determine the function list accordingly.

Attribute:

Nonreusable

JOS Main Program:

MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
A1	Real*8	CCB
A2	Real*8	WCBARG1 - contains the name of the input or output record.
A8	Integer*4	WXNTGR - specifies the number of qualifiers in A2.
A15	Integer*4	MCBCFCT

Subprograms:

Control Section

Entry Point

LINK
XLOC

BNAM
XLOC

Objective:

When the record named in A2 exists, to alter the function list so that the existing data will be displayed. None of the argument values is to be altered.

Program Logic:

- An attempt is made to read the record defined in A2. When successful, the positions in the function list defining the next subfunction (TDISPLAY) and its control information are replaced by DDISPLAY and its control information.
- Execution terminates with a RETURN.

Language:

FORTTRAN IV

Author:

J. E. Suich

JOSHUA

DPSTM-500

Title

Module: DALLOC

Vol.

IX

Page

E.1.19

Rev.

1

Date

6/73

Purpose:

To give JOSHUA terminal users the ability to selectively allocate and deallocate direct access devices (2314/3330) and data sets.

Attribute:

Nonreusable

JOS Main Program Name: DALLOC

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
REQUEST	F	Request element where first character is 'A' for allocate, 'U' for deallocate.
DCB POINTER	F	DCB adcon (4 bytes)
VOLID	F	Volume I.D. (6 EBCDIC characters)
DSN	F	Data set name (44 characters)
SPACE	F	3 binary bytes, P, S, D, Where P is the primary space requested in cylinders, S is secondary space in cylinders, D is directory space.
BADGE	F	Badge number (4 EBCDIC characters)

Internal Tables:

List of authorized volumes for accessing existing data sets; another table lists authorized volumes for creating new data sets.

External References:

OPEN, CLOSE, STAE, SVC 32, SVC 251

Data Sets:

There is one data set per call. The name is passed via the argument list. Old data sets must be catalogued, and reside on one of the volumes in the table described above.

Objective:

The objective is to provide access to and creation of user data sets using the JOSHUA terminal system without inserting a specific DD card for each data set in the monitor job deck (JTMON).

JOSHUA DPSTM-500	Title Module: DALLOC	Vol. IX	Page E.1.20
		Rev. 1	Date 6/7

Program Logic:

Dummy DD cards are inserted into the JOSHUA deck and are used by the 360 operating system to build DD entries in the task input output table (TIOT) for each DD statement. The operating system does not allocate a device to dummy DD statements, and all IO references to that DD statement are ignored. DALLOC uses these dummy DD entries in the TIOT to store the address of the associated unit control block, and other information used by the space allocation routine and OPEN modules. The correct UCB address is found by searching the UCB chain and matching the volume ID field with the volume requested. A model job file control block (JFCB) is built and a bit is set to indicate the JFCB is to be written to the JOBQ during OPEN type J processing. If a request for disk space is made, the direct access data set management (DADSM) bit in the TIOT is set, and the amount of space requested is stored in the JFCB before control is passed to the space allocation routine (SVC 32). The allocation bit in the UCB is set providing the online indicator bit is on; otherwise, device allocation does not take place.

A STAE (Specify Task Abnormal Exit) macro is issued at the beginning of DALLOC and control is passed to an exit routine in case of an ABEND. The exit routine does some cleanup such as making sure the data set was not left open.

The deallocation request via DALLOC requires virtually the same input as described above, except the first character of the request element is a 'U'. The routine will turn off the allocation bit in the appropriate UCB providing the allocation bit is on and no DCB's are open to the IO device.

DALLOC returns the unit address in the last three bytes of the first parameter for allocation requests. Register 15 is used for the completion code return; a value of zero indicates successful completion. Return codes and their meanings are shown in the attached table.

Referenced by:

JTDADR

Language:

Assembler

Author:

F. C. Fortune

TABLE II

RETURN CODES FOR DALLOC

Contents of
Register 15

Description

00	Successful completion.
01	Volume requested not found.
02	UCB found is not 2314/3330.
03	Device is not online.
04	No DD entry in TIOT.
05	Invalid DCB, DSORG not PS, PO or DA.
06	DCB is already OPEN.
07	Disk space requested on unauthorized volume.
08	Volume requested was not a permanent resident volume.
09	Tried to allocate a Systems Data Set. (SYS)
10	Data set failed to OPEN.
11	Space allocation failed.
12	SCB for STAE not created.
13	Badge # in job card does not match JTMON's.
14	Device was not allocated. No action was taken (deallocation request).
15	2314/3330 was still being used by another job. No action was taken (deallocation request).

JOSHUA DPSTM-500	Title Module: DALLOC	Vol. IX	Page E.1.26
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Module: DC

Vol.

IX

Page

E.1.27

Rev.

1

Date

6/73

Purpose:

Provide a formatted display of various data contained in the Joshua Data Set Name Catalog (JDSNC), the Joshua Data Set Control Blocks (JDSCB), the Joshua Catalog Extension Blocks (JCEB), and the Free Space Elements (FSE) records.

Attribute:

Nonreusable

JOS Main Program:

MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
MCB	Real*8	CCB
NAME	Real*8	WCBARG1 - specifies the desired option.
TBUF	Logical*1	Area pointed to by MCBBUFFG - the formatted output is written from this area.

Subprograms:

<u>Control Section</u>	<u>Entry Point</u>
LCDINT	\$CDINT, HCDINT
LINK	JTREE, WPAGE
MOVEIT	MOVEIT
TRNSLT	TRNSLT

Data Sets:

Joshua Data Set Name Catalog (JDSNC).
Joshua Data Set Control Blocks (JDSCB).
Joshua Catalog Extension Blocks (JCEB).
Joshua tree and data files (for FSE records).
Communication Data Set (CDS - output display).

Program Logic:

- Determine specific request by examination of keywords in NAME.
- Format requested data a page at a time on the CDS. If the request cannot be satisfied, write a message to that effect on the CDS.
- The following options result in a list of data sets meeting the appropriate specifications:

JOSHUA DPSTM-500	Title	Module: DC	Vol. IX	Page E.1.28
			Rev. 1	Date 6/73

ALL	all data sets.
STD	all standard data sets.
JOB.ALL	all job data sets.
JOB.nnnn	all job data sets for user nnnn.
JOB	all job data sets identifiable with the requestor.
USER.ALL	all user data sets.

For this list the JDSNC is read and the appropriate entries are extracted and placed in the list.

- The following options result in formatting the control data for the specified data sets:

XXXXXXXX	a specific data set.
USER.nnnn	all extents for user nnnn.
-	all extents of the data set for which the most recent SW is in effect.

In order to present the catalog control data, the JDSNC is read to locate the extents for the specified data set and obtain the record number in the JDSCB and JCEB containing the control data. These data sets are then read and their data formatted. The data for each extent is placed on a separate CDS record.

- The option SPACE is used to request a list of FSE's in the tree and data files. Each file contains its own list of FSE's in the first relative record of the file. The records are read and the data set reference number, the starting record and the number of contiguous records in each block are formatted.

Language:

FORTRAN IV

Author:

J. C. Jensen

JOSHUA DPSTM-500	Title	Module: DC	Vol. IX	Page E.1.30
			Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Module: DDISPLAY

Vol.

IX

Page

E.1.31

Rev.

1

Date

6/73

Purpose:

To create formatted data records from unformatted data records under the control of the template record for the data record type.

Attribute:

Nonreusable

JOS Main Program:

MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
A1	Real*8	CCB
A2	Real*8	WCBARG1 - specifies the name of the data record to be formatted.
A3	Real*8	WCBARG2 - specifies the name of the potential output record.
A4	Real*8	Area addressed by MCBUFFG - data is formatted in and written to the CDS from this area.
A8	Integer*4	WXTNTGR - defines the number of qualifiers in A2.
A10	Integer*4	WXTWORK1 - used as the array for BNAM calls for the data record.
A11	Integer*2	WXTWORK2 - used as the array for BNAM calls for the data record.

Subprograms:

<u>Control Section</u>	<u>Entry Points</u>
\$TERR	\$TERR
BTFP	BTFP
CONVER*	CONVER
FLEXIT	FLEXIT
LCDINT	\$CDINT
LINK	BNAM, JTREE, WPAGE
LOADPG*	LOADPG
MOVEIT	MOVEIT
RHAR	CHAR
SHIFTB	SHIFTB
TRNSLT	TRNSLT

Data Sets:

JOSHUA Data File
Communication Data Set (CDS)

JOSHUA DPSTM-500	Title Module: DDISPLAY	Vol. IX	Page E.1.32
		Rev. 1	Date 6/7

Objective:

To format, on the CDS, the record or records indicated by A2. For each record the name of a potential output record is to be completed and placed in A3. One record is to be formatted for each execution of the module.

Program Logic:

- On the initial execution the designated subtree is established; succeeding executions will supply BNAM with the data from the preceding call which was preserved in A10 and A11. The initial execution is recognized when A10(3) is zero. To locate the first record, the standard searching procedure is used; failure results in termination through a call to \$TERR.
- On each execution, the name of a potential output record is established in A3. If A3 contains no qualifiers, the contents of A2 (less a job or user data set designation) are placed in A3 after the input record has been located. When A3 initially contains qualifiers, those qualifiers added to the original value of A2 are added to A3. The base status of A3 is determined on each execution by locating the end of the qualifiers in A3 (hexadecimal 55's) and subtracting from it the low order qualifiers added to A2 on the previous execution.
- A8 is adjusted for the possible prefixion of a qualifier to A2.
- When a deleted record is retrieved, it is ignored and BNAM immediately called for another record.
- JTREE is called to assure that the data file is on-line. When it isn't, \$TERR is called.
- The search for the template record is made first in the data set in which the data record was located; failing that, the data sets named in the default search sequence are searched. When a template cannot be found, \$TERR is called.
- Each nest in the template is worked separately. The portion of a template containing the nest must be be completely in core when the nest is being worked. Decoding of the page type and replacing of variable data fields with values from the data record are done by subroutine LOADPG which will process the first data field it encounters and continue formatting to the start of the next variable data field or the end of the page whichever is first.

- Before each call to LOADPG, data for the next variable must be available and in core. Except for variable-length items (i.e. type C), data does not span physical data records. When there are too few bytes remaining in a physical record to satisfy a fixed-length item, these remaining slack bytes are ignored. They are not deducted from the remaining length of the data record. When data is not available, the function is terminated by a call to \$TERR after having written any partially formatted page.
- As each page is completed loop start positions for continuing are picked up from the template. If the page start location is before the previous position at which this loop began (i.e. a "rolloff" is indicated), the data already formatted and the pointers to the locations at which all inner loops previously started must be adjusted by the difference between the two indicated positions.
- The low-order two bytes of each data item is maintained for possible use as a loop limit; if so used, it is assumed to be a fixed point (i.e. integer) value.
- Pages are written to the CDS as they are completed, when the nest has been completed, and when, in the case of a loop with variable limits, when the loop is completed.
- If the qualifiers entered in A2 specified a fully qualified record name (A10(4) = A10(7) after reading), A10(3) is set to zero to signal that all records have been retrieved. When more than one record is involved, this field will be set to zero by BNAM on the execution following that in which the final record was retrieved.
- When BNAM indicates that the lownam is complete (BNAM has set A10(3) to zero), termination of function execution after completion of this module is determined by a call to FLEXIT.

Terminating Conditions:

- A. BNAM errors:
 1. Unable to locate data record.
 2. Unable to locate template record.
- B. DDISPLAY errors:
 1. Invalid template.
 2. Template nest too large for area provided.
 3. Unable to accomodate data item in core.
 4. Data record too short to satisfy template.
 5. Invalid loop limits.
 6. Data set for date or template not on-line.

Language:

FORTRAN IV

Author:

J. E. Suich

DDISPLAY OUTPUT ROUTINE

JOSHUA

DPSTM-500

Title

Module: DDISPLAY

Subroutine: BTFP

Vol.

IX

Page

E.1.37

Rev.

1

Date

6/73

Purpose: To edit a floating-point value.

Attribute: Nonreusable

Usage: CALL BTFP(FLOAT, OUTPUT, NBR, WIDTH, DECMAL)

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
FLOAT	Real*8	Input floating point value.
OUTPUT	Logical*1	Output field.
NBR	Integer*4	Code for Fortran-type edit format: 2 = 'E' type 3 = 'F' type
WIDTH	Integer*4	Width of the output field.
DECMAL	Integer*4	The number of decimal fraction positions (not including the decimal point) in the output.

Subprograms:

Control Section

Entry Point

LCDINT	\$CDINT
MOVEIT	MOVEIT
RHAR	CHAR

Program Logic:

- FLOAT is adjusted so that when $\text{FLOAT} \neq 0$,
 $1.0 > |\text{FLOAT}| \geq 0.1$.
- FLOAT is rounded up to eight significant digits. The value is a function of the number of decimal positions in the edit and the adjustment made to FLOAT to place its value within the above limits.
- A determination is made as to whether the edit field can accommodate the results. If not, the edit field is filled with asterisks and the subroutine ends.
- The fractional portion of the floating point value in FLOAT is converted to the desired configuration by dividing FLOAT by the factor $16^8 \times 10^{-8}$ and the result transferred to the three low order bytes of a fullword and then adjusted for normalization if required.
- The integer value is converted (by \$CDINT) to EBCDIC and edited in the output field. If the value(I) is not defined so that $10^8 > I > -10^7$ or if the output field is too small, asterisks are placed in the output field.

JOSHUA DPSTM-500	Title Module: DDISPLAY Subroutine: BTFP	Vol. IX	Page E.1.38
		Rev. 1	Date 6/7

- For an E format, the exponent value is converted to EBCDIC and placed in the output field following the literal E and a sign (plus understood).

Language:

FORTRAN IV

Author:

W. H. Hightower - R. C. Haywood

JOSHUA

DPSTM-500

Title	Module: DDISPLAY Subroutine: CONVER	Vol. IX	Page E.1.41
		Rev. 1	Date 6/73

Purpose: To edit unformatted binary data.

Usage: Call CONVER (FMT, VAR, DATA, OUT, INDEX)

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>												
FMT	Logical*1	A two byte field describing the format. The first byte specifies a FORTRAN format code — <table><tr><th><u>byte value</u></th><th><u>format code</u></th></tr><tr><td>251</td><td>Z</td></tr><tr><td>252</td><td>A</td></tr><tr><td>253</td><td>I</td></tr><tr><td>254</td><td>E</td></tr><tr><td>255</td><td>F</td></tr></table> and 250 indicates an index field. The second byte specifies the length of the edited field and, where applicable, the number of decimal positions of the index level.	<u>byte value</u>	<u>format code</u>	251	Z	252	A	253	I	254	E	255	F
<u>byte value</u>	<u>format code</u>													
251	Z													
252	A													
253	I													
254	E													
255	F													
VAR	Logical*1	A two byte field describing the variable. The first byte is not used here. The second byte defines the length of the variable in storage.												
DATA	Logical*1	Input data												
OUT	Logical*1	Edit field. The length of OUT is one byte greater than the edit field length specified in FMT. The first byte has a colon (:) placed in it and the remaining bytes are available for data.												
INDEX	Logical*1	Array of two-byte nodes containing the current value of each loop index in the nest.												

Subprograms:

BTFFP
\$CDINT
MOVEIT
CHAR

Program Logic:

- Hexadecimal (Z)
 - Any extra high order positions in OUT are blanked.
 - To obtain the hexadecimal representation of a byte value, each half byte is considered by itself and the EBCDIC representation of its value (0-9, A-F) determined. For instance, the national character @ has a value of 124₁₀ or 7C in hexadecimal. By dividing 124 by 16, a quotient of 7 and a remainder of 12 are obtained. Each value has 183 added to it. The original quotient now has a value of 190. Since this is less than the value of an EBCDIC 'A' (193) it is increased by an additional 57 bringing the value to 247 which equals the EBCDIC '7'. The original remainder value is now greater than 193 (actually 195 which is equivalent to the EBCDIC 'C') so its conversion is complete.
 - The converted DATA is right-justified in OUT.
- Alphameric (A)
 - The value in DATA is left-justified in OUT without conversion. Depending on the relative lengths of the two fields, any extra positions in DATA are ignored, extra positions in EDIT are blank filled.
- Fixed-point (I)
 - DATA is considered to carry a fullword or halfword fixed point value depending upon VAR(2); a fullword is the default if VAR(2) ≠ 2.
 - After establishing the value as a local fullword, it is converted to EBCDIC by calling \$CDINT.
 - The result is right-justified in OUT. Any leading positions in EDIT are blanked; if EDIT is not large enough to contain the converted value, the field is filled with asterisks.
 - Due to the limitations of \$CDINT, if the converted result requires more than eight positions, the result will be asterisks in the low order positions of OUT.

JOSHUA

DPSTM-500

Title

Module: DDISPLAY
Subroutine: CONVER

Vol.

IX

Page

E.1.43

Rev.

1

Date

6/73

- Floating-Point (E & F)
 - The output field width and decimal positions are calculated from FMT(2). The value of FMT(2) equals the width plus the product of sixteen and the number of decimal positions.
 - DATA is transferred - up to eight bytes depending on VAR(2) - to a local variable on a double-word boundary.
 - BTFP is called to perform the edit.
- Index
 - The edit field length and the subscript to INDEX are determined from FMT(2). The value in FMT(2) equals the edit field length plus the product of 16 and the subscript value.
 - The value in INDEX is then handled like a fixed-point halfword value.

Language:

FORTRAN IV

Author:

J. E. Suich - R. C. Haywood

JOSHUA

DPSTM-500

Title

Module: DDISPLAY
SUBROUTINE: LOADPG

Vol.

IX

Page

E.1.45

Rev.

1

Date

6/73

Purpose:

To create an edited page.

Attribute:

Nonreusable

Usage:

CALL LOADPG (IN, INPTR, OUT, OUTPTR, DATA, TYPE, OUTSIZ, INIT)

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
IN	Logical*1	The portion of the template associated with the page type begin developed.
INPTR	Integer*2	Index to IN - preset on entry and incremented as IN is examined.
OUT	Logical*1	The edit field.
OUTPTR	Integer*2	Index for OUT - preset on entry and incremented as OUT is loaded.
DATA	Logical*1	Array containing variable data.
TYPE	Logical*1	Two byte field - the first byte specifies the variable type (I - integer; R-real; C-character) in DATA; the second byte specifies the variable length in bytes.
OUTSIZ	Integer*4	Upper limit of OUTPTR.
INIT	Integer*2	Array containing the current value of the index of each loop in the current template nest.

Subprograms:

CONVER

Objective:

Each execution is to create in OUT, to the limit OUTSIZ, the portion of a page from the point OUT(OUTPTR) through the first variable field not an index field, editing the value of DATA in that field, to the start of the next variable field.

JOSHUA DPSTM-500	Title Module: DDISPLAY SUBROUTINE: LOADPG	Vol. IX	Page E.1.46
		Rev. 1	Date 6/73

Program Logic:

- IN is decoded starting at IN(INPTR) to create the output in OUT starting at OUT(OUTPTR) to the limit OUTSIZ. Where the first data field is encountered, subroutine CONVER, is called to edit DATA into OUT and a flag turned on unless the data field is an index field in which case CONVER is called to edit the appropriate node of INIT and the flag is not turned on. If the flag is already on when a data field (including an index field) is encountered, control is returned to the calling routine.
- For each page, the pointers IN and OUT are initialized by the calling routine for the start of the page and from there on are maintained by LOADPG.

Language:

FORTRAN IV

Author:

J. E. Suich

JOSHUA DPSTM-500	Title Module: DDISPLAY Subroutine: LOADPG	Vol. IX	Page E.1.48
		Rev. 1	Date 6/73

(Blank)

Purpose:

To delete selected nodes as indicated in a list of brother nodes.

Attribute:

Nonreusable

JOB Main Program:

MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
A1	Real*8	CCB
A2	Real*8	WCBARG1 - a partially defined record name.
A4	Real*8	Area pointed to by MCBUFFG and used as the CDS buffer.
A8	Integer*4	WXTNTGR - the number of qualifiers in A2.

Subprograms:

Control Section

Entry Point

\$TERR
LINK
MOVEIT

\$TERR
BNAM, RPAGE
MOVEIT

Data Set:

Communication Data Set (CDS)

Program Logic:

- A formatted list of qualifiers is read from the CDS and each one checked for a delete request - a non-blank in the request field.
- Each qualifier marked for deletion is added to the end of A2 and, unless already deleted, functional request 5 made to BNAM to delete the node from the subtree. If a BNAM error occurs, \$TERR is called.
- When all requested deletions have been made, a close (functional request 4) is issued to BNAM.

Language:

FORTRAN IV

Author:

R. C. Haywood

Purpose: To delete a named data record or set of data records.

Attribute: Nonreusable

JOS Main Program: MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
A1	Real*8	CCB
A2	Real*8	WCBARG1 - a partially or fully qualified record name. The terminal qualifer (excluding delete indicators) defines the node to be deleted.
A8	Integer*4	WXTNTGR - the number of qualifiers in A2.

Subprograms:

<u>Control Section</u>	<u>Entry Point</u>
\$TERR	\$TERR
LINK	BNAM
MOVEIT	MOVEIT

Program Logic:

- o A2 is examined for the presence of the delete indicator '<D>' as the final qualifier (the last two qualifiers if the request is for deletion of the complete data set). If this is not in order, \$TERR is called.
- o Functional request 5 is made of BNAM with A2 stripped of delete indicators. Deletion takes place at the node defined by the final qualifier in A2. \$TERR is called if an error is encountered.
- o A close (functional request 4) is issued to BNAM.

Language:

FORTRAN IV

Author:

J. C. Jensen

JOSHUA

DPSTM-500

Title	Module: DSDUMP	Vol.	Page
		IX	E.1.53
Rev.	1	Date	2/73

Purpose: In the batch, to unload a JOSHUA data set to an external sequential file.

Attribute: Nonreusable

JOS Main Program: MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
\$MCB	Real*8	CCB - precompiler-supplied argument.
NAME	Real*8	WCBARG1 - specifies the name of the data set to be unloaded.

Subprograms: The following excludes any references developed by the JOSHUA precompiler.

<u>Control Section</u>	<u>Entry Point</u>
JTJOSDEQ	JOSDEQ
JTJOSENQ	JOSENQ

External References: The following excludes any references developed by the JOSHUA precompiler.

JERSET

Data Sets:

FT15F001 external file to which data set is unloaded.
FT20F001 JDSNC
FT21F001 JDSCB
JOSHUA tree and data files for data set being unloaded.

Program Logic:

- JERSET is executed to limit I/O errors to a single occurrence.
- If NAME specifies 'SCRATCH' it is replaced by the SCRATCH data set name.
- The catalog is searched for the specified data set; only data sets with a single extent may be unloaded.
- An ENQ is issued, the JDSCB and the file tree and data are unloaded to FT15F001, and a DEQ issued.

Terminating Conditions:

- The specified data set cannot be located.
- The data set has more than one extent.
- The ENQ fails.

Language: FORTRAN IV (JOSHUA precompiled as a module).

Author: R. C. Haywood - J. C. Jensen

JOSHUA

DPSTM-500

Title

Module: DSLOAD

Vol.

IX

Page

E.1.55

Rev.

1

Date

6/73

Purpose: To load a JOSHUA data set.

Attribute: Nonreusable

JOS Main Program: Main

<u>Arguments:</u>	<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
	\$MCB	Real*8	CCB - precompiler - supplied.
	NAME	Real*8	WCBARG1 - specifies the data set into which the file is to be loaded.

Subprograms: The following excludes any references developed by the JOSHUA precompiler.

<u>Control Section</u>	<u>Entry Point</u>
\$LINK	BNAM, JTREE
JTJOSDEQ	JOSDEQ
JTJOSENQ	JOSENQ

External References: The following excludes any references developed by the JOSHUA precompiler.

JERSET

Data Sets: JOSHUA Data Set Name Catalog (JDSNC)
JOSHUA Data Set Control Block (JDSCB)
JOSHUA tree and data files for the data set into which external data is loaded.
FT14FOO1 - the external sequential file containing the data to be loaded.

Program Logic:

- JERSET is executed to limit I/O errors to a single occurrence.
- If NAME specifies 'SCRATCH', it is replaced by the scratch data set name.
- The JDSNC is searched for the specified data set. If it either does not exist or contains more than one extent, execution is terminated by a non-zero STOP.
- Unless the receiving data set is the scratch data set or an enqueue has already been effected by this task, an attempt is made to enqueue on the receiving data set. If it fails, a non-zero STOP is issued.

JOSHUA DPSTM-500	Title Module: DSLOAD	Vol. IX	Page E.1.56
		Rev. 1	Date 6/7

- The JDSCB's of both data sets are compared. If the receiving data set's specifications are inadequate, execution is terminated by a non-zero STOP.
- The JDSCB of the receiving data set is adjusted to reflect the amount of tree and data being loaded, and the tree and data records are transferred. Note that any previously existing data in the receiving data set is destroyed.
- BNAM is called (functional request 7) to update the in-core catalogs.
- The enqueue is removed.

Language:

FORTRAN IV - (JOSHUA precompiled as a module)

Author:

J. C. Jensen

JOSHUA DPSTM-500	Title Module: DSLOAD	Vol. IX	Page E.1.58
		Rev. 1	Date 6/73

(Blank)

Purpose:

In the batch, to provide a means of compressing a data set in place while making a copy of the data set on an external sequential file.

Attribute:

Nonreusable

JOS Main Program:

DSMAINT

Arguments:

Mnemonic

Type

Description

\$MCB

Real*8

CCB - precompiler - supplied argument.

A2

Real*8

WCBARG1 - The first doubleword specifies the data set to be processed. Care must be taken to maintain this specification as WCBARG1 is used by routines executed by DSMAINT.

External references:

The following exclude any references developed by the JOSHUA precompiler.

\$YUDLY

JERSET

PRECOP

Subprograms:

The following exclude any referenced developed by the JOSHUA precompiler.

Control Section

Entry Point

JTJOSDEQ

JOSDEQ

JTJOSENQ

JOSENQ

LODINT

\$CDINT

LINK

BNAM, JTREE

Data Sets:

FT10F001

FT11F001

FT20F001

FT21F001

JDSNC

JDSCB

JOSHUA tree and data files for 'SCRATCH' and subject data sets.

JOSHUA DPSTM-500	Title Module: DSMAINT	Vol. IX	Page E.1.60
		Rev. 1	Date 6/7

Program Logic:

- JERSET is executed to suppress extended error handling of FT10F001 and/or FT11F001.
- BNAM APPEND flag is turned on.
- Any existing SCRATCH data set is deleted.
- An attempt to rerun is suppressed by checking for the record DSMAINT and terminating with an error if it is located.
- The catalog is searched for each extent of the subject data set. Each time an extent is located, the JDSCB is read and the category checked. When all extents have been examined, execution will terminate if the data set has not met the following requirements: extents all of the same category or of two different categories of which one category is zero. Any extent with a nonzero category must be of the category specified for the task.
- The sequence in which maintenance is performed is:
 1. The data with the higher classification is copied to SCRATCH (thus compressing the contents) by executing PRECOP.
 2. SCRATCH is listed by executing \$YUDLY.
 3. The contents of SCRATCH, tree and data, are dumped to FT10F001.
 4. If another category is to be processed, SCRATCH is deleted and steps 1-3 repeated, this time dumping the contents to FT11F001.
 5. A JOSHUA ENQ is issued for the subject data set, the contents of all extents of the category just processed are deleted, a DSMAINT record is written, SCRATCH copied back to the data set, and SCRATCH deleted.
 6. If two categories are involved, SCRATCH is reopened, FT10F001 is loaded with SCRATCH and SCRATCH copied to the subject data set.
 7. The DSMAINT record is deleted.
- All terminations include the issuance of a DEQ (through a call to JOSDEQ), turning off the APPEND flag, blanking the WF and RF flags and issuance of a JOSHUA STOP statement.

JOSHUA

DPSTM-500

Title

Module: DSMAINT

Vol.

IX

Page

E.1.61

Rev.

1

Date

6/73

Terminating Conditions:

- Presence of a DSMAINT record, indicating an attempt to rerun with a data set left in an unpredictable condition due to failure of a previous execution.
- Failure to allocate a SCRATCH data set.
- Failure in copying or listing the data records.
- Prior issuance of an ENQ of the data set by another task.
- Failure of JOSHUA deletes and writes issued by the main program.
- Improper specification of the data set to be maintained (nonexistent, wrong category, etc.).

Language:

FORTRAN IV (JOSHUA precompiled as a module)

Author:

R. C. Haywood - J. C. Jensen

JOSHUA

DPSTM-500

Title

Module: DSMERGE

Vol. IX

Page E.1.63

Rev. 1

Date 6/73

Purpose: To merge JOSHUA data records on an external sequential file into a JOSHUA data set.

Attribute: Nonreusable

JOS Main Program: MAIN

<u>Arguments:</u>	<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
	\$MCB	Real*8	CCB - precompiler - supplied
	DSN	Real*8	WCBARG1 - specifies the data set into which the file is to be merged.

Subprograms: This list excludes any references developed by the JOSHUA precompiler.

<u>Control Section</u>	<u>Entry Point</u>
\$LINK	BNAM, JTREE
ERRSET	ERRSET
JTJOSDEQ	JOSDEQ
JTJOSENQ	JOSENQ

External References: This list excludes any references developed by the JOSHUA precompiler.

JERSET
PRECOP

Data Sets: JOSHUA Data Set Name Catalog (JDSNC).
JOSHUA Data Set Control Block (JDSCB).
JOSHUA tree and data files for the data set into which data is merged.
JOSHUA scratch data set.
FT12F001 External sequential files containing tree and
FT13F001 data files to be merged.

Program Logic:

- The option table for extended error handling is altered to permit only a single occurrence of I/O errors except that two occurrences are allowed for a missing data definition statement (DD Card).
- The BNAM APPEND flag is turned on; it will be turned off before execution is terminated.
- If the scratch data set does not exist a non-zero STOP is issued.
- An enqueue by this task on the data set into which data is being merged is assured; otherwise, a non-zero STOP is issued.

JOSHUA DPSTM-500	Title Module: DSMERGE	Vol. IX	Page E.1.64
		Rev. 1	Date 6

- Up to two external files may be merged. If one file is being merged, it may be of a zero or non-zero category; for two files, the first file must be of non-zero category and the second of category zero. The non-zero category must be of the category indicated for the task.
- The merging of a file requires the following steps:
 1. The JDSCB is read from the external file; an end-of-file condition terminates the merger of that file.
 2. The JDSCB is examined. The following conditions result in a non-zero STOP:
 - a. apparently invalid JDSCB;
 - b. illegal category;
 - c. inadequate scratch data set.
 3. The external data set is loaded into the scratch data set. An end-of-file condition results in a non-zero STOP.
 4. The external file is closed.
 5. A BNAM call (functional request 7) is issued to update the in-core references of the scratch data set.
 6. PRECOP is executed to effect the copy of the data in the scratch data set to the output file. An error results in a non-zero STOP.
 7. If this was category zero data the merging is complete; otherwise the WRITE flag is set to zero, the data set reference number incremented, and the steps repeated.
- A delete of a possible DSMAINT record is effected.
- A CLOSE is issued and the APPEND flag turned off.
- The enqueue is lifted.

Terminating Conditions:

- No scratch data set.
- Unable to enqueue on output data set.
- Invalid JDSCB from external file.

JOSHUA

DPSTM-500

Title

Module: DSMERGE

Vol.

IX

Page

E.1.65

Rev.

1

Date

6/73

- Category level of data on external file illegal.
- Scratch data set too small.
- External file incomplete.
- Error in copying from scratch to output data set.

Language:

FORTRAN IV (JOSHUA precompiled as a module)

Authors:

R. C. Haywood - J. C. Jensen

JOSHUA

DPSTM-500

Title

Module: DUMP61

Vol.

IX

Page

E.1.67

Rev.

1

Date

6/73

Purpose:

To provide a dump of the BNAM path pool.

Attribute:

Nonreusable

JOS Main Program Name: DUMP61

Arguments:

Parameter

Type

Description

CCB

D

CCBMCB - Monitor Control Block

External References:

OPEN, READ, CHECK, CALL, WDUMP, CLOSE

Data Sets:

BNAM path pool data set (FT61F001)

Program Logic:

The address of the CCB is received via the argument list. The parameter control block is checked for, and if there is none, the record number for path one is set up as a starting record. If the path number is located in PCB, the data set is then opened via the OPEN marco. The record is read and dumped via call to WDUMP every two records. This routine dumps the first two records out of every eight. At the end of the dumping (36 records) the data set is closed and a return is issued.

Referenced By:

JTSEN

Language:

Assembler

Author:

W. H. Brotherton, Jr.

JOSHUA DPSTM-500	Title Module: DUMP61	Vol. IX	Page E.1.70
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Module: EDIT

Vol. IX

Page
E.1.71

Rev. 1

Date
6/73

MODULE EDIT

PURPOSE : CREATE THE NEXT EDIT FUNCTION NAME TO BE ATTACHED.

USAGE : ATTACH EDIT(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

<u>PARAMETER</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK.
\$A	L*1	860 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12) - NOT REFERENCED.
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32) - NOT USED.
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210) - NOT USED.
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

PROGRAM LOGIC : A SETS CCT RESTART OPTION;
 B IF 'PRECS' (SEE 'TX' DSECT) EQ 0 THEN
 FOR THE NEXT MODULE IN THE FUNCTION LIST IT
 1 CREATES THE NAME,
 2 TURNS ON THE BNAM FLAG IF APPROPRIATE, AND
 3 SETS THE SIZE BITS AND EXITS.
 OR ELSE IT
 1 SETS PRECS = 0 AND EXITS.

JOSHUA CONTROL BLOCKS USED : CCB CCT FCT MCB

SUBROUTINES USED : NONE

LANGUAGE : ASSEMBLER

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDIT	Vol. IX	Page E.1.72
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Module: EDIT\$

Vol.

IX

Page
E.1.73

Rev.

1

Date
6/73

MODULE EDIT\$

PURPOSE : EDITOR DUMMY INSTRUCTION HANDLER.

USAGE : ATTACH EDIT\$(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

<u>PARAMETER</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	I*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32) - NOT USED.
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210) - NOT USED.
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

PROGRAM LOGIC : SETS UP A TERMINAL DISPLAY ERROR MESSAGE FOR THOSE
'NXTCS' VALUES (SEE 'TX' DSECT) FOR WHICH THERE IS NO
CURRENT EDITOR HANDLER.

SUBROUTINES USED : \$DP \$MVC JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDIT\$	Vol. IX	Page E.1.74
		Rev. 1	Date 6/73

JOSHUA

DPSTM-500

Title

Module: EDITA

Vol.

IX

Page

E.1.75

Rev.

1

Date

6/73

MODULE EDITA

PURPOSE : EDITOR 'A' (ADD) INSTRUCTION HANDLER.

USAGE : ATTACH EDITA(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

<u>PARAMETER</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	T*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	T*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210).
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : THE ADD COMMAND PLACES THE EDITOR IN THE 'A' MODE WHICH IS NORMALLY TERMINATED BY PLACING A FINAL '!' AFTER THE LAST CHARACTER ADDED.

\$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

- A - NEW LINES WILL BE ADDED IN FRONT OF THE FIRST PAGE IN THE FILE. IF THE FIRST PAGE IS 1 THEN THE NEW LINES WILL BE NUMBERED .240 .241 .242 ETC., IMPLYING PAGE 0 LINES 240, 241, ETC.
- A FP1 - NEW LINES WILL BE ADDED IN FRONT OF THE LINE DESCRIBED BY 'FP1'.
- A FP1*FP2 - NEW LINES WILL REPLACE THE RANGE OF FILE POSITIONS DESCRIBED BY 'FP1*FP2'.

PROGRAM LOGIC :

- A THE FILE POSITION ARRAY, \$W, IS SEARCHED FOR THE PAGE CONTAINING THAT LINE AT WHICH NEW LINES ARE TO BE ADDED.
- B IF THE PAGE MUST BE SPLIT THEN THE LAST SEGMENT IS WRITTEN BACK INTO THE USER'S WORK SPACE. THE FIRST SEGMENT IS FORMATTED FOR A TERMINAL DISPLAY.
- C WHEN CONTROL IS RETURNED TO THE EDITOR FROM A TERMINAL DISPLAY THEN
 - 1 IF AN 'A' IS NOT FOUND IN \$A(2,11) THEN THE

JOSHUA DPSTM-500	Title Module: EDITA	Vol. IX	Page E.T.76
		Rev. 1	Date 6/73

- RETURNING PAGE IS IGNORED AND THE ORIGINAL PAGE IS REFORMATTED FOR A TERMINAL DISPLAY.
- 2 IF AN 'A' IS FOUND THEN THE RETURNING PAGE IS ACCEPTED AND WRITTEN INTO THE USER'S WORK SPACE.
 - 3 IF A '"" DID NOT APPEAR ON THE RETURNING TERMINAL DISPLAY THEN A BLANK PAGE IS FORMATTED FOR THE NEXT TERMINAL DISPLAY BUT IF A '"" DID APPEAR THEN THE NEXT TERMINAL DISPLAY IS THE PAGE AT WHICH THE ADDITION BEGAN AND EXIT IS TAKEN.

SUBROUTINES USED : \$\$DP \$\$IP \$\$PT \$\$TL \$CFD \$CLC
 \$CLI \$EXC \$MVC \$RLI JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA

DPSTM-500

Title	Module: EDITC	Vol. IX	Page E.1.77
		Rev. 1	Date 6/73

MODULE EDITC

PURPOSE : EDITOR 'C' (COPY) AND 'M' (MOVE) INSTRUCTION HANDLER.

USAGE : ATTACH EDITC(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 30 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	640 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210).
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : \$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

C FP1 FP3	- A COPY OF THE LINE IDENTIFIED BY 'FP1' WILL BE PLACED IMMEDIATELY AHEAD OF THE LINE 'FP3'.
C FP1*FP2 FP3	- A COPY OF THE LINES IDENTIFIED BY 'FP1*FP2' WILL BE PLACED IMMEDIATELY AHEAD OF THE LINE DESCRIBED BY 'FP3'.
C FP1*FP2 FP3*FP4	- A COPY OF THE LINES IDENTIFIED BY 'FP1*FP2' WILL REPLACE THE LINES IDENTIFIED BY 'FP3*FP4'.
M FP1 FP3	- THE LINE IDENTIFIED BY 'FP1' WILL BE MOVED TO A POSITION IN THE FILE IMMEDIATELY AHEAD OF THE LINE IDENTIFIED BY 'FP3'.
M FP1*FP2 FP3	- THE LINES IDENTIFIED BY 'FP1*FP2' WILL BE MOVED TO A POSITION IN THE FILE IMMEDIATELY AHEAD OF THE LINE IDENTIFIED BY 'FP3'.
M FP1*FP2 FP3*FP4	- THE LINES IDENTIFIED BY 'FP1*FP2' WILL REPLACE THE LINES IDENTIFIED BY 'FP3*FP4'.

PROGRAM LOGIC : A THE FILE POSITION ARRAY, \$W, IS SEARCHED FOR THE PAGE CONTAINING THE LINE AT WHICH LINES ARE EITHER

JOSHUA DPSTM-500	Title	Module: EDITC	Vol. IX	Page E.1.78
			Rev. 1	Date 6/73

TO BE COPIED OR MOVED. IF THE PAGE MUST BE SPLIT
 THEN TWO PAGES ARE CREATED FROM THE ORIGINAL.
 B THE FILE POSITION ARRAY, \$W, IS SEARCHED FOR THE
 PAGE CONTAINING THE LINE FROM WHICH LINES ARE EITHER
 TO BE COPIED OR MOVED. NEW PAGES ARE THEN CREATED
 FROM THOSE PAGES FROM WHICH LINES ARE EITHER BEING
 COPIED OR MOVED. IF THE MOVE VERB IS BEING USED
 THEN THE LINES ON THE ORIGINAL PAGE ARE BLANKED
 AND, IF APPROPRIATE, THE ORIGINAL PAGE REWRITTEN.

SUBROUTINES USED : \$\$DP \$\$IP \$\$PI \$CLC \$EXC \$MVC JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA

DPSTM-500

Title Module: EDITD

Vol. IX	Page E.1.79
Rev. 1	Date 6/73

MODULE EDITD

PURPOSE : EDITOR 'D' (DELETE) INSTRUCTION HANDLER.

USAGE : ATTACH EDITD(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

<u>PARAMETER</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210).
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : \$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

D FP1	- THE LINE DESCRIBED BY 'FP1' WILL BE DELETED.
D FP1*FP2	- THE RANGE OF FILE POSITIONS DES- CRIBED BY 'FP1*FP2' WILL BE DELETED.

PROGRAM LOGIC : A THE FILE POSITION ARRAY, \$W, IS SEARCHED FOR THE
PAGE ON WHICH THE FIRST LINE IS TO BE DELETED.
B THE PAGES ARE THEN READ SEQUENTIALLY AND THE
DELETED LINES BLANKED OUT AND, IF APPROPRIATE,
THE PAGES REWRITTEN INTO THE USER'S WORK SPACE.

SUBROUTINES USED : \$\$IP \$\$PI \$\$TL \$CLC \$EXC \$MVC JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDITD	Vol.	Page
		IX	E.1.80
		Rev.	Date
		1	6/73

(Blank)

JOSHUA

DPSTM-500

Title

Module: EDITG

Vol.

IX

Page

E.T.81

Rev.

1

Date

6/73

MODULE EDITG

PURPOSE : EDITOR 'G' (GET) INSTRUCTION HANDLER.

USAGE : ATTACH EDITG(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER

TYPE

DESCRIPTION

CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(90,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210).
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : THE CARD IMAGES FOUND IN THE 'GET DATASET' WILL BE CONCANNATED WITH THOSE IN THE WORK SPACE.

\$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

- G - THIS FORM OF THE G INSTRUCTION IS USED WHEN THE 'GET DATASET' IS A SEQUENTIAL DATASET.
- G MEMBER - THE 'MEMBER', FOUND IN THE PARTITIONED 'GET DATASET', WILL BE APPENDED TO THE LINES WHICH CURRENTLY EXIST.

PROGRAM LOGIC :

- A ALLOCATE AND OPEN THE 'GET DATASET'.
- B READ THE 'GET DATASET' CREATING, BY GROUPS OF EIGHT OF LINES, NEW PAGES IN THE USER'S WORK SPACE. THE NEW FILE POSITIONS ARE SAVED BEGINNING IN \$W(1,LSTPG) - (SEE 'TX' DSECT).
- C AT END CLOSE AND DEALLOCATE THE 'GET DATASET', CREATE A TERMINAL DISPLAY MESSAGE AND EXIT.

SUBROUTINES USED : \$TOP \$TIP \$TPT \$AER \$CFQ \$CLC \$CLI
\$EXC \$MVC \$RDRC \$RDRO \$RORR JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDITG	Vol. IX	Page E.1.82
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title
Module: EDITL

Vol.	Page
IX	E.1.83
Rev. 1	Date 6/73

MODULE EDITL

PURPOSE : EDITOR 'L' (LOOK) INSTRUCTION HANDLER.

USAGE : ATTACH EDITL(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

<u>PARAMETER</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210).
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : ALL LOOK COMMANDS WILL TERMINATE THE 'Q' AND 'R' MODES
AND PLACE THE EDITOR INTO THE 'L' MODE.
\$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

L	- THE NEXT PAGE TO BE DISPLAYED WILL BE THE CURRENT PAGE.
L FP1	- THE NEXT PAGE DISPLAYED WILL BE THAT PAGE CONTAINING THE FILE POSITION 'FP1'.

PROGRAM LOGIC : THE FILE POSITION, 'FP1', IS SEARCHED FOR IN THE \$W
ARRAY OF EDITOR FILE POSITIONS AND THAT PAGE WHICH IS
THE CLOSEST IS THE NEXT PAGE SETUP FOR THE TERMINAL
DISPLAY.

SUBROUTINES USED : \$\$DP \$\$IP \$CLC \$MVC JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDITL	Vol.	Page
		IX	E.1.84
		Rev.	Date
		1	6/73

(Blank)

JOSHUA

DPSTM-500

Title Module: EDITO

Vol. IX	Page E.1.85
Rev. 1	Date 6/73

MODULE EDITO

PURPOSE : EDITOR 'O' (OVERLAY) INSTRUCTION HANDLER.

USAGE : ATTACH EDITO(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'IX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210) - NOT USED.
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : \$A(2,11) WILL CONTAIN THE FOLLOWING INSTRUCTION -
 O ?FIELD1??FIELD2? - '?' IS ANY ARBITRARY CHARACTER USED TO DELIMIT BOTH 'FIELD1' AND 'FIELD2'. BEGINNING ON THE FIRST PAGE ALL OCCURANCES OF 'FIELD1' WILL BE REPLACED BY 'FIELD2'. IF 'FIELD2' IS SHORTER THAN 'FIELD1' THEN 'FIELD1' WILL BE PADDED TO THE RIGHT WITH BLANKS. IF 'FIELD2' IS LONGER THAN 'FIELD1' THEN 'FIELD2' WILL BE TRUNCATED.

PROGRAM LOGIC :

- A SEPARATE 'FIELD1' AND 'FIELD2' FROM THE ARBITRARY DELIMITER '?' AND SAVE LOCALLY.
- B BEGINNING ON THE FIRST PAGE CHECK EACH PAGE FOR THE OCCURANCE OF 'FIELD1' AND, IF FOUND, REPLACE IT WITH 'FIELD2'.
- C AFTER ALL PAGES HAVE BEEN CHECKED PREPARE A MESSAGE OF THE NUMBER OF SUBSTITUTIONS AND EXIT.

SUBROUTINES USED : \$SDP \$CDINT \$CLC \$MVC \$RLT JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDITO	Vol.	Page
		IX	E.1.86
		Rev.	Date
		1	6/73

(Blank)

JOSHUA

DPSTM-500

Title Module: EDITQ

Vol. IX	Page E.1.87
Rev. 1	Date 6/73

MODULE EDITQ

PURPOSE : EDITOR 'Q' (QUERY) INSTRUCTION HANDLER.

USAGE : ATTACH EDITQ(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
-----------	------	-------------

CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210) - NOT USED.
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : \$A(2,11) WILL CONTAIN THE FOLLOWING INSTRUCTION -
Q ?FIELD? THE QUERY COMMAND PLACES THE EDITOR INTO THE QUERY MODE. '?' IS ANY ARBITRARY CHARACTER USED TO DELIMIT 'FIELD'. THE FILE IS SEARCHED SEQUENTIALLY, BEGINNING ON THE FIRST PAGE, AND EACH PAGE CONTAINING 'FIELD' IS DISPLAYED. IF THE CURSER IS NOT ON THE TOP LINE AFTER A DISPLAY THEN THE EDITOR ASSUMES AN UPDATE HAS TAKEN PLACE. THE QUERY MODE IS TERMINATED BY EITHER A LOOK COMMAND OR AFTER ALL PAGES HAVE BEEN DISPLAYED.

PROGRAM LOGIC : A SEPARATE 'FIELD' FROM DELIMITERS AND SAVE IT IN BYTES 321-400 OF THE FIRST RECORD IN THE USER'S WORK SPACE.
B SET \$X(1) TO FALSE (R MODE SWITCH), AND SET \$X(3) TO TRUE (Q MODE SWITCH).
C BEGINNING AT THE FIRST EDITOR PAGE PRESENT TO THE TERMINAL DISPLAY ALL PAGES ON WHICH 'FIELD' APPEARS.
D AFTER ALL PAGES HAVE BEEN DISPLAYED SET \$X(3) TO FALSE, SET TERMINAL DISPLAY TO THAT PAGE ON WHICH THE ORIGINAL INSTRUCTION WAS RECEIVED AND EXIT.

SUBROUTINES USED : \$TL \$CDINT \$CLC \$MVC \$RLI JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDITQ	Vol.	Page
		IX	E. 1. 88
		Rev. 1	Date 6/73

(Blank)

MODULE EDITR

PURPOSE : EDITOR 'R' (REPLACE MODE) INSTRUCTION HANDLER.

USAGE : ATTACH EDITR(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
CCR	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32) - NOT USED.
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210) - NOT USED.
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : \$A(2,11) WILL CONTAIN THE FOLLOWING INSTRUCTION -

- R - THE REPLACE COMMAND PLACES THE EDITOR INTO THE REPLACE MODE. IF AFTER ANY DISPLAY THE CURSER IS NOT FOUND ON THE TOP LINE THEN THE EDITOR ASSUMES THE PAGE HAS BEEN UPDATED. THE REPLACE MODE IS TERMINATED BY ANY LOOK COMMAND.

PROGRAM LOGIC :

- A SET \$X(1) TO TRUE (R MODE SWITCH)
- B SET \$X(3) TO FALSE (Q MODE SWITCH)
- C CREATE A TERMINAL MESSAGE AND EXIT

SUBROUTINES USED : \$\$DP \$MVC JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDITR	Vol.	Page
		IX	E.1.90
		Rev.	Date
		1	6/73

(Blank)

JOSHUA DPSTM-500	Title Module: EDITS	Vol.	Page
		IX	E.1.92
		Rev.	Date
		1	6/73

(Blank)

Title	Module: EDITW	Vol.	Page
		IX	E.1.93
Rev.	1	Date	6/73

MODULE EDITW

PURPOSE : EDITOR 'W' (WRITE) INSTRUCTION HANDLER.

USAGE : ATTACH EDITW(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

<u>PARAMETER</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	240 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210).
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : ALL WRITES WILL BE FROM THE EDITOR WORK SPACE TO THE 'WRITE DATASET' AS DEFINED ON THE EDITOR CONTROL PAGE.

\$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

- D MEMBER <D> - THE 'MEMBER' OF THE PARTITIONED 'WRITE DATASET' WILL BE DELETED.
- W <R> REPLACE THE LINES IN THE 'WRITE DATASET' SEQUENTIAL FILE. THE LAST FIVE COLUMNS WILL CONTAIN NEW EDITOR FILE POSITIONS.
- W <RN> REPLACE THE LINES IN THE 'WRITE DATASET' SEQUENTIAL FILE. DO NOT PLACE NEW EDITOR FILE POSITIONS ON THE LINES.
- W MEMBER <A> ADD A NEW 'MEMBER' TO THE 'WRITE DATASET' PARTITIONED FILE. NEW EDITOR FILE POSITIONS WILL BE PLACED IN THE LAST FIVE COLUMNS OF THE NEW LINES.
- W MEMBER <AN> ADD A NEW 'MEMBER' TO THE 'WRITE DATASET' PARTITIONED FILE. DO NOT PLACE NEW EDITOR FILE POSITIONS ON THE LINES.
- W MEMBER <D> - THE 'MEMBER' OF THE PARTITIONED 'WRITE DATASET' WILL BE DELETED.
- W MEMBER <R> REPLACE THE OLD OR ADD A NEW

JOSHUA DPSTM-500	Title Module: EDITW	Vol. IX	Page E.1.94
		Rev. 1	Date 6/7

W MEMBER <RN>

'MEMBER' TO THE 'WRITE DATASET' PARTITIONED FILE. THE LAST FIVE COLUMNS WILL CONTAIN NEW EDITOR FILE POSITIONS. REPLACE THE OLD OR ADD A NEW 'MEMBER' TO THE 'WRITE DATASET' PARTITIONED FILE. DO NOT PLACE NEW EDITOR FILE POSITIONS ON THE LINES.

PROGRAM LOGIC : A ALLOCATE AND OPEN THE 'WRITE DATASET'.
B READ THE PAGES FROM THE USER'S WORK SPACE AND WRITE THE LINES TO THE 'WRITE DATASET'. IF THE SECOND LETTER QUALIFIER WITHIN THE < > IS NOT AN 'N' THEN USE THE LAST FIVE COLUMNS OF EACH LINE TO RECORD THE FILE POSITION OF THAT LINE.
C AT END CLOSE AND DEALLOCATE THE 'WRITE DATASET', CREATE A TUBE MESSAGE AND EXIT.

SUBROUTINES USED : \$SDP \$AER \$CDINT \$CF\$ \$CLC \$CLI
\$MVC \$RLI \$WRTC \$WTRO \$WTRW JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

MODULE EDITX

PURPOSE : EDITOR 'X' (EXECUTE) INSTRUCTION HANDLER.

USAGE : ATTACH EDITX(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 90 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	T*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	T*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210) - NOT USED.
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : \$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

X

- THE CARD IMAGES IN THE EDITOR WORKSPACE WILL BE FED TO HASP. THE FIRST IMAGE MUST BE A JOB CARD WITH THE TIME, LINES, PUNCH CARDS, AND COST CODE FILLED IN. THE JOB NAME WILL BE RECONSTRUCTED WITH A 'Q' IN THE FIRST CHARACTER FOLLOWED BY THE USER'S BADGE NUMBER. IN THE ACCOUNTING FIELD THE USER'S BADGE NUMBER WILL REPLACE THE BADGE NUMBER FIELD; THE JOB TYPE, I.E. 'T' OR 'P' WILL BE REPLACED BY AN 'E'; AND THE LAST WILL CONTAIN THE TUBE NUMBER FROM WHICH THE JOB WAS EXECUTED.

X S

SIMILAR TO THE 'X' COMMAND ABOVE EXCEPT THE JOB WILL BE GIVEN SPECIAL HANDLING AND A '#' WILL OVERLAY THE LAST DIGIT OF THE BADGE NUMBER IN THE RECONSTRUCTED JOB NAME.

PROGRAM LOGIC : A BEGINNING ON THE FIRST PAGE, READ ALL PAGES IN THE

JOSHUA DPSTM-500	Title Module: EDITX	Vol. IX	Page E.1.96
		Rev. 1	Date 6/

- USER'S WORKSPACE AND WRITE THE LINES TO THE COMMUNICATIONS DATASET.
- B CHECK THE FIRST RECORD AS BEING A VALID JOB CARD WITH THE TIME AND LINES WITHIN THE 'EXPRESS CATEGORY' I.E. 15K LINES AND 15 MINUTES.
 - C CONTROL IS THEN PASSED TO THE JOSHUA MONITOR WHICH STARTS THE INTERNAL READER ON THE COMMUNICATIONS DATASET.
 - D AFTER THE JOB HAS BEEN READ INTO HASP THE JOB NUMBER IS DISPLAYED ON THE NEXT TERMINAL DISPLAY.

SUBROUTINES USED : \$ADP \$CLC \$EXC \$MVC \$PKE
\$PLI \$UNT RCODEX JTREE WPAGE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

MODULE EDITY

PURPOSE : EDITOR 'Y' (COMPRESS WORK SPACE) INSTRUCTION HANDLER.

USAGE : ATTACH EDITY(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'IX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210).
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : \$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

Y	- COMPRESS AND RENUMBER THE LINES IN THE EDITOR WORKSPACE. ALSO PLACE THE NEW EDITOR EPS IN THE LAST FIVE COLUMNS OF EACH LINE.
Y R	- COMPRESS AND RENUMBER THE LINES IN THE EDITOR WORKSPACE. PLACE BLANKS IN THE LAST FIVE COLUMNS OF EACH LINE.
Y N	- COMPRESS AND RENUMBER THE LINES IN THE EDITOR WORKSPACE. DO NOT PLACE THE NEW EDITOR EPS IN THE LAST FIVE COLUMNS OF EACH LINE.

PROGRAM LOGIC : BEGINNING ON THE FIRST PAGE READ ALL THE PAGES IN THE USER'S WORKSPACE CREATING NEW PAGES FILLED WITH VALID LINES AND CONTAINING RECALCULATED FILE POSITIONS. IF THE 'Y' FORM OF THE INSTRUCTION WAS USED THEN ALSO SAVE THE NEW FILE POSITIONS IN THE LAST FIVE COLUMNS OF EACH LINE.

SUBROUTINES USED : \$SDP \$SDP \$FXC \$MVC \$RLI HODINT JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDITY	Vol.	Page
		IX	E.1.98
		Rev.	Date
		1	6/73

MODULE EDITZ

PURPOSE : EDITOR 'Z' (DATASET UTILITIES) INSTRUCTION HANDLER.

USAGE : ATTACH EDITZ(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	1*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	1*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	1*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210) - NOT USED.
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : \$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

- Z - THE MEMBER NAMES IN THE PARTITIONED 'WRITE DATASET' ARE DISPLAYED.
- Z S - THE MEMBER NAMES IN THE PROTECTED PARTITIONED 'WRITE DATASET' ARE DISPLAYED.
- Z G - THE MEMBER NAMES IN THE PARTITIONED 'GET DATASET' ARE DISPLAYED.
- Z GS - THE MEMBER NAMES IN THE PROTECTED PARTITIONED 'GET DATASET' ARE DISPLAYED.
- Z GZ - A BATCH JOB IS INITIATED WHICH CLEANS UP THE 'GET DATASET'.
- Z GZS - A BATCH JOB IS INITIATED WHICH CLEANS UP THE PROTECTED 'GET DATASET'.
- Z 7 - A BATCH JOB IS INITIATED WHICH CLEANS UP THE 'WRITE DATASET'.
- Z ZS - A BATCH JOB IS INITIATED WHICH CLEANS UP THE PROTECTED 'WRITE DATASET'.

PROGRAM LOGIC : A IF THE INSTRUCTION IS EITHER

JOSHUA

DPSTM-500

Title

Module: EDITZ

Vol.

IX

Page

E.1.100

Rev.

1

Date

6/7

```
      Z (Z S) FOR THE 'WRITE DATASET' OR
      Z G (Z GS) FOR THE 'GET DATASET' THEN
        1 ALLOCATE AND OPEN THE DIRECTORY.
        2 READ ALL MEMBER NAMES AND SAVE THEM IN
          JOSHUA DATASET FT77.
        3 AT END CLOSE THE DIRECTORY AND DEALLOCATE
          THE APPROPRIATE DATASET.
        4 SET PRECS=0, SET CSECT='EDIT8' (SEE THE
          TX DSECT FOR A DESCRIPTION OF THESE NMEM-
          ONICS) AND PASS CONTROL TO 'EDIT8'.
      R IF THE INSTRUCTION IS EITHER
        Z GZ (Z GZS) FOR THE 'GET DATASET' OR
        Z Z (Z ZS) FOR THE 'WRITE DATASET' THEN
          1 A BATCH JOB IS CREATED WHICH WILL 'CLEANUP'
            THE APPROPRIATE DATASET BY RUNNING AN IFCOPY
            OF THE DATASET TO ITSELF. THE JOB WILL BE
            GIVEN SPECIAL HANDLING IF THE DATASET IS
            PROTECTED I.E. IF AN 'S' APPEARED IN THE
            INSTRUCTION.
```

SUBROUTINES USED : \$SDP \$AER \$BDG \$BOX \$CC\$ \$CLI \$DIRC \$DIRO
\$DIRP \$EXC \$MVC \$PLI \$UNT JTREE WPAGE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA

DPSTM-500

Title	Module: EDITO	Vol.	Page
		IX	E.1.101
Rev.	1	Date	6/73

MODULE EDITO

PURPOSE : EDITOR'S CONTROL PAGE HANDLER.

USAGE : ATTACH EDITO(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	T*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	T*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210).
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4).

PROGRAM LOGIC :

- A IF THIS IS THE FIRST TIME IN FROM THE SELECT FRAME THEN:
 - 1 SETUP THE USER'S WORK SPACE NAME IN \$N(1-3) AND SET \$N(4) TO HEX 5'S:
 - 2 SET WRITE ON \$N(1):
 - 3 ISSUE A RNAM READ OF THE QUALIFIER NAME IN \$N AND IF THE USER'S WORK SPACE EXISTED THEN SAVE ITS SIZE AND IF THE USER CATEGORY IS A '1' VERIFY THAT THE WORK SPACE IS ALSO.
- B CREATE THE EDITOR'S CONTROL PAGE TERMINAL DISPLAY.
- C UPON RETURNING SAVE THE DISPLAY IN FT77 AND
 - 1 CREATE THE USER WORK SPACE IF IT DIDN'T EXIST ASSIGNING THE USER CATEGORY TO IT.
 - 2 VERIFY THAT THE 'GET' AND 'WRITE' DATASETS ARE CATALOGUED AND ARE EITHER PARTITIONED OR SEQUENTIAL. OTHERWISE GO BACK TO 'B' ABOVE.
- D READ FT77 AND PICK UP ANY EDITOR INSTRUCTION FOUND. IF THE INSTRUCTION IS NOT AN 'A' OR 'G' OR IF THE 'RESTART' FIELD IS A 'Y' THEN RECALCULATE THE FILE POSITIONS FOR THOSE LINES FOUND IN THE USER'S WORK SPACE. IF THE INSTRUCTION IS AN 'A' OR 'G' AND IF THE 'RESTART' FIELD IS NOT A 'Y' THEN ZERO OUT THE FILE POSITIONS AND PASS CONTROL TO THE APPROPRIATE INSTRUCTION HANDLER.

JOSHUA DPSTM-500	Title Module: EDIT0	Vol.	Page
		IX	E.1.102
		Rev.	Date
		1	6/75

COMMON USED : \$JCOMA

SUBROUTINES USED : \$\$IP \$\$PI \$\$TL \$AER \$BDG \$CDINT \$CF\$
\$CFD \$CLC \$CLI \$EXC \$MVC \$RLI \$SF\$
\$YUDB \$YUDC \$YUDEX \$YUD4 HCDINT JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA

DPSTM-500

Title Module: EDIT1

Vol.	IX	Page	E.1.103
Rev.	1	Date	6/73

MODULE EDIT1

PURPOSE : EDITOR'S RETURNING TERMINAL DISPLAY HANDLER.

USAGE : ATTACH EDIT1(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(80,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210).
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : THE FOLLOWING TEXT EDITOR INSTRUCTIONS ARE HANDLED LOCALLY WHILE ALL OTHERS ARE PASSED ON, AFTER BEING PLACED INTO \$A(2,11), TO THE APPROPRIATE INSTRUCTION HANDLER -

+INCR	THE NEXT PAGE DISPLAYED WILL BE PLUS THE 'INCR' FROM THE CURRENT.
-INCR	THE NEXT PAGE DISPLAYED WILL BE MINUS THE 'INCR' FROM THE CURRENT.
E	THIS INSTRUCTION WILL TAKE THE USER BACK TO THE EDITOR CONTROL PAGE.
FP1	THIS INSTRUCTION IS MODIFIED TO 'L FP1' AND CONTROL PASSED TO THE LOOK INSTRUCTION HANDLER.
L +INCR	THE NEXT PAGE DISPLAYED WILL BE PLUS THE 'INCR' FROM THE CURRENT.
L -INCR	THE NEXT PAGE DISPLAYED WILL BE MINUS THE 'INCR' FROM THE CURRENT.
S	THIS INSTRUCTION WILL TAKE THE USER BACK TO THE SELECT FRAME.

ALL UPDATE COMMANDS OR IF IN EITHER THE 'Q' OR 'R' MODES AND IF THE CURSER IS NOT ON THE TOP LINE OF THE RETURNING TERMINAL DISPLAY WILL CAUSE THE RETURNING DISPLAY TO REPLACE THE CURRENT PAGE.

U THE NEXT PAGE TO BE DISPLAYED WILL BE THE CURRENT PAGE.

U FPI AFTER THE UPDATE THIS INSTRUCTION IS CHANGED TO 'L FPI' AND CONTROL PASSED TO THE LOOK INSTRUCTION HANDLER.

U +INCR THE NEXT PAGE DISPLAYED WILL BE PLUS THE 'INCR' FROM THE CURRENT.

U -INCR THE NEXT PAGE DISPLAYED WILL BE MINUS THE 'INCR' FROM THE CURRENT.

PROGRAM LOGIC : A IN THE FOLLOWING ORDER CHECK FOR A COMMAND ON THE RETURNING TERMINAL DISPLAY,

- 1 \$A(1,12),
- 2 \$A(1,1),
- 3 \$A(2,11).

NOTE THAT THE JOSHUA MONITOR WILL, IF \$A(1,12) WAS BLANK, MOVE THE FIRST FOUR BYTES OF \$A(1,1) INTO \$A(1,12) AND IN ANY EVENT BLANK OUT THE FIRST FOUR BYTES IN \$A(1,1).

B IF THE COMMAND WAS EITHER AN 'E' OR AN 'S' THEN EXIT IS TAKEN IMMEDIATELY.

C THE TOP AND BOTTOM LINES OF THE RETURNING DISPLAY ARE CLEARED TO BLANKS.

D THE RETURNING EDITOR LINES ARE CHECKED FOR "" AND IF FOUND THE FIRST TWO ARE CONVERTED TO FILE POSITIONS USING THE \$S ARRAY WHICH CONTAINS THE BEGINNING AND ENDING FILE POSITIONS OF EACH LINE ON THE RETURNING DISPLAY.

E IF 'NXTBR' (SEE 'TX' DSECT) IS GT 10 THEN EXIT IS TAKEN WHERE CONTROL WILL PASS TO AN EDITOR MODULE WHOSE NAME, PREVIOUSLY DEFINED, APPEARS IN 'CSECT' (SEE 'TX' DSECT).

F IF IN EITHER THE 'Q' OR 'R' MODES AND THE CURSER WAS NOT ON THE TOP LINE OF THE DISPLAY I.E. \$S(2,20) .NE. 0, THEN THE RETURNING PAGE WILL REPLACE THE ORIGINAL IN THE USER'S WORK SPACE.

G IF THE COMMAND IN \$A(2,11) IS BLANK THEN EITHER DEFAULT IT TO 'L +1' IF IN THE 'L' MODE OR TO 'U +1' IF IN EITHER THE 'Q' OR 'R' MODES.

H IF THE COMMAND IS ONE OF THOSE MENTIONED UNDER COMMENTS ABOVE THEN IT IS HANDLED LOCALLY AND THE NEXT TERMINAL DISPLAY IS PREPARED OR ELSE EXIT IS TAKEN WHERE CONTROL WILL PASS TO THE APPROPRIATE EDITOR MODULE.

SUBROUTINES USED : \$\$DP \$\$IP \$\$PI \$\$SC \$CLC \$CLI \$MVC \$RLI JTREE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

MODULE EDIT8

PURPOSE : EDITOR '8' (DATASET UTILITIES).
THIS IS A CONTINUATION OF THE 'Z', 'Z G', 'Z GS',
AND 'Z S' INSTRUCTIONS INITIALLY HANDLED IN 'EDITZ'.

USAGE : ATTACH EDIT8(CCB,\$A,\$C,\$S,\$W,\$N)

ARGUMENTS :

PARAMETER	TYPE	DESCRIPTION
CCB	VARIOUS	JOSHUA CCB CONTROL BLOCK - NOT REFERENCED.
\$A	L*1	960 BYTE AREA USED FOR TERMINAL DISPLAYS. DIMENSIONED \$A(20,12).
\$C	VARIOUS	JOSHUA 80 BYTE AREA WHICH HOLDS THE EDITOR CONTROL FIELDS DESCRIBED BY THE 'TX' DSECT.
\$S	I*2	128 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$S(2,32).
\$W	I*2	840 BYTE AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$W(2,210) - NOT USED.
\$N	R*8	STANDARD AREA PASSED BY JOSHUA TO EACH EDITOR ROUTINE. DIMENSIONED \$N(4) - NOT USED.

COMMENTS : \$A(2,11) WILL CONTAIN ONE OF THE FOLLOWING INSTRUCTIONS -

Z	- THE MEMBER NAMES IN THE PARTITIONED 'WRITE DATASET' ARE DISPLAYED.
Z G	- THE MEMBER NAMES IN THE PARTITIONED 'GET DATASET' ARE DISPLAYED.
Z GS	- THE MEMBER NAMES IN THE PROTECTED PARTITIONED 'GET DATASET' ARE DISPLAYED.
Z S	- THE MEMBER NAMES IN THE PROTECTED PARTITIONED 'WRITE DATASET' ARE DISPLAYED.

PROGRAM LOGIC :

- A READ FT77 AND PREPARE TERMINAL DISPLAYS OF THE 'MEMBER NAME', 40 TO THE PAGE (SEE EDITZ MODULE).
- B WITH EACH RETURNING DISPLAY SAVE THE ACTION CODES (1-PRINT, 2-PUNCH, 4-DELETE OR ANY SUMMATION THEREOF) ASSOCIATED WITH EACH 'MEMBER NAME'. PREPARE THE NEXT GROUP OF 40 NAMES, ROLLING BACK AROUND TO THE FIRST AFTER ALL HAVE BEEN DISPLAYED.
- C DISPLAYING THE 'MEMBER NAMES' IS TERMINATED WHEN

JOSHUA DPSTM-500	Title Module: EDIT8	Vol. IX	Page E.1.106
		Rev. 1	Date 6/77

ONE OF THE FOLLOWING CODES APPEARS IN THE FIRST COLUMN OF EITHER THE FIRST OR LAST LINE OF THE DISPLAY:

- 'S' TAKES THE USER BACK TO THE SELECT FRAME;
- 'F' TAKES THE USER BACK TO EDITOR'S INITIAL PAGE.
- 'X' TAKES THE USER TO A SUBSEQUENT FRAME WHERE HE MUST FILL IN JOB CARD INFORMATION NEEDED TO COVER A BATCH JOB WHICH WILL PERFORM THE ACTIONS DESCRIBED BY THE CODES (SEE B ABOVE).
- 'X S' IS THE SAME AS 'X' ABOVE WITH THE ADDED FEATURE THAT THE DATASET IN QUESTION IS IDENTIFIED AS PROTECTED. THIS IS NOT NECESSARY IF THE ORIGINAL 'Z' INSTRUCTION IDENTIFIED IT AS PROTECTED.
- 'L' TAKES THE USER BACK TO THE EDITOR PAGE ON WHICH THE ORIGINAL 'Z' INSTRUCTION OF WHICHEVER FORM WAS KEYED.

D IF THE INSTRUCTION ENDING THE DISPLAY OF THE MEMBER NAMES WAS EITHER 'X' OR 'X S' THEN A JOB CARD FRAME ON WHICH THE BOX, TIME, LINES, CARDS, AND COST CODE WILL BE DISPLAYED. THERE ARE SEVEN WAYS TO EXIT FROM THIS FRAME:

- 1 A S/E (SHIFT ENTER) WITH NO MODIFICATIONS WILL CAUSE A BATCH RUN TO BE CREATED WITH THE DEFAULTS SHOWN ON THE SCREEN;
- 2-6 ARE IDENTICAL TO THOSE DESCRIBED UNDER C ABOVE EXCEPT THAT THE 'X' AND 'X S' WILL NOW CAUSE THE BATCH JOB TO BE CREATED;
- 7 A 'Z' WILL TAKE THE USER BACK TO THE MEMBER DISPLAYS.

SUBROUTINES USED : \$\$\$P \$RDG \$BOX \$CC\$ \$CDINT \$CLC \$CLI \$EXC
 \$MVC \$RLI BCDEFX JTREE LCDINT WPAGE

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA

DPSTM-500

Title Module: EDUTIL

Vol. IX

Page
E.1.107

Rev. 1

Date
6/73

MODULE EDUTIL

PURPOSE : EDITOR'S BATCH UTILITY PROGRAM.

USAGE : LOAD EDUTIL

COMMENTS : THIS PROGRAM IS USED WHEN A TEXT EDITOR USER WISHES TO RUN THE EDITOR'S UTILITY PROGRAM AGAINST HIS PARTITIONED DATASET LIBRARY. BASED ON INPUT CREATED AT THE TUBE THIS ROUTINE WILL PERFORM ANY COMBINATION OF PRINT, PUNCH, AND DELETION OF THE 'MEMBERS' IN THE LIBRARY.

PROGRAM LOGIC : A READ THE FIRST CONTROL CARD WHICH CONTAINS THE
 NUMBER OF 'MEMBERS' AND THE DATASET NAME OF THE
 USER'S LIBRARY.
 B LOOP THROUGH ALL THE INCOMING 'MEMBER NAMES' PER-
 FORMING THE REQUIRED ACTIONS I.E. PRINT, PUNCH,
 AND/OR DELETE.

SUBROUTINES USED : \$CDINT \$RDRC \$RDRO \$RDRR \$RLI \$WTRD
 IBCOM# IBMJUL JULDAT TIME XLOC

LANGUAGE : FORTRAN

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title Module: EDUTIL	Vol. IX	Page E.1.108
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Module: FMBUILD

Vol.

IX

Page

E.1.109

Rev.

1

Date

6/73

Purpose: To make the requested alterations to facemap data.

Attribute: Nonreusable

JOS Main Program: MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
A1	Real*8	CCB
A2	Integer*2	Starting at WCBARG1, the contiguous area defined by WCBARG1 and WCBARG2. It is used for the communication of data as described in module FMSTART or RMSTART.
A4	Logical*1	The area addressed by MCBBUFFG - the display buffer.
A13	Integer*4	A13 is equated to the start of WXTWORK3. The actual area used is the contiguous block starting twenty-eight bytes before WXTWORK3 in WXTWORK2 and including all of WXTWORK3 and WXTWORK4. It is used to communicate the coded assembly symbols.
A14	Integer*4	MCBCFCT.

Subprograms:

<u>Control Section</u>	<u>Entry Point</u>
BCDFIX	BCDFIX
FMPACK	FMPACK
FMSCAN*	FMSCAN
FMSWCH*	FMSWCH
MNUREV*	MNUREV
MOVEIT	MOVEIT
XYHEX	XYHEX
ZCOMP	ZCOMP

JOSHUA DPSTM-500	Title	Module: FMBUILD	Vol. IX	Page E.1.110
			Rev. 1	Date 6/77

Program Logic:

- If the 'end' option was set to 'S', A15 is incremented by 16 and a return issued.
- The list of assemblies is updated by MNUREV.
- The requested symmetry, replacement, and substitute characters are loaded into their respective positions in A2.
- FMSWCH is called to set up the proper reactor type.
- FMSCAN is called to scan the facemap display in order to modify the data in A13.
- If the 'end' option was set to 'E' a return is issued; otherwise, the X and Y coordinates and the window type requests are picked up, A15 decremented by 48 and then a return issued.

Language:

FORTRAN IV

Author:

R. C. Haywood - J. E. Suich

Purpose:

To modify the MAP array to reflect changes made by the user on the current CRT buffer.

Usage:

CALL FMSCAN(TUBE, MAP, OX, OY, WINDOW, SYM, REP, SUB)

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
TUBE	Logical*1	DISPLAY buffer area.
MAP	Logical*1	Array of 868 codes assembly symbols.
OX	Integer*2	X value of lower left corner of "window".
OY	Integer*2	Y value of lower left corner of "window".
WINDOW	Integer*2	Literal window type.
SYM	Integer*2	Literal symmetry class assembly
REP	Integer*2	symbol to be replaced by SUB.
SUB	Integer*2	Assembly symbol to be used in place of REP.

Subprograms Used:

MOVEIT
PACK
ZCOMP

Program Logic:

The window, symmetry, and replacement values are interpreted to control the modification of the MAP array from the CRT buffer, as follows: If the symmetry declaration is non-numeric, do not modify the MAP. Next, the window held in the CRT buffer is scanned, assembly by assembly. If the buffer symbol is 'blank-blank' or if the MAP array contains the same symbol as the CRT buffer (after any necessary replacements of REP symbols with SUB symbols), then no changes are made in the MAP array for that assembly. Otherwise, the new symbol from the CRT buffer is stored in the MAP array for that assembly and for all other assemblies located symmetrically as given by the symmetry option. S: 2, 3, or 6 indicates 2, 3, or 6 fold symmetry. Any other numeric value will give 1-fold or no symmetry.

Language:

FORTRAN IV

Author:

J. E. Suich

JOSHUA DPSTM-500	Title Module: FMBUILD Subroutine: FMSWCH	Vol. IX	Page E.1.114
		Rev. 1	Date 6/7

Purpose: To interchange facemaps for C, K and P reactors.

Usage: CALL FMSWCH(RTYPE, TUBE, MAP).

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
RTYPE	Integer*2	Existing reactor type.
NTYPE	Logical*1	Reactor type requested.
MAP	Logical*1	Array of 868 facemap symbols.

Subprograms Used:

FMPACK
 XYHEX
 MOVEIT

Program Logic:

If NTYPE is VALID ('C', 'K', 'P', 'A' or blank) but is neither 'A', blank, nor equal to the value of RTYPE, a reactor type switch is performed.

The symbols in the MAP array corresponding to sparjet locations for K (or P) and C type reactors are interchanged.

The control locations in the C-type buckled zone are changed to "?" unless NTYPE specifies "C" type, in which case these locations are set to "C" for "control".

NOTE: No shift is made in changing from K to P or from P to K type reactors.

Language:

FORTTRAN IV

Author:

J. E. Suich

Restrictions:

EBCDIC used in data statements.

JOSHUA DPSTM-500	Title Module: FMBUILD Subroutine: MNUREV	Vol. IX	Page E.1.116
		Rev. 1	Date 6/

Purpose: To make requested addition or deletion to the list of assembly symbols and mnemonics.

Usage: CALL MNUREV(MENU, TUBE, NMEN, IERR)

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
MENU	Integer*2	Set of forty six-byte nodes, each node consisting of a two-byte assembly symbols and a related four-byte mnemonic.
TUBE	Logical*1	Display buffer of 960 bytes containing the request for change.
NMEN	Integer*2	Number of nodes defined in MENU.
IERR	Integer*2	Error flag. Although only addition and deletion requests are handled here, the request is checked for validity and IERR set to one if the test fails and zero if it passes. IERR is also set on for an invalid assembly symbol or exceeding forty entries in MENU.

Subprograms:

FMPACK
MOVEIT

Program Logic:

- o The request is checked. If not for an addition or deletion to MENU, a return is issued after setting IERR.
- o If the assembly symbol is blank a return is issued; if a question mark a return is issued after setting IERR to one.
- o MENU is searched for the assembly symbol. If it is not found in MENU and the number of entries in MENU is less than forty, the symbol is added to the list if the request is to add; a return is issued for a delete request when the symbol is not found.
- o When a symbol is found in the list, if an addition is requested the mnemonic is replaced; if a deletion is requested, the entry is removed and the list closed up.

Language:

FORTRAN IV

Author:

R. C. Haywood

JOSHUA DPSTM-500	Title Module: FMBUILD Subroutine: MNUREV	Vol. IX	Page E.1.118
		Rev. 1	Date 6/73

(Blank)

Purpose: To write the record INPUT.GAUGE.GEOMETRY.? as specified by the facemap.

Attribute: Nonreusable

JOS Main Program: MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
A1	Real*8	CCB.
A2	Integer*2	Starting at WCBARG1, the contiguous area defined by WCBARG1 and WCBARG2. It is used for the communication of data as described in module FMSTART.
A4	Logical*1	The area addressed by MCBBUFFG the display buffer.
A13	Integer*2	A13 is equated to the start of WXTWORK3. The actual area used in the contiguous block starting twenty-eight bytes before WXTWORK3 in WXTWORK2 and including all of WXTWORK3 and WXTWORK4. It is used to communicate the coded assembly symbols.
A15	Integer*4	MCBCFCT.

Subprograms:

<u>Control Section</u>	<u>Entry Point</u>
\$TERR	\$TERR
FMPACK	FMPACK
FMUNPK	FMUNPK
IPUT	IPUT, JPUT
LCDINT	\$CDINT
LINK	BNAM, JTREE
MOVEIT	MOVEIT
XLOC	XLOC
XYHEX	XYHEX

Program Logic: The list of assembly symbols is scanned to determine all assembly symbols listed. Each symbol listed is given an index corresponding to its position in the menu. All symbols not listed in the menu are given an index of zero.

JOSHUA DPSTM-500	Title	Module: FMEND	Vol.	Page
			IX	E.1.120
			Rev.	Date
			1	6/

From the reactor type, the number of patches and the radius of the tank are determined.

A sweep is made, patch by patch, over all hexes in the reactor. The X-Y coordinates of the hex are obtained and checks are made to determine if the hex is inside the reactor boundary. Hexes inside the reactor are given the index of the corresponding assembly symbol. Hexes outside the reactor are given either the index of the moderator or the index of the reflector, depending on whether or not they are inside or outside the reactor tank.

If, during the sweep, a hex is found with the symbol "?" or with a symbol not in the menu, the CRT buffer is set up for another display to correct the error. The window portion is set to display the patch containing the error, other options (END and Symmetry) are reset to their initial value, thirty-two subtracted from A15, and a return issued.

If no errors are found the geometry record is written. A call to BNAM locates a position in the user data set for the record. A call to IPUT initializes the buffering routine. The number of patches and a value of 1 are transferred to the buffer. For each patch, the patch number and the index (padded out to make 4-byte word) for the seven hexes in the patch are transferred to the buffer. Then the number of entries in the assembly symbol list and the list itself are transferred to the buffer. A final call to IPUT assures transfer of any partially-filled buffer. A BNAM close issued to update the TREE pages.

Language:

FORTRAN IV

Author:

J. E. Suich

JOSHUA DPSTM-500	Title Module: FMEND	Vol. IX	Page E.1.122
		Rev. 1	Date 6/73

(Blank)

Purpose: To establish the facemap display.

Attribute: Nonreusable

JOS Main Program: MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
A1	Real*8	CCB
A2	Integer*2	The contiguous area defined by WCBARG1 and WCBARG2 used for the communication of data as described in module FMSTART or RMSTART.
A4	Logical*1	Area addressed by MCBUFFG - the display buffer.
A13	Integer*4	A13 is equated to the start of WXTWORK3. However the actual area utilized covers the contiguous block starting twenty-eight bytes before WXTWORK3 in WXTWORK2 and including all of WXTWORK3 and WXTWORK4. It is used to communicate the coded assembly symbols.

Subprograms:

<u>Control Section</u>	<u>Entry Point</u>
FMUNPK	FMUNPK
HEXXY	HEXXY
LCDINT	\$CDINT, HCDINT
MOVEIT	MOVEIT
MNUFIL*	MNUFIL
MOVEIT	MOVEIT
SETMAP*	SETMAP
ZCOMP	ZCOMP

Program Logic:

- The map of the requested portion of the reactor is prepared.
- The list of assembly symbols and mnemonics is prepared.
- The remaining control characters and symbols are loaded into the buffer.

Language: FORTRAN IV

Author: R. C. Haywood - J. E. Suich

JOSHUA

DPSTM-500

Title

Module: FMFORM
Subroutine: MNUFIL

Vol.

IX

Page

E.1.125

Rev.

1

Date

6/73

Purpose:

To accept requests for and make changes to the assembly mnemonics display.

Usage:

CALL MNUFIL(MENU, TUBE, NMEN, PAGE)

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
MENU	Integer*2	Array of six-byte nodes containing 2-byte assembly symbols and 4-byte mnemonics.
TUBE	Logical*1	Display buffer of 960 bytes.
NMEN	Integer*2	Number of entries defined in MENU.
PAGE	Integer*2	The 'page' of assembly symbols and mnemonics being displayed. Pages are disjoint sets of 12 contiguous entries in MENU.

Subprograms:

MOVEIT

Program Logic:

- Display requests may take the form of either 'Pn' which requests page n of MENU where $0 < n < 5$; or 'DI' which requests the page of MENU which includes some specified assembly symbol. When any request cannot be satisfied, page 1 is the default.
- The specified portion of MENU is loaded into the appropriate area in TUBE.

Language:

FORTTRAN IV

Author:

R. C. Haywood

JOSHUA

DPSTM-500

Title

Module: FMFORM
Subroutine: SETMAP

Vol.

IX

Page

E.1.127

Rev.

1

Date

6/73

Purpose:

To extract the requested window of the facemap from the array of assembly symbols and to position the symbols in the display buffer.

Usage:

CALL SETMAP(TUBE, MAP, OX, OY, WINDOW, OXP, OYP)

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
TUBE	Logical*1	Buffer area.
MAP	Logical*1	Array of 868 assembly symbols.
OX	Integer*2	Requested X-origin.
OY	Integer*2	Requested Y-origin.
WINDOW	Logical*1	Requested window type.
OMP	Integer*2	Actual X-origin.
OYP	Integer*2	Actual Y-origin.

Subprograms:

FMUNPK
HEXXY

Program Logic:

First the limits of the display are set up. The lower left corner of the display is set to OX and OY. The number of lines in the display is set at 31 and the limits of the X index to be used are set up.

If the requested window is a cluster, the lower left corner is shifted to properly position the cluster on the display. The number of lines and the X limits are reduced to the appropriate cluster size.

Next the program sweeps over the display setting up each line in sequence. The Y index for this row is obtained; if it is out of range, the line is skipped. The left and right X limits, truncated by the reactor edge, are calculated. If the window is a sector, the X limits are further reduced to exclude points not in the appropriate sector. If, in any of these steps, it is determined that no requested points occur in this line, the line is immediately skipped.

Within the calculated X limits, the program sweeps over the X values in the line. If the window is a gang, HEXXY is used to obtain the patch/point coordinates of the particular point and its ring number is checked to see if it lies in the requested gang. If the window is a cluster, HEXXY is used to obtain the patch/point coordinates relative to the center of the cluster. If the ring number is not within the appropriate limits, the point is skipped.

JOSHUA DPSTM-500	Title Module: FMFORM Subroutine: SETMAP	Vol. IX	Page E.1.128
		Rev. 1	Date 6/7

For any point which has passed these tests, the assembly symbol is unpacked and placed in the CRT buffer, preceded by a colon.

Finally OXP and OYP are set to the values actually used as the lower left corner of the display so that they will have a consistent meaning in the remaining subroutines which do not depend on whether or not the window is a cluster.

Language:

FORTRAN IV

Author:

J. E. Suich

Restrictions:

EBCDIC assumed.

JOSHUA DPSTM-500	Title Module: FMFORM Subroutine: SETMAP	Vol. IX	Page E.1.130
		Rev. 1	Date 6/73

(Blank)

Purpose:

In the FM function, to initialize the data arrays.

Attribute:

Nonreusable

JOS Main Program:

MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
A1	Real*8	CCB
A2	Real*8	WCBARG1 - contains on input the full name of the input geometry record.
A3	Real*8	WCBARG2 - may contain the full name of the output geometry record.
A2 is also considered as the total area defined by WCBARG1 and WCBARG2 and is used for communication between sub-functions of the following:		
	<u>Byte</u>	<u>Data</u>
	1-8	Variable qualifier of the input record.
	9-16	Variable qualifier of the output record.
	17-18	Reactor type.
	19-20	X-coordinate.
	21-22	Y-coordinate
	23-24	Window type.
	25-26	Symmetry type.
	27-28	Replacement character.
	29-30	Substitute character.
	31-32	Menu size.
	33-272	Menu for up to forty two-character assembly symbols and associated four-character assembly mnemonic.
	273-274	Menu page number.

JOSHUA DPSTM-500	Title Module: FMSTART	Vol. IX	Page E.1.132
		Rev. 1	Date 6/7

A4	Logical*1	Area pointed to by MCBUFFG - the CRT buffer.
A13	Integer*4	Defined as WXTWORK3, the actual area utilized includes all of WXTWORK3 and WXTWORK4 and the last seven fullwords of WXTWORK2 (which immediately precedes WXTWORK3). The area is used for communicating the facemap assembly sequence numbers.

Subprograms:

<u>Control Section</u>	<u>Entry Point</u>
\$TERR	\$TERR
FMINIT	FMINIT
FMPACK	FMPACK
FMREAD*	FMREAD
HEXXY	HEXXY
LINK	BNAM, JTREE
MOVEIT	MOVEIT
XYHEX	XYHEX

Program Logic:

- The input record name is assured of being of the form INPUT.GAUGE.GEOMETRY.? with or without a preceding data set designation.
- When an output record is specified it is checked for proper form; when it is not specified, the input record is assumed for output also.
- The variable qualifier for both records is loaded into A2.
- The default specifications for the display are established in A2 and A13; A4 is blanked.
- A BNAM read is issued to attempt to locate the input record. When successful, FMREAD is called to establish facemap specifications in accordance with that record. When the record does not exist, the default specifications are allowed to stand and execution terminated normally. An exception is when a specific data set is given in which case failure to locate the record results in a call to \$TERR.

Terminating Conditions:

- Invalid record name.
- Failure to locate a record when the data set search sequence is overridden by a specific data set request.

Language:

FORTRAN IV

Author:

J. E. Suich - R. C. Haywood

JOSHUA

DPSTM-500

Title

Module: FMSTART
Subroutine: FMREAD

Vol.

IX

Page

E.1.134

Rev.

1

Date

6/73

Purpose:

To read the geometry record and establish the facemap (X,Y) - coordinates.

Usage:

CALL FMREAD(JDS, JREC, TYPE, MAP, MENU, NMEN)

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
JDS	Integer*4	The reference number (FT-number) of the data set on which the specified geometry record resides.
JREC	Integer*4	The relative record number on JDS at which the specified geometry record begins.
TYPE	Integer*2	Changed to 'C' if the number of points in the geometry record is 169.
MAP	Logical*1	Array to be loaded with 868 assembly sequence numbers.
MENU	Integer*2	Array to be loaded with up to 40 2-character assembly symbols and related 4-character assembly mnemonics.
NMEN	Integer*2	Number of assembly symbols defined.

Subprograms:

FMPACK
JTREE
MOVEIT
XYHEX

Program Logic:

- The geometry record is read. If the number of points is neither 127 nor 169, a return is issued.
- The GAUGE patch point representation is converted to SRP (X,Y) - coordinates.
- The sequence number of each assembly is loaded from the buffer into the appropriate location in MAP for temporary storage.
- MENU is loaded from the record.
- Using the data stored in MAP and MENU, the values in MAP are converted to one byte codes representing the assembly types defined for those positions.
- If the number of points is 169, the reactor type is changed to 'C'.

Language:

FORTRAN IV

Author:

J. E. Suich

JOSHUA DPSTM-500	Title Module: FMSTART Subroutine: FMREAD	Vol. IX	Page E.1.136
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title	Module: FRECHECK	Vol.	Page
		IX	E.1.137
Rev.	1	Date	6/73

Purpose: To provide check for data overlap in JOSHUA tree and data base.

Attribute: Nonreusable

JOS Main Program Name: FRECHECK

Subprograms Used: JTREE
MOVEIT
*TRESRT
*SIFTUP
*SIFTU2

Data Sets: JOSHUA Data Set Name Catalog
JOSHUA Data Set Control Block
SYSOUT Data Set

Objective: The objective is to check the JOSHUA Data Set Control Block information for overlaps of data. The overlap could be found in the tree or data base.

Program Logic: The JOSHUA Data Set Name Catalog is read via a CALL to JTREE with the appropriate arguments. There are four records that contain the data set names. Each of these records are read and if the data set names are other than FREE, their corresponding record is read from the JOSHUA Data Set Control Block with the information contained in the record stored in an internal area. TRESRT is then called to sort on the tree catalog reference number and tree catalog base. Then a tree overlap is checked for from sort. If there is no space allocated, no check is made. Certain tree reference numbers require no check. The sorted data is checked against the Free Space Elements associated with the data set. This is gone through with the data reference numbers and base address after the tree is done. Upon reaching some type overlap the data at which the error occurred is written out and then the entire data is written out in sorted form to aid in solving the problem. An abnormal termination is also induced on an error condition.

Terminating Condition: On finding an error the program will abnormally terminate with system code of OCI.

Referenced by: JCATALOG

Language: Fortran IV

Author: W. H. Brotherton, Jr.

JOSHUA DPSTM-500	Title Module: FRECHECK Subroutine: TRESRT	Vol. IX	Page E.1.140
		Rev. 1	Date 6/

Purpose: To sort given array.

Usage: CALL TRESRT(A,L,M)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
A	I*2	Array to be sorted.
L	I*4	Number of groups of 20 I+2's to be sorted.
M	I*4	Indicates where sort is desired.

Subprograms Used: *SIFTUP
*SIFTU2

Program Logic: If no sort is necessary (one array of 20 integers), a return is issued. It then goes through SIFTUP or SIFTU2 depending on whether M=1 or M=2, respectively. It goes through this the desired number of times to complete the sort using a work area as needed. It then calls MOVEIT when sort is completed on each group.

Language: FORTRAN IV.

Authors: R. S. Wood, W. H. Brotherton, Jr.

JOSHUA

DPSTM-500

Title Module: FRECHECK
 Subroutine: SIFTUP, SIFTU2

Vol. IX	Page E.1.142
Rev. 1	Date 6/73

Purpose: To sort desired fields.

Usage: CALL SIFTUP(A,L,N)
 SIFTU2(A,L,N)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
A	I*2	Array to be sorted.
L	I*4	Field to compare to.
N	I*4	Number of compares to make.

Subprograms Used: MOVEIT

Program Logic: The routine saves the array section that compares will be made against; this is done by the MOVEIT routine. The sorting is then done by a series of arithmetic if's. SIFTU2 just reverses the if's in SIFTUP. After this is gone through for all of the array, MOVEIT is called to put the correct group in place. A return is then issued.

Language: FORTRAN IV

Authors: R. S. Wood, W. H. Brotherton, Jr.

JOSHUA DPSTM-500	Title Module: FRECHECK Subroutine: SIFTUP, SIFTU2	Vol. IX	Page E.1.144
		Rev. 1	Date 6/73

(Blank)

Title	Module: GALLOC	Vol. IX	Page E.1.145
		Rev. 1	Date 6/73

Purpose: To give JOSHUA terminal users the ability to allocate and unallocate 2260 graphic devices.

Attribute: Nonreusable

JOS Main Program Name: GALLOC

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
REQUEST	F	Request element, where first character is an 'A' for allocate, and 'U' for deallocate. The last three characters are the device address.
DEVICE ADDRESS	F	Device address is returned as 4 EBCDIC characters.

External References: SVC 30, SVC 38, SVC 251

Objective: The objective is to allow JOSHUA terminal users to have all terminals available for production usage with the ability to free a terminal (deallocate) which in turn can be allocated to a test program in the batch mode. The test terminal can be reallocated back to a production status after completion of the batch program.

Program Logic: A search for the UCB, which corresponds to the device address passed in the first argument, is made down the UCB chain. After finding the proper UCB, the following checks are made before allocation takes place:

1. Device must be deallocated.
2. Device must be online.
3. It must be a 2260 Mod I.
4. No DCB's can be open to this device.

The allocation bits in the UCB are turned on after the above checks have been made.

Deallocation requests are handled in a similar manner with allocation bits being turned off in the UCB.

Register 15 is used for a completion code return; a value of zero indicates a successful completion. Other return codes and their meanings are shown in the attached table.

JOSHUA DPSTM-500	Title Module: GALLOC	Vol. IX	Page E.1.146
		Rev. 1	Date 6/73

Language:

Assembler

Author:

F. C. Fortune

TABLE I

RETURN CODES FOR GALLOC

Contents of
Register 15

00	Successful completion.
01	UCB could not be found.
02	Device already allocated to another job.
03	Device is not online.
04	System is enqueued on allocation.
05	Device requested is not a 2260.
06	Device is being used by another job (deallocation request). No action taken.
07	Device allocation bit in UCB was not on (deallocation request). No action taken.

JOSHUA DPSTM-500	Title Module: GALLOC	Vol. IX	Page E.1.150
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Module: GETBRTHR

Vol.

IX

Page

E.1.151

Rev.

1

Date

6/73

Purpose:

Given an incomplete record name, to provide a list of the son node and all its brother nodes so that any of them may be marked for deletion.

Attribute:

Nonreusable

JOS Main Program:

MAIN

Arguments:

Mnemonic

Type

A1	Real*8	CCB
A2	Real*8	WCBARG1 - the partially defined record name
A4	Real*8	Area pointed to by MCBUFFG and used as the CDS buffer.
A8	Integer*4	WXTNTGR - the number of qualifiers in A2.

Subprograms:

Control Section

Entry Point

\$TERR	\$TERR
LINK	BNAM, WPAGE
MOVEIT	MOVEIT

Data Sets:

Communication Data Set (CDS).

Program Logic:

- Functional symbol '\$?\$\$' is added as a final qualifier to A2. This marks the level in the subtree which the nodes are to be listed.
- A series of calls (functional request 11) is made to BNAM retrieving on each call a number of brothers up to the number sufficient to complete a formatted CDS record which is written by a call to WPAGE.
- The first CDS record contains instructions for the user to indicate which nodes are to be deleted.
- If the data set from which the retrievals are made is not the 'write' data set, \$TERR is called.

Language:

FORTRAN IV

Author:

R. C. Haywood - J. C. Jensen