THE JOSHUA SYSTEM

Vol. 9 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.~~

October 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 A Precompiler

Part B Data Management

Part C JOSHUA Monitor

JOSHUA

DPSTM-500

Title

Part A. Precompiler

Vol.

IX

Page

A.O.1

Rev.

1

Date

6/73

Contents

- A.1 Language Specifications
- A.2 Precompiler Module
- A.3 Run Time Programs

JOSHUA DPSTM-500	Title	Part A.	Precompiler	Vol.	Page
				IX	A.O.2
				Rev.	Date
				1	6/73

(Blank)

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.1
		Rev. 1	Date 6/73

Work Done By

H. C. Honeck
 J. C. Jensen
 F. D. Knight
 J. E. Suich
 R. S. Wood

Report Written By

R. S. Wood

Contents

	<u>Page</u>
1. Introduction	A.1.3
2. Source Record	A.1.5
3. Backus-Naur Symbolism	A.1.7
3.1 Basic Elements	A.1.8
3.2 FORTRAN Elements	A.1.8
3.3 JOSHUA Elements	A.1.9
4. Precompiled Statements	A.1.11
4.1 BLOCK DATA	A.1.11
4.2 CLOSE	A.1.11
4.3 COMMON	A.1.11
4.4 COMPLEX (See Explicitly Defined Variables)	A.1.11
4.5 DEFAULT DATA SET	A.1.11
4.6 DEFINE DATA SET	A.1.11
4.7 DELETE (Also see JOSHUA I/O)	A.1.12
4.8 DIMENSION	A.1.12
4.9 DOUBLE PRECISION (See Explicitly Defined Variables)	A.1.12
4.10 END	A.1.12
4.11 ENTRY	A.1.12
4.12 EXEC	A.1.12
4.13 EXECUTE	A.1.13
4.14 EXIT	A.1.13
4.15 Explicitly Defined Variables	A.1.13
4.16 FUNCTION	A.1.13
4.17 IF	A.1.14
4.18 IMPLICIT	A.1.14
4.19 INTEGER (See Explicitly Defined Variables)	A.1.14
4.20 J Statements	A.1.14
4.21 JOSHUA I/O	A.1.15
4.21.1 Invariant Qualifier Mask	A.1.15
4.21.2 Statement Generation	A.1.16

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.2
		Rev. 1	Date 6/73

Contents (Cont.)

	<u>Page</u>
4.22 JOSHUA Variables	A.1.18
4.23 LIST	A.1.19
4.24 LOGICAL (See Explicitly Defined Variables)	A.1.19
4.25 MODULE	A.1.19
4.26 NOLIST	A.1.19
4.27 OPEN	A.1.20
4.28 PRETIMING	A.1.20
4.29 PRINT	A.1.20
4.30 PROGRAM TIMINGS	A.1.21
4.31 Pseudo Qualifier (->)	A.1.23
4.32 READ (Also see JOSHUA I/O)	A.1.23
4.33 REAL (See Explicitly Defined Variables)	A.1.23
4.34 RETURN	A.1.23
4.35 STOP	A.1.23
4.36 SUBROUTINE	A.1.24
4.37 SYSTEM NAMES	A.1.24
4.38 WRITE (Also see JOSHUA I/O)	A.1.24

1. Introduction

Users of the JOSHUA Operating System write their programs in a modified FORTRAN language, i.e., to the normal syntax is added a JOSHUA syntax used to perform JOSHUA functions. The purpose of the Precompiler is to examine each FORTRAN source statement and to translate it, if necessary, into other valid FORTRAN source statement(s) which, when compiled by a normal FORTRAN compiler, will perform the desired functions under the JOSHUA Operating System.

JOSHUA

DPSTM-500

Title Language Specifications

Vol.
IX

Page
A.1.4

Rev.
1

Date
6/73

2. Source Records

Normal FORTRAN conventions apply to the form of the source statements:

- Input is assumed to be sequential on 80-column records.
- Columns 1 thru 5 of the first record of a statement can contain a label.
- Column 6 on the first record of a statement is either a blank or a zero.
- Columns 7 thru 72 contain the statement.
- Columns 73 thru 80 are not used.
- A statement can be continued on up to and including 19 additional records. Each of these must have a non-blank and non-zero in Column 6. Columns 7 thru 72 are used to continue the statement. Again, columns 73 thru 80 are not used.

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.6
		Rev. 1	Date 6/73

(Blank)

3. Backus-Naur Symbolism

The following Backus-Naur notation is used to express the various statements handled by the Precompiler:

<X> the object named X

:= ... can be formed from ...

| exclusive OR

{Z}_i^j Z is to be repeated at least i times but not
more than j times. When i is omitted its value
is assumed to be 1, and when j is omitted its
value is assumed to be infinity.

[...] optional set

JOSHUA DPSTM-500	Title Language Specifications	Vol.	Page
		IX	A.1.8
		Rev.	Date
		1	6/73

Whenever several elements appear consecutively, it is assumed they are to be concatenated to form a larger element.

3.1 Basic Elements:

- the digit set
 $\langle d \rangle := 0|1|2|3 \dots |9$
- the alpha set
 $\langle a \rangle := \$|A|B|C \dots |Z$
- the character set
 $\langle c \rangle := \text{all printable characters}$

3.2 FORTRAN Elements:

- a statement label
 $\langle l \rangle := \left\{ \langle d \rangle \right\}_1^5$
- A FORTRAN integer constant
 $\langle ic \rangle := \left\{ \langle d \rangle \right\}_1$
- A simple FORTRAN variable
 $\langle sv \rangle := \langle a \rangle \left\{ \langle a \rangle | \langle d \rangle \right\}_0^5$
- An array FORTRAN variable
 $\langle av \rangle := \langle sv \rangle \left(\langle v \rangle \left\{ , \langle v \rangle \right\}_0^6 \right)$
- Any FORTRAN variables
 $\langle v \rangle := \langle sv \rangle \langle av \rangle$
- The ERROR condition branch
 $\langle err \rangle := \text{ERR} = \langle l \rangle$
- The END condition branch
 $\langle end \rangle := \text{END} = \langle l \rangle$

3.3 JOSHUA Elements:

- The qualifier character set

$\langle qc \rangle :=$ all printable characters except the single quote (') and the period (.)

- Invariant qualifier

$\langle iq \rangle := ' \left\{ \langle qc \rangle_1^8 \right\} | \langle d \rangle \left\{ \langle a \rangle | \langle d \rangle \right\}_0^7$

- All qualifiers

$\langle q \rangle := \langle iq \rangle | \langle v \rangle$

- Qualifier record name

$\langle qr \rangle := \langle q \rangle \left\{ . \langle q \rangle \right\}_0^{11}$

- Pseudo qualifier

$\langle pq \rangle := \langle sv \rangle | \langle d \rangle \left\{ \langle a \rangle | \langle d \rangle \right\}_0^7$

- Pseudo record name

$\langle pr \rangle := \langle q \rangle | \langle pq \rangle \left\{ . \langle q \rangle | \langle pq \rangle \right\}_0^{11}$

- Colon record name

$\langle cr \rangle := \langle av \rangle : \langle av \rangle$

- Record name

$\langle r \rangle := \langle qr \rangle | \langle cr \rangle$

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.10
		Rev. 1	Date 6/73

(Blank)

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.11
		Rev. 1	Date 6/73

4. Precompiled Statements

4.1 BLOCK DATA

The BLOCK DATA statement is recognized by the Precompiler and causes all statement processing to be stopped until the next END card.

4.2 CLØSE

[<i>] CLØSE

The Precompiler changes this JOSHUA statement to a Comment and generates the following instead:

[<i>] CALL \$YUD4

4.3 CØMMØN

All CØMMØN statements are scanned for dimensioned variables which are kept in internal tables for Precompiler use.

4.4 CØMPLEX (See Explicitly Defined Variables)

4.5 DEFAULT DATA SET

[<i>] DEFAULT DATA SET

The Precompiler changes this JOSHUA statement to a comment and generates the following:

[<i>] CALL \$YUDFL (\$JCB)

See Run Time Program \$YUDFL for a description of the actions performed.

4.6 DEFINE DATA SET

[<i>] DEFINE DATA SET $\left(\langle q \rangle \left\{ , \langle q \rangle \right\}_0^3 \right)$

The Precompiler changes this JOSHUA statement to a Comment and generates the following:

[<i>] CALL \$YUDFN $\left(\$JCB \left\{ , \langle q \rangle \right\}_1^4 \right)$

See Run Time Program \$YUDFN for a description of the work performed.

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.12
		Rev. 1	Date 6/73

4.7 DELETE (Also see JOSHUA I/O)

[<l>] DELETE(<pr>|<cr>[,<err>])

The Precompiler changes this JOSHUA statement to a Comment. See JOSHUA I/O Statement Generation for a discussion of the source generation.

4.8 DIMENSION

All variables appearing in a DIMENSION statement are kept in internal tables for Precompiler use.

4.9 DOUBLE PRECISION(See Explicitly Defined Variables)

4.10 END

END statements are used by the Precompiler as the signal to:

- Clean up after the current routine; and
- Initialize for the next routine.

4.11 ENTRY

ENTRY statements are recognized and, if needed, program timings code is inserted behind them. See PROGRAM TIMINGS for a discussion of the statements generated.

4.12 EXEC

[<l>]EXEC['<sv>'|<v>]₁¹ [(FORTRAN Parameter List)][ERR=<l>]

The Precompiler changes this JOSHUA statement to a Comment and generates the following:

- No ERR = <l>

[<l>] CALL \$YUDEX(0,'<SV>'|\$RAJ(code, <v>),\$JCB
[,FORTRAN parameter list])

- With ERR = <l>

[<l>] CALL \$YUDEX(&<l>,1,'<SV>'|\$RAJ(code <v>,\$JCB
[,FORTRAN parameter list])

4.13 EXECUTE

[<l>]EXECUTE<sv>[(FORTRAN parameter list)][ERR=<l>]

The Precompiler changes this JOSHUA statement to a Comment and generates the following:

o No ERR=<l>

[<l>]CALL \$YUDEX(0, '<sv>', \$JCB[,FORTRAN parameter list])

o With ERR=<l>

[<l>]CALL \$YUDEX(&<l>, 1, '<sv>', \$JCB[,FORTRAN parameter list])

4.14 EXIT

[<l>]EXIT

The Precompiler changes this JOSHUA statement to a Comment and generates the following instead:

[<l>]CALL \$YUD4

4.15 Explicitly Defined Variables

The FORTRAN keyword statements

COMPLEX
DOUBLE PRECISION
INTEGER
LOGICAL
REAL

are scanned and two internal tables updated:

- o Explicitly defined FORTRAN variables are kept for use in the construction of the byte count expression needed for JOSHUA WRITE statements.
- o Dimensioned variables are kept in internal tables used to check the validity of the EXEC, EXECUTE, and colon type I/O statements.

4.16 FUNCTION

<div style="border-left: 1px solid black; border-right: 1px solid black; padding: 5px; display: inline-block;"> REAL INTEGER LOGICAL COMPLEX DOUBLE PRECISION </div>	FUNCTION <sv>	<div style="border-left: 1px solid black; border-right: 1px solid black; padding: 5px; display: inline-block;"> *16 *8 *4 *2 *1 </div>	(FORTRAN Parameter List)
--	------------------	--	--------------------------

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.14
		Rev. 1	Date 6/7

The Function name, i.e., <sv>, is kept by the Precompiler for use in the generation of program timings code or as an explicitly defined variable whose byte count is needed when constructing the byte count expression for JOSHUA WRITE statements.

4.17 IF

[<i>] IF (logical expression) executable statement.

Logical IF statements are checked for the following executable statements:

- STØP (See STØP keyword for JOSHUA syntax) statements are replaced by the appropriate generation appended to the Logical IF.
- RETURN statements, if program timings code is being generated, generate the Logical IF twice:
 - The first, prefixed by the label <i>, has appended the program timings coding; and,
 - The second has the original RETURN statement.

4.18 IMPLICIT

Information from a normal FORTRAN IMPLICIT statement is stored internally and is used when constructing the byte count expression for JOSHUA WRITE statements.

4.19 INTEGER (See Explicitly Defined Variables)

4.20 J Statements

Early versions of the Precompiler required a J in the first column of those source statements containing JOSHUA syntax. In the current Precompiler this usage is optional and not recommended for the general user. However, system programmers can use a J in column one to force the Precompiler to accept FORTRAN Direct Access READ or WRITE statements; normally they would be flagged as errors.

4.21 JOSHUA I/Ø

The Precompiler replaces JOSHUA oriented I/Ø with inline assignment statements, a call to a YUD routine, and if appropriate, the actual READ or WRITE statement needed. Communication with the JOSHUA System programs is accomplished by the YUD program which completes any initialization not performed inline and then calls BNAM which is in the JOSHUA System.

4.21.1 Invariant Qualifier Masks

Calls to \$YUDI routines for Qualifier Record Names have the following form:

```
CALL $YUDI($LGL$(j),$LNAM$....)
```

where the first two parameters contain enough information for the \$YUDI routines to load Invariant Names, <iq>'s, from the \$LNAM\$ array into the \$NAMES array prior to calling BNAM. The Real*8 \$LNAM\$ vector is dumped out by the Precompiler, initialized with all the separate Invariant Qualifiers found during the processing of a routine, immediately prior to the END statement. Its maximum dimension is 255.

The Logical*1 \$LGL\$ vector is also dumped out immediately prior to the END statement and it contains a series of masks for each unique Qualifier Record Name found.

- The first byte contains the hexadecimal value of the length of the mask, i.e., the number of bytes, starting in Relative byte 3, contained in the mask.
- The second byte contains the hexadecimal value of the total number of qualifiers in this Record Name. This value is loaded into \$BNAMI(4) by \$YUDI.
- The n bytes beginning in byte 3 contain the hexadecimal subscript of the position within the \$LNAM\$ array of the Invariant Qualifier. A value of zero indicates to \$YUDI that the qualifier has been loaded inline. There is a congruence between byte 3 and \$NAMES subscript 1, byte 4 and subscript 2, etc.

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.16
		Rev. 1	Date 6/

4.21.2 Statement Generation

The keywords READ, WRITE, and DELETE are used to perform JOSHUA oriented I/O. The general form of these statements is:

$[<l>]$
 $\left[\begin{array}{l} \text{READ} \\ \text{WRITE} \\ \text{DELETE} \end{array} \right] \quad (<pr> \mid <cr>[,<err>]) \quad [I/O \text{ List}]$

where:

- $<l>$ is a statement label
- $<pr>$ is a pseudo record name
- $<cr>$ is a colon record name
- $<err>$ is the error condition branch
- I/O List is in the standard FORTRAN form

In all cases the Precompiler will make the original statement a Comment and will generate substitute source code.

- The optional label parameter, $<l>$, will be placed on the first source statement generated.
- For WRITE's the first generated statement will be:

$\$BNAMI(5) = \text{byte count expression for the I/O list}$

- For colon record names, i.e., $<cr>$, the next statement generated is

$\$BNAMI(4) = 1 - (\text{Paren}_1) + (\text{Paren}_2)$

where: Paren_1 and Paren_2 are, respectively, the parenthetical expressions of the array subscripts appearing before and after the colon.

- For pseudo record names, i.e., $<pr>$, there will be a series of statements generated equal to the number of variables, $<v>$, in the qualifier record name, $<qr>$. To convert $<pr>$'s to a $<qr>$ the Precompiler scans the $<pr>$ repeatedly, substituting for each pseudo qualifier found, $<pq>$, the pseudo record it defines until there are no $<pq>$'s left.
(See section which discusses pseudo qualifiers.)

The statements generated are of two types:

- For Real*8 variables
\$NAME(i) = <v>
- For all other variable types
\$NAMES(i) = \$RAJ(code,<v>)

where i is the relative position of
the variable in the qualifier string,
and,

code = 1 for Logical*1,
 = 2 for Integer*2,
 = 3 for Logical*4,
 = 4 for Integer*4,
 = 5 for Real*4,
 = 6 for Real*8
 = 7 for Complex*8,
 = 8 for Complex*16 variables;

<v> is the variable qualifier.

- For colon record names the next generated
statement is one of the following two forms:

- If <err> was not present
CALL \$YUDj(0,<av>,code)
- If <err> was present
CALL \$YUDj(1,&label,<av>,code)

where: j = B for READ
 = C for WRITE
 = E for DELETE

label is the label from the
<err>;

code has values as defined
above and is a type code for
the variable appearing ahead
of the colon.

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.18
		Rev. 1	Date 6/77

- For a qualifier record name the next statement has one of the following two forms:

- If <err> was not present
CALL \$YUDI(\$LGL\$(k), \$LNAM, 0)
- If <err> was present CALL \$YUDI
(\$LGL\$(k), \$LNAM, 1, &label)

where: 1 = 2 for READ
= 3 for WRITE
= 5 for DELETE;

label is the label from the <err>;

k is the subscript of the beginning
in the \$LGL\$ logical *1 array of the
Invariant Qualifier Mask.

- For READ's and WRITE's the last statement generated is

READ WRITE	(\$DSNAM'\$RECRD)I/O LIST
-------------------	---------------------------

This statement will not be generated if
there was no I/Ø list.

4.22 JOSHUA Variables

Each routine precompiled has inserted immediately behind
the MODULE, SUBROUTINE or FUNCTION statement, the following:

```
EXTERNAL $CDINT,$RAJ,$FLGO6
COMMON/$JCØMA/JCB(16), $NAMES(18), $BNAMI
REAL*8$NAMES,$JCB,$CDINT,$RAJ,$LNAM$(1), $JØB,$BADGE
INTEGER*4$BNAMI(8), $DSNAM,$RECRD,$JCBI(1), $SYS,$ERRNØ
EQUIVALENCE($JCB(1), $JCBI(1)), ($BNAMI(2), $DSNAM),
*($BNAMI(3), $RECRD), ($JCBI(5), $SYS), ($BADGE, $JCB(2)),
*($ERRNØ, $JCBI(8)), ($JØB, $JCBI(1))
LOGICAL*1/$LGL$(1)/F/,$$$,$FLGO6*4
```

These are the JOSHUA mnemonics which are either used by the
Precompiler in its source statement generation or can be
expected to be used by a programmer. Note that the above
generation is deferred until after an IMPLICIT statement,
if one is present.

4.23 LIST

This statement will be changed to a Comment and it is the instruction to the Precompiler that a source listing is desired. The default, when the Precompiler is invoked, is to list the source. (See NØLIST keyword.)

4.24 LØGICAL (See Explicitly Defined Variables)

4.25 MODULE

MODULE [<sv>][(FORTRAN Parameter List)]

The Precompiler changes this JOSHUA statement to a Comment and generates one of the following four statements instead:

- Neither optional parameter present
SUBROUTINE MAIN(\$MCB)
- First optional parameter present, second missing
SUBROUTINE <sv> (\$MCB).
- First optional parameter missing, second present
SUBROUTINE MAIN(\$MCB,FØRTRAN Parameter List).
- Both optional parameters present
SUBROUTINE <sv> (\$MCB,FØRTRAN Parameter List)

Any label which appears on this statement is ignored.

After the standard JOSHUA variables have been defined the following three statements are always generated:

- REAL*8 \$MCB(18)
- \$JCB(1) = 0.000
- CALL \$LINK ('JIBINIT',\$MCB,\$JCB)

A fourth statement is optionally generated if the PROGRAM TIMINGS option is being used:

- IF (\$JCBI(13).EQ.1) CALL \$\$1

4.26 NØLIST

This statement will be changed to a Comment and is the instruction to the Precompiler that a source listing is not desired (See LIST keyword.)

JOSHUA DPSTM-500	Title Language Specifications	Vol.	Page
		IX	A.1.20
		Rev.	Date
		1	6/

4.27 ØPEN

[<I>]ØPEN

The Precompiler changes this JOSHUA statement to a Comment and generates the following instead:

[<I>] CALL \$YUD1

4.28 PRETIMING ØN|ØFF

These statements will be changed to a Comment and it is the instruction to the Precompiler that Precompiler subroutine timing is either to be performed or not performed. The default is to not perform. This statement should only be used by programmers who are modifying the Precompiler.

4.29 PRINT

[<I>] PRINT (<r>[,<err>])

The Precompiler changes this JOSHUA statement to a Comment and generates the following:

o No <err>

[<I>]CALL \$YUDEX (0, '\$YUDPR', \$JCB, value {, \$RAJ (code, <v>)}₁¹²)

o With <err>

[<I>]CALL \$YUDEX (&label, 1, '\$YUDPR', \$JCB, value {, \$RAJ (code, <v>)}₁¹²)

where:

<I> on the left is the statement label;

&label is from the error condition;

'\$YUDPR' is the name of the routine to which \$YUDEX will pass control;

\$JCB is the standard JOSHUA control block;

value is an integer indicating how many names are in the Record Name;

\$RAJ(code, <v>) is a Real*8 function which will convert the FORTRAN variable <v> into a Hollerith suitable for use as a Record Name Qualifier based on the following integer 'code'

1	Logical*1	5	Real*4
2	Integer*2	6	Real*8
3	Logical*4	7	Complex*8
4	Integer*4	8	Complex*16

4.30 PROGRAM TIMINGS

PROGRAM TIMINGS {<d>} [,ØVLY]

The Precompiler changes this statement to a Comment. It is the instruction to the Precompiler that each subsequent user routine in the input stream is to have source statements inserted which will keep track, at run time, of the net amount of CPU time each used and the number of times each was entered.

The first parameter, {<d>}, is an estimate of the number of user routines plus the number of entry points for which timing code will be generated. The second parameter instructs the Precompiler that the user module will have an overlay structure.

- Coding inserted immediately following the Standard JOSHUA Variables.

CØMMØN/\$\$TM/\$\$\$

- The next statement only appears in those routines defined by the MODULE statement. It allows the user to 'turn on' the timing function from the EXEC card of his JCL i.e., TIMING=ØN

IF (\$JCBI(13).EQ.1) CALL \$\$1

- All routines will have the following generation

IF (\$\$\$\$) CALL \$\$3(\$LGL\$, '<sv>')

where <sv> is either the MØDULE, SUBRØUTINE or FUNCTION name.

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.22
		Rev. 1	Date 6

- Coding inserted immediately following any ENTRY statement.

IF (\$\$\$) CALL \$\$\$(\$LGL\$, '<sv>')

where <sv> is the ENTRY name.

- Coding inserted immediately prior to any simple RETURN statement

[<l>] IF (\$\$\$) CALL \$\$\$(\$LGL\$)

where <l> is the label on the original RETURN statement. The RETURN statement is then generated without a label.

- Coding inserted immediately prior to a RETURN appended to a logical IF statement.

[<l>] IF (\$\$\$.(logical expression))CALL \$\$\$(\$LGL\$)

where <l> is the label on the original logical IF statement. The original logical IF statement is then generated without a label.

Printing out the accumulated statistics is accomplished indirectly by the STOP statements which call either \$YUD98 or \$YUD99 where the actual demand for printing is made. (See Run Time Programs for a discussion on other user control of the run time timings functions).

At the completion of the Precompiler run a Block Data subprogram is generated which is used by the run time timings program to accumulate statistics. The form of this subprogram is:

BLOCK DATA

COMMON/\$\$TM/\$\$\$\$(2),\$\$A(3),\$\$B(m)

LOGICAL*1 \$\$\$/F, l/

INTEGER*2/n, 2*0/, \$\$B/m*0/

END

where: n is the first parameter on the PROGRAM TIMINGS card, i.e., [<d>],

m is six times n, and

l is T if, ØVLY was present and F otherwise.

JOSHUA DPSTM-500	Title Language Specifications	Vol. IX	Page A.1.23
		Rev. 1	Date 6/73

4.31 Pseudo Qualifier

- `<pr>[,<err>] - > <pq>`
- `<err> - > <pq>`

The value of the Pseudo Qualifier is defined as the expression to the left of the - > sign. Pseudo Qualifiers are used as shorthand notation when expressing Pseudo Record Names. The Precompiler considers each non-colon type JOSHUA Record Name to be a Pseudo Record Name and converts each Pseudo Qualifier found into its value when expanding the Pseudo Record Name into a valid Qualifier Record Name.

An Error Condition Branch appearing as part of a Pseudo Qualifier value can be considered a default which will be overridden if the Error Condition Branch is coded into the actual READ/WRITE/DELETE statement.

4.32 READ (Also see JOSHUA I/O)

`[<lt>] READ (<pr>|<cr>[,<err>]) [I/Ø List]`

Normal FORTRAN READ statements are not modified by the Precompiler. However, a JOSHUA READ statement is made a Comment and the source generated for it is discussed in JOSHUA I/Ø Statement Generation.

4.33 REAL (See Explicitly Defined Variables)

4.34 RETURN

If program timings code is being generated then the appropriate line of coding, prefixed by the label `<lt>` on the RETURN, is produced immediately prior to the generation of the RETURN statement.

4.35 STØP

`[<lt>] STØP [<ic>] ['{<c>}631']`

The Precompiler changes this statement to a Comment and generates one of the following four statements instead:

- Neither optional parameter two or three is present.
`[<lt>] CALL $YUD98`

JOSHUA DPSTM-500	Title Language Specifications	Vol.	Page
		IX	A.1.24
		Rev.	Date
		1	6/

- Second optional parameter present, third missing
[<I>] CALL \$YUD99(\$LGL\$(n),\$LNAM\$,<ic>|<sv>)
- Second optional parameter missing, third present
[<I>] CALL \$YUD99(LGL\$(n),\$LNAM\$,0)
- Second and third optional parameters present
[<I>] CALL \$YUD99(\$LGL\$(n),\$LNAM\$,<ic>|<sv>)

4.36 SUBROUTINE

SUBROUTINE <sv> [(FORTRAN Parameter List)]

The Subroutine name, i.e., <sv>, is kept by the Pre-compiler for use in the generation of program timings code.

4.37 SYSTEM NAMES

More knowledgeable programmers will, at times, have a need to use JOSHUA variables which are not part of the standard generation. The above keyword source statement will be changed to a Comment and will cause the following source statements, identifying little used JOSHUA variables, to be generated:

```

1234 7
C*** SYSTEM NAMES
      INTEGER*4 $LENTH,$NAMLN,$BFLAG,$EFLAG,$MFLAG,$NDERR
      EQUIVALENCE ($BNAMI(5),$LENTH),($BNAMI(4),$NAMLN),
      *($JCBI(7),$BFLAG),($JCBI(6),$EFLAG),
      *($JCBI(10),$MFLAG),($JCBI(9),$NDERR)

```

4.38 WRITE (Also see JOSHUA I/Ø)

[<I>] WRITE (<pr>|<cr> [,<err>]) [I/Ø List]

Normal FORTRAN WRITE statements are not modified by the Precompiler. However, a JOSHUA WRITE statement is made a Comment and the source generated for it is discussed in JOSHUA I/Ø Statement Generation.

JOSHUA

DPSTM-500

Title Precompiler Module

Vol. IX	Page A.2.1
Rev. 1	Date 6/73

Work Done By

A. H. Dobyns
E. C. Olson
J. E. Suich
R. S. Wood

Report Written By

R. S. Wood

Contents

	<u>Page</u>
1. Main Program	A.2.3
2. Subprograms	
2.01 \$AN\$	A.2.5
2.02 \$E\$	A.2.7
2.03 \$IMPLY	A.2.9
\$DFLT\$	
\$MPLCT	
2.04 \$T\$	A.2.11
\$C\$	
\$D\$	
\$I\$	
2.05 \$XPLCT	A.2.13
CØMØN	
DIMEN	
2.06 \$YUP	A.2.15
2.07 CHARCHEK	A.2.17
CHAR	
RHAR	
2.08 CLEAR	A.2.19
2.09 DEFLTS	A.2.21
2.10 DEFSET	A.2.23
2.11 ERRMSG	A.2.25
ENDMSG	
2.12 EXECUT	A.2.27
2.13 GSTRIN	A.2.29
PSTRIN	
QSTRIN	
2.14 IF	A.2.31
2.15 JASIGN	A.2.33
2.16 JMØDUL	A.2.35

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.2
		Rev. 1	Date 6/7

Contents

	<u>Page</u>
2. Subprograms (Cont.)	
2.17 JØCE	A.2.37
2.18 JPRINT	A.2.39
2.19 JREAD	A.2.41
2.20 JSTØP	A.2.43
2.21 LIMIT	A.2.45
LINDEX	
2.22 LISTLN	A.2.47
2.23 MØVEIT	A.2.49
2.24 MULTFC	A.2.51
2.25 NEWTAB	A.2.53
NWTAB1	
2.26 NSTRIN	A.2.55
2.27 ØUTPUT	A.2.57
2.28 RETURN	A.2.59
2.29 SUBFUN	A.2.61
2.30 TIMCD	A.2.63
TIMBD	
2.31 UNPACX	A.2.65
2.32 ZCØMP	A.2.67
2.33 CARDIN	A.2.69

JOSHUA DPSTM-500	Title Precompiler Main Program	Vol. IX	Page A.2.3
		Rev. 1	Date 6/73

Purpose: Read in JOSHUA Fortran source programs and convert the JOSHUA syntax to standard Fortran which can be handled by a normal compiler.

Main Program Name: PRECOM2

Common: \$\$TM, CHARST, IOUNIT, NEWPGE, NTAB, PACKED, RAW, SYSNAM, TIMING, UNPACK

Subprogram Used: \$\$1, \$\$2, \$\$3, \$\$4, \$\$5, \$DFLT\$, \$I\$, \$MPLCT, \$XPLCT, CHAR, COMON, DEFLT\$, DEFSET, DIMEN, ERRMSG, EXECUT, IF, JASIGN, JMODUL, JOCE, JPRINT, JREAD, JSTOP, MOVEIT, OUTPUT, PSTRI, QSTRIN, RETURN, RHAR, SUBFUN, TIMBD, TIMCD, ZCOMP, CARDIN.

Data Set: Logical unit 5 to read source program cards.

Language: OS 360 FORTRAN IV

Authors: R. S. Wood, A. H. Dobyns

JOSHUA

DPSTM-500

Title Precompiler Module
 Subroutine: \$AN\$

Vol. IX	Page A.2.5
Rev. 1	Date 6/73

Purpose: Define the limits of an alphameric field.

Usage: CALL \$AN\$(Z, I, J, M, K)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
Z	L*1	Contains a packed, i.e., no embedded blanks, FORTRAN source statement.
I	I*4	Pointer to the first byte within Z of the mnemonic to be defined.
J	I*4	Pointer, set by \$AN\$, to the last byte of the mnemonic.
M	I*4	Total length of the inputted source statement.
K	I*4	Return code: 0 No mnemonic found; 1 FORTRAN type mnemonic, i.e., first character is alpha; 2 Non-FORTRAN type, first character is numeric but some alpha found; 3 Pure numeric.

Common: \$\$TM,CHARST

Subprograms Used: \$\$3, \$\$4, RHAR

Language: OS/360 FORTRAN IV

Author: R. S. Wood

Remarks: Beginning at Z(I) the packed FORTRAN source statement is scanned until a non-alphameric is found or the end of the string is encountered. The return code, K, is set and the right-hand limit of the field is placed in J.

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.6
		Rev. 1	Date 6/73

(Blank)

JOSHUA DPSTM-500	Title Precompiler Module Subroutine: \$E\$	Vol. IZ	Page A.2.7
		Rev. 1	Date 6/73

Purpose: Update the Precompiler lists with the standard JOSHUA mnemonics and data types.

Usage: CALL \$E\$

Common: \$\$TM, SYSNAM

Subprograms Used: \$\$3, \$\$4, \$T\$

Language: OS/360 FORTRAN IV

Author: E. C. Olson

Remarks: The SYSNAM named Common area is initialized using a BLOCK DATA program with all the standard JOSHUA mnemonics and their type codes. Prior to processing the first I/O source statement \$E\$ is called and it adds the mnemonics and type codes to the lists kept by the \$T\$ routine.

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.8
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Function: \$IMPLY

Vol. IX	Page A.2.9
Rev. 1	Date 6/73

Purpose: Return the default type code for a FORTRAN mnemonic.

Entry: \$DFLT\$ Establish standard FORTRAN mnemonic type codes.

\$MPLCT Modify the FORTRAN mnemonic type codes by the information found on an IMPLICIT statement.

Usage: I=\$IMPLY(A)
CALL \$DFLT\$ (dummy)
CALL \$MPLCT (dummy)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
I	I*4	Returned value of FORTRAN type code. In use are: 0 not a FORTRAN mnemonic 1 Logical*1 2 Integer*2 3 Logical*4 4 Integer*4 5 Real*4 6 Real*8 7 Complex*8 8 Complex*16
A		Contains mnemonic. Only first byte is checked.
Dummy	Dummy variable.	

Common: \$\$TM, CHARST, NEWPGE, PACKED

Subprograms Used: \$\$3, \$\$4, CHAR, OUTPUT, RHAR, ZCØMP

Language: OS/360 FORTRAN IV

Author: R. S. Wood

JOSHUA DPSTM-500	Title Precompiler Module Function: \$IMPLY	Vol. IX	Page A.2.10
		Rev. 1	Date 6/75

Remarks:

\$IMPLY is an integer function used by the pre-compiler to determine the byte lengths, i.e., through use of the returned code, of FORTRAN mnemonics found in source statements. This function is used for all mnemonics not explicitly typed.

The entry \$DFLT\$ is used to initialize internal tables before each subprogram is precompiled and the entry \$MPLCT is used to modify the internal tables by the IMPLICIT statement.

Purpose: Maintain lists of explicitly defined variables and return the type code for any FORTRAN mnemonic.

Entry: \$C\$ Maintain lists of pseudo qualifiers and return the subscript of where the name strings are stored.

\$D\$ Maintain lists of dimensioned variables and return a non-zero value if a variable has been dimensioned.

\$I\$ Initialize the internal arrays and counters for list maintenance.

Usage:

```
I = $T$(AN, CODE)
J = $C$ (AN, CODE)
J = $D$ (AN, CODE)
CALL $I$(K)
```

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
I	I*4	Returned value of the FORTRAN type code. In use are: <ul style="list-style-type: none"> 0 not a FORTRAN mnemonic 1 Logical*1 2 Integer*2 3 Logical*4 4 Integer*4 5 Real*4 6 Real*8 7 Complex*8 8 Complex*16
J	I*4	Returned value stored with the pseudo qualifier or dimensioned variable when it was first entered into the table. (See CODE).
K		Dummy variable.
AN	R*8	Contains the alphameric field.

JOSHUA DPSTM-500	Title Precompiler Module Function: \$T\$	Vol. IX	Page A.2.12
		Rev. 1	Date 6/7

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CØDE	I*4	If equal to 0 the program searches its lists for the alphameric field. If not equal to 0 the program stores the name in its lists along with the CØDE. On subsequent searches for the alphameric field this CØDE is returned as the function value.

Common: \$\$TM, \$NMEN\$

Subprograms Used: \$\$3, \$\$4, \$IMPLY, CLEAR, MOVEIT, ZCØMP

Language: OS/360 FORTRAN IV

Author: R. S. Wood

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: \$XPLCT

Vol. IX	Page A.2.13
Rev. 1	Date 6/73

Purpose: Strip all explicitly defined and dimensioned variables from REAL, INTEGER, LOGICAL, and COMPLEX keyword source statements.

Entry: COMMON Strip dimensioned variables from COMMON statements.

DIMEN Strip dimensioned variables from DIMENSION statements.

Usage: CALL \$XPLCT (&label)
CALL COMMON (&label)
CALL DIMEN (&label)

Arguments:

<u>Variable</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label.	RETURN 1 is always taken from this routine.

Common: \$\$TM, CHARST, PACKED, UNPACK

Subprograms Used: \$\$3, \$\$4, \$AN\$, \$D\$, \$T\$, ERRMSG, MOVEIT, OUTPUT, PSTRIN, QSTRIN, RHAR, ZCOMP

Language: OS/360 FORTRAN IV

Authors: R. S. Wood, E. C. Olson

Remarks: As variable names are stripped from the typing statement they are fed to the \$T\$ routine with the appropriate type code. If the variable is dimensioned then its name is also fed to the \$D\$ routine which maintains this list.

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.14
		Rev. 1	Date 6/7

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: \$YUP

Vol. IX	Page A.2.15
Rev. 1	Date 6/73

Purpose: Construct the Invariant Qualifier Masks (See Section 3.) used in CALL's to the YUDi routines.

Usage: CALL \$YUP(S, TN, A, AN)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
S	R*8	Contains the mask to be added to the \$LGL\$ array in the user's program (see Section 3.)
TN	I*4	Contains, on input, the number of qualifiers in the record name. On output, it contains the \$LGL\$ index to be used when constructing the CALL \$YUDi for the user.
A	L*1	Array where all discrete masks are kept by the Precompiler.
AN	I*4	Number of bytes currently used in the A array.

Common: \$\$TM

Subprograms Used: \$\$3, \$\$4, MOVEIT, RHAR, ZCØMP

Language: OS/360 FORTRAN IV

Author: R. S. Wood

JOSHUA
DPSTM-500

Title Precompiler Module

Vol.
IX

Page
A.2.16

Rev.
1

Date
6/73

(Blank)

JOSHUA DPSTM-500	Title Precompiler Module Subroutine: CHARCHEK	Vol. IX	Page A.2.18
		Rev. 1	Date 6/73

Remarks (Cont.):

string '1234567890'. All ten characters in the literal string will participate in the test because the length argument was specified as being equal to 10. If A(16) matches any one of the 10 bytes in the literal string, control is transferred to statement number 2 and I is set to the relative position within the string where the equality occurred. If it is desired to know only that the equality does exist but not which character it matched, the last argument need not be supplied. If no match is found, control is returned to the next statement below the CALL statement. The optional last argument is undefined if no match is found. The FORTRAN statement number which appears as the fourth argument in the CALL statement must have the ampersand (&) coded in front of the number to ensure proper compilation of the CALL statement. The length argument, L, is checked to ensure the $L > 1$. If $L < 1$ the subroutine returns as if no match was found.

CHAR has an entry point of RHAR and is called using either of the two forms shown under "Usage". RHAR reverses the nature of the branching in the following manner. If the first argument is NOT equal to ANY of the L characters in the second argument, a branch IS taken to the statement number shown as the fourth argument. If the branch has been taken and if the fifth argument has been coded in the call statement, the value of the fifth argument is UNDEFINED. If, when calling RHAR, an equality is found, the statement receiving control is the statement below the call statement. In this case the fifth variable is defined (if supplied as an argument). The main difference between the use of CHAR and RHAR is in the branching of control.

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: CLEAR

Vol. IX	Page A.2.19
Rev. 1	Date 6/73

Purpose: Clear a block of core to binary zeroes.

Usage: CALL CLEAR(A,N)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
A	*4 or greater	Beginning location to be cleared.
N	I*4	The number of full words to be cleared.

Language: OS/360 Assembler

Authors: R. S. Wood, A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.20
		Rev. 1	Date 6/7

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: DEFLTS

Vol. IX	Page A.2.21
Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statement to be substituted for the DEFAULT DATASETS keyword.

Usage: CALL DEFLTS(&label)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label.	RETURN 1 is always taken from this routine.

Common: \$\$TM, PACKED, UNPACK

Subprogram Used: \$\$3, \$\$4, MOVEIT, ØUTPUT, PSTRIN

Language: OS/360 FORTRAN IV

Author: E. C. Olson

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.22
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: DEFSET

Vol. IX	Page A.2.23
Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statement to be substituted for the DEFINE DATASETS keyword.

Usage: CALL DEFSET(&label)

Argument: .

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label.	RETURN 1 is always taken from this routine.

Common: \$\$TM, CHARST, IK, NTAB, PACKED, UNPACK

Subroutine Used: \$\$3, \$\$4, \$C\$, \$CDINT, \$T\$, ERRMSG, MOVEIT,
NWTAB1, OUTPUT, PSTRIN, RHAR

Language: OS/360 FORTRAN IV

Author: E. C. Olson

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.24
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: ERRMSG

Vol. IX	Page A.2.25
Rev. 1	Date 6/73

Purpose: Print all Precompiler error messages.

Entry: ENDERR Print a final message before the END statement.

Usage: CALL ERRMSG(ERRCDE)
CALL ENDERR

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
ERRCDE	I*4	The ERRCDE is used as a subscript to locate the error message which is kept in a local array.

Common: \$\$TM, IØUNIT

Subroutine Used: \$\$3, \$\$4

Data Set: Logical unit 6 to output error messages.

Language: OS/360 FORTRAN IV

Authors: R. S. Wood and A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.26
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Precompiler Module
Subroutine: EXECUT

Vol. IX	Page A.2.27
Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statement to be substituted for the EXEC or EXECUTE keywords.

Usage: CALL EXECUT(&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label.	RETURN 1 is always taken from this routine.

Common: \$\$TM, PACKED, PT, UNPACK

Subroutine Used: \$\$3, \$\$4, \$AN\$, \$CDINT, \$D\$, \$T\$, ERRMSG,
MOVEIT, OUTPUT, PSTRIIN

Language: OS/360 FORTRAN IV

Authors: E. C. Olson, A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.28
		Rev. 1	Date 6/7

(Blank)

Purpose: Pack blanks out of FORTRAN source statements.

Entry: QSTRIN Same purpose as above.
PSTRIN Same purpose as above.

Usage: CALL GSTRIN(IN,LIN,ØUT,LØUT)
(IN,LIN,ØUT,LØUT,EQ)
(IN,LIN,ØUT,LØUT,EQ,LP,RP)
(IN,LIN,ØUT,LØUT,EQ,LP,RP,LQ,RQ)
(IN,LIN,ØUT,LØUT,EQ,LP,RP,LQ,RQ,LS,RS)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
IN	Any	Contains the Fortran statement to be packed.
LIN	I*4	Length, in bytes, of the input statement.
ØUT	Any	Contains the packed statement.
LØUT	I*4	Length, in bytes, of output statement
EQ	I*4	Relative position in the packed output statement of the last equal sign found while the parentheses pairs were balanced, the quote mark pairs were balanced, and that was not part of a Hollerith specification. <u>Optional, defaults to zero.</u>
LP,RP	I*4	Relative position in packed output string of first matching pair of parentheses found. <u>Optional, defaults to zero.</u>
IQ,RQ	I*4	Relative position in the packed output string of the first matching pair of quote marks found. If a single quote is found, i.e., in direct access READ or WRITE, its relative position will appear in IQ and RQ will be set to zero. <u>Optional, defaults to zero.</u>

JOSHUA DPSTM-500	Title Precompiler Module Subroutine: GSTRIN	Vol. IX	Page A.2.30
		Rev. 1	Date 6/7

Argument (Cont.):

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
LS,RS	I*4	Relative position in the packed output string of the first matching pair of shashes found while the parentheses pairs were balanced, the quote mark pairs were balanced, and which were not part of a Hollerith specification.

Optional, defaults to zero.

Common: \$\$TM

Subroutine Used: \$\$3, \$\$4

Language: OS/360 Assembler

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

JOSHUA

DPSTM-500

Title: Precompiler Module
Subroutine: IF

Vol. IX	Page A.2.31
Rev. 1	Date 6/73

Purpose: Generate the appropriate source code for those JOSHUA keywords which appear on a logical IF statement.

Usage: CALL IF (&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label	RETURN 1 is always taken from this routine.

Common: \$\$TM, CHARST, PACKED, UNPACK

Subprogram Used: \$\$3, \$\$4, DEFLT5, DEFSET, EERMSG, EXECUT, JPRINT, JSTØP, MØVEIT, ØUTPUT, PSTRI, RETURN, ZCØMP

Language: OS/360 FORTRAN IV

Authors: E. C. Olson, R. S. Wood, A. H. Dobyns

Remarks: The current list of keywords which can be appended to a logical IF statement are:

DEFAULT DATASETS
DEFINE DATASETS
EXEC
EXECUTE
PRINT
RETURN
STØP

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.32
		Rev. 1	Date 6/

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: JASIGN

Vol. IX	Page A.2.33
Rev. 1	Date 6/73

Purpose: Check each source statement for a pseudo variable definition and save the definition if true.

Usage: CALL JASIGN(&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label	RETURN 0 is taken if the source statement is not a pseudo variable definition and RETURN 1 is taken if it is.

Common: \$\$TM,\$NMEM\$,CHARST,PACKED

Subroutine Used: \$\$3,\$\$4,\$C\$,MØVEIT,ØUTPUT,RHAR

Language: OS/360 FORTRAN IV

Authors: R. S. Wood, A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.34
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: JMØDUL

Vol. IX	Page A.2.35
Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statements to be substituted for the MØDULE keyword.

Usage: CALL JMODUL(&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label	RETURN 1 is always taken from this routine.

Common: \$\$TM,NEWPGE,PACKED,UNPACK

Subroutine Used: \$\$3,\$\$4,ERRMSG,MØVEIT,ØUTPUT

Language: OS/360 FORTRAN IV

Author: R. S. Wood, A. H. Dobyns

JOSHUA
DPSTM-500

Title Precompiler Module

Vol.
IX

Page
A.2.36

Rev.
1

Date
6/73

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: JØCE

Vol. IX	Page A.2.37
Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statements to be substituted for the ØPEN, CLØSE, and EXIT keywords.

Usage: CALL JØCE (&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label	RETURN 1 is always taken from this routine.

Common: \$\$TM,CHARST,NTAB,PACKED,UNPACK

Subroutine Used: \$\$3,\$\$4,MØVEIT,ØUTPUT

Language: OS/360 FORTRAN

Author: R. S. Wood, A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.38
		Rev. 1	Date 6/

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: JPRINT

Vol. IX	Page A.2.39
Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statements to be substituted for the PRINT keyword.

Usage: CALL JPRINT (&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN state- ment label	RETURN 1 is always taken from this routine.

Common: \$\$TM,\$NMEM\$,CHARST,IK,NTAB,PACKED,SYSNAM,UNPACK

Subroutine Used: \$\$3,\$\$4,\$AN\$,\$C\$,\$CDINT,\$E\$,\$T\$,CHAR,ERRMSG,EXECUT,
MOVEIT,NSTRIN,NWTABI,OUTPUT,PSTRIN

Language: OS/360 FORTRAN IV

Author: E. C. Olson

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.40
		Rev. 1	Date 6/

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: JREAD

Vol. IX	Page A.2.41
Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statements to be substituted for the READ, WRITE, and DELETE keywords.

Usage: CALL JREAD (&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label	RETURN 1 is always taken from this routine.

Common: \$\$TM,\$NMEM\$,CHARST,NTAB,PACKED,SYSNAM,UNPACK

Subroutine Used: \$\$3,\$\$4,\$AN\$,\$C\$,\$CDINT,\$E\$,\$T\$,\$YUP,CHAR,LISTLN,MOVEIT,NSTRIN,NWTABI,OUTPUT,PSTRIN,ZCOMP

Language: OS/360 FORTRAN IV

Authors: R. S. Wood, A. H. Dobyms

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.42
		Rev. 1	Date 6/

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: JSTØP

Vol. IX	Page A.2.43
Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statements to be substituted for the STØP keyword.

Usage: CALL JSTØP (&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN state- ment label	RETURN 1 is always taken from this routine.

Common: \$\$TM,CHARST,NTAB,PACKED,UNPACK

Subroutine Used: \$\$3,\$\$4,\$CDINT,\$YUP,CHAR,MØVEIT,NEWTAB,ØUTPUT

Language: OS/360 FORTRAN IV

Authors: R. S. Wood, A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.44
		Rev. 1	Date 6/

(Blank)

Purpose: Break out the loop limit mnemonics for either implied or normal DØ statements.

Entry: LINDEX Breakout the implied loop mnemonic.

Usage: CALL LIMIT(PCD,EQ,MAX,NMEM,&label)
CALL LINDEX(PCD,EQ,MIN,NMEM,&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
PCD	L*1	Contains the packed i.e. nonembedded blanks FORTRAN source statement.
EQ	I*4	Pointers to the byte within PCD that contains the loop equal (=) sign.
MAX	I*4	Length of the packed source statement.
MIN	I*4	Minimum pointer within PCD allowed.
NMEM	R*8	Vector of three (3) for LINDEX to hold the mnemonics found. The implied loop increment defaults to '1' if none was present.
&label	FORTLAN	RETURN 1 is taken if an error was statement detected. label.

Common: \$\$TM,CHARST

Subroutine Used: \$\$3, \$\$4, \$AN\$, CHAR, MOVEIT, RHAR

Language: OS/360 FORTRAN IV

Author: R. S. Wood

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.46
		Rev. 1	Date 6/7

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: LISTLN

Vol. IX	Page A.2.47
Rev. 1	Date 6/73

Purpose: Construct the I/O list byte count expression for JOSHUA WRITE's.

Usage: CALL LISTLN (I,&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
I	I*4	Pointer to the first character of the I/O list which has been packed and is in the PCD L*1 array in PACKED Common.
&label	FORTTRAN statement label	RETURN 1 on an error.

Common: \$\$TM,CHARST,LIST,PACKED,SYSNAM,UNPACK

Subroutine Used: \$\$3,\$\$4,\$AN\$,\$CDINT,\$E\$,\$T\$,CHAR,CLEAR,ERRMSG,LIMIT,LINDEX,MØVEIT,MULTFC,ØUTPUT,PSTRIN,RHAR,ZCØMP

Language: OS/360 FORTRAN IV

Author: E. C. Olson

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.48
		Rev. 1	Date 6

(Blank)

Purpose: Move variable length fields.

Usage: CALL MØVEIT(TØ,I,FRØM,J,L)
value=MØVEIT(TØ,I,FRØM,J,L)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
TØ		Field receiving the byte string (may be any type and/or length)
I	I*4	Relative byte within receiving field
FRØM		Field sending the byte string (may be any type and/or length)
J	I*4	Relative byte within sending field
L	I*4	Number of bytes to be moved $L < 0$. If $L \leq 0$ no move is performed.
value	I*4	Number of bytes moved.

Language: OS/360 Assembler

Authors: R. S. Wood, R. L. Boyce

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.50
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: MULTFC

Vol.	IX	Page	A.2.51
Rev.	1	Date	6/73

Purpose: Construct the implied loop multipliers needed for the LISTLN routine.

Usage: CALL MULTFC(LEVEL,NN)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
LEVEL	I*4	Current level of implied loop nesting.
NN	I*4	Pointer to the next available byte in the construction area.

Common: \$\$TM,LIST,UNPACK

Subroutine Used: \$\$3,\$\$4,MØVEIT,PSTRIN

Language: OS/360 FØRTRAN IV

Author: E. C. Olson

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.52
		Rev. 1	Date 6/7

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: NEWTAB

Vol. IX	Page A.2.53
Rev. 1	Date 6/73

Purpose: Maintain the table of invariant qualifiers.

Entry: NWTAB1 Check the invariant qualifier table for a name.

Usage: CALL NEWTAB (I,J)
CALL NWTAB1 (NAME)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
I	I*4	Pointer to the beginning byte of the name.
J	I*4	Pointer to the ending byte of the name.
NAME	R*8	Contains the name to be checked and added if not present.

Common: \$\$TM,CHARST,IK,NTAB,PACKED

Subroutine Used: \$\$%, \$\$4,EERMSG,MØVEIT

Language: OS/360 FORTRAN IV

Authors: R. S. Wood, A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.54
		Rev. 1	Date 6/7

(Blank)

Purpose: Break out the record name qualifiers from a qualifier namestring.

Usage: CALL NSTRIN (IN,LP,RP,ØUT,CLASS,N)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
IN	L*1	Packed source statement.
LP	I*4	Left limit in IN of the record name.
RP	I*4	Right limit in IN of the record name.
ØUT	R*8	Vector of qualified mnemonics. Position 17 is used for the label in END= and position 18 is used for the label in ERR= .
CLASS	I*4	Coded classification of the qualifiers. -1 Subscripted FORTRAN mnemonic. 1 FORTRAN mnemonic. 2 Non-FORTRAN type i.e. began with a digit but some alpha embedded 3 Pure integer 4 Literal constant
N	I*4	Number of qualifiers in name.

Common: \$\$TM,\$NMEM\$,CHARST

Subroutine Used: \$\$3,\$\$4,\$AN\$,CHAR,ERRMSG,MØVEIT,PSTRIN,RHAR,ZCØMP

Language: OS/360 FORTRAN IV

Author: R. S. Wood

JOSHUA DPSTM-500	Title Precompiler Module Subroutine: NSTRIN	Vol. IX	Page A.2.56
		Rev. 1	Date 6/7

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: OUTPUT

Vol. IX	Page A.2.57
Rev. 1	Date 6/73

Purpose: Generate all Precompiler output except error messages.

Usage: CALL ØOUTPUT (KEY)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
KEY	I*4	KEY is used in a computed GØTØ to generate the appropriate output.

Common: \$\$TM,CHARST,IØUNIT,NEWPGE,NTAB,PACKED,TIMING,UNPACK

Subroutine Used: \$\$3,\$\$4,\$CDINT,ENDERR,ERRMSG,MØVEIT,PSTRIN,UNPACX

Data Sets: Logical unit 6 to output the source input.
Logical unit 10 to output the modified source input.

Language: OS/360 FORTRAN IV

Author: R. S. Wood, A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.58
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: RETURN

Vol. IX	Page A.2.59
Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statements required by program timing prior to the RETURN statement.

Usage: CALL RETURN (&label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
&label	FORTTRAN statement label	RETURN 1 is always taken by this routine.

Common: \$\$TM, CHARST, PACKED, TIMING, UNPACK

Subroutine Used: \$\$3, \$\$4, MOVEIT, OUTPUT

Language: OS/360 FORTRAN IV

Authors: R. S. Wood, A. H. Dobyns

JOSHUA

DPSTM-500

Title Precompiler Module

Vol.
IX

Page
A.2.60

Rev.
1

Date
6/7

(Blank)

JOSHUA

DPSTM-500

Title	Precompiler Module Subroutine: SUBFUN	Vol. IX	Page A.2.61
		Rev. 1	Date 6/73

Purpose: Generate the FORTRAN source statements needed by JOSHUA and program timings to follow the SUBROUTINE, FUNCTION, and ENTRY statements.

Usage: CALL SUBFUN

Common: \$\$TM, PACKED, UNPACK

Subroutine Used: \$\$\$3, \$\$\$4, \$AN\$, ERRMSG, MOVEIT, OUTPUT

Language: OS/360 FORTRAN IV

Authors: R. S. Wood, A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.62
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
Subroutine: TIMCD

Vol. IX	Page A.2.63
Rev. 1	Date 6/73

Purpose: Initialize the Precompiler based on the PRØGRAM
 TIMINGS keyword.

Entry: TIMBD Generate the BLØCK DATA subprogram needed
 for program timings at run time.

Usage: CALL TIMBD
 CALL TIMCD

Common: \$\$TM,CHARST,PACKED,TIMING,UNPACK

Subroutine Used: \$\$3,\$\$4,\$AN\$,\$CDINT,ERRMSG,MØVEIT,ØUTPUT,RHAR,
 ZCØMP

Language: OS/360 FORTRAN IV

Author: R. S. Wood

JOSHUA
DPSTM-500

Title Precompiler Module

Vol.
IX

Page
A.2.64

Rev.
1

Date
6/73

(Blank)

JOSHUA

DPSTM-500

Title	Precompiler Module Subroutine: UNPACK	Vol. IX	Page A.2.65
		Rev. 1	Date 6/73

Purpose: Unpack a byte field into its hexadecimal value
for printing.

Usage: CALL UNPACK (L,I,M,J,N)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
L	L*1	Field in which hexadecimal constant will be constructed.
I	I*4	Byte within L at which construction is to begin.
M	any	Field to be unpacked.
J	I*4	Byte within M at which unpacking is to begin.
N	I*4	Number of bytes to be unpacked.

Language: OS/360 FORTRAN IV

Author: A. H. Dobyns

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.66
		Rev. 1	Date 6/

(Blank)

Purpose: To compare logically two byte string.

Usage: CALL ZCØMP (A,B,L,I)
CALL ZCØMP (A,B,L,I,N)
CALL ZCØMP (A,B,L,I,N,M)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
A		The first parameter to be compared (may be of any type and/or length).
B		The second parameter to be compared (may be of any type and/or length)
L	I*4	The length in bytes of the comparison to be made ($1 \leq L \leq 256$)
I	I*4	The return argument which contains 1 for $A < B$ 2 for $A = B$ 3 for $A > B$
N	I*4	(Optional argument) points to the first byte of parameter A to be compared. If the <u>first</u> form of the call statement is used, N will automatically be set equal to +1.
M	I*4	(Optional argument) points to the first byte of parameter B to be compared. If the <u>first</u> form of the call statement is used, M will automatically be set equal to +1. If the <u>second</u> form of the call statement is used, M will automatically be set equal to N.

Language: OS/360 Assembler

Authors: R. S. Wood, L. F. Zimmerman

JOSHUA DPSTM-500	Title Precompiler Module	Vol. IX	Page A.2.68
		Rev. 1	Date 6/

(Blank)

JOSHUA

DPSTM-500

Title Precompiler Module
 Subroutine: CARDIN

Vol.
IX

Page
A.2.69

Rev.
1

Date
6/73

Purpose: To handle the reading of all Precompiler input.

Usage: CALL CARDIN (C80, &label)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
C80	D	Card image.
&label	FORTTRAN statement label	RETURN 1 is taken if an end of data is reached; otherwise RETURN 0 is taken.

Language: Assembler

Author: W. H. Brotherton, Jr.

JOSHUA
DPSTM-500

Title Precompiler Module

Vol. IX	Page A.2.70
Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Run Time Programs

Vol.
IX

Page
A.3.1

Rev.
1

Date
6/73

Work Done By

R. S. Wood
R. C. Haywood

Report Written By

R. S. Wood
R. C. Haywood

Contents

	<u>Page</u>
1. JOSHUA I/Ø	
1.1 \$YUDI Routine (i = 1, 2, 3, 4, 5, B, C, E)	A.3.3
1.2 \$YUDFL	A.3.7
1.3 \$YUDFN	A.3.9
1.4 \$YUDPR	A.3.11
1.5 \$RAJ	A.3.13
1.6 \$CDINT or HCDINT	A.3.15
2. JOSHUA Paths	
2.1 \$YUDEX	A.3.17
2.2 \$FLGØ6	A.3.19
3. JOSHUA STØP's	
3.1 \$YUDj Routines (j = 98, 99)	A.3.21
4. Timing Subprograms	
4.1 \$\$1, \$\$2, \$\$3, \$\$4, \$\$5	A.3.23
4.2 CPUT	A.3.27

JOSHUA DPSTM-500	Title Run Time Programs	Vol. IX	Page A.3.2
		Rev. 1	Date 6/75

(Blank)

Subprograms Used: \$BERR, BNAM, \$RAJ, ERRTRA

Program Logic: JOSHUA I/O statements of OPEN, CLOSE, EXIT, READ, WRITE, and DELETE are replaced by the Precompiler with appropriate calls to \$YUDI which are really entry points in the \$YUD routine.

Table 1

Entry Point	JOSHUA Key Word	\$BNAMI(1)	Remarks
1	OPEN	1	
2	READ	2	Qualifier Records
B	READ	2	Colon Records
3	WRITE	3	Qualifier Records
C	WRITE	3	Colon Records
4	CLOSE	4	
4	EXIT	4	
5	DELETE	5	Qualifier Records
E	DELETE	5	Colon Records

The \$YUD routine completes the setup (some is done in-line) of the \$JCB, \$BNAMI, and \$NAMES vectors which are in the \$JCØMA common area and issues a CALL BNAM(\$JCB, \$BNAMI, \$NAMES, DUMMY, DUMMY, DUMMY) to accomplish the desired I/O action.

Some of the setup for the call to BNAM is:

- \$BNAMI(1) is set by \$YUD to the values listed in Table 1.
- \$BNAMI(4) for Qualifier Record Names is set by \$YUD to the value of the second byte in the Invariant Qualifier Mask.
- \$BNAMI(5) is set in-line.

Purpose:

Setup for the \$JCB, \$BNAMI, and \$NAMES vectors with user information for a call to BNAM.

Entry:

See Table 1 under Program Logic.

Usage:

- Non-record name I/O functions of OPEN, CLOSE, and EXIT (i = 1, 4, 4):
CALL \$YUDI
- Record name I/O functions of READ, WRITE, and DELETE:
 - Qualifier Record Names (i = 2, 3, 5):
CALL YUDI(\$LGL\$(l), \$LNAM, 0)
CALL YUDI(\$LGL\$(l), \$LNAM, 1, &label)
 - Colon Record Names (i = B, C, E)
CALL YUDI(0, <av>, code)
CALL YUDI(1, &label, <av>, code)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
\$LGL\$(l)	L*1	Invariant Qualifier Mask.
\$LNAM	R*8	Vector containing Invariant Qualifiers.
&label	numeric	Label from the <err> condition.
<av>	any	Array variable used in Colon I/O statements.
code	I*4	Code indicating the FORTRAN type of <av> .
	Logical*1	1
	Integer*2	2
	Logical*4	3
	Integer*4	4
	Real*4	5
	Real*8	6
	Complex*8	7
	Complex*16	8

Common:

\$JCOMA

- The \$NAMES array for
 - Qualifier Record Names has the Invariant Qualifiers added by \$YUD while the Variable Qualifiers were loaded in-line; and for
 - Colon Records Names is completely initialized by \$YUD.

On return from BNAM, \$JCBI(8) is checked to see if an error occurred. If

- No error, then a normal return is taken; and,
- With error, then either
 - CALL ERRTRA followed by a CALL \$BERR (\$JCB,\$BNAMI,\$NAMES) is taken if no error condition branch appeared in the I/O statement or,
 - RETURN 1 is taken if there was an error condition branch in the I/O statement.

Language: OS/360 Assembler.

Author: R. S. Wood

JOSHUA

DPSTM-500

Title Subroutine: \$YUDFL

Vol.
IX

Page
A.3.7

Rev.

Date

1

6/73

Purpose: Default to standard Record Name Qualifiers in the \$JCB vector. (Names are used by BNAM when searching for data).

Usage: CALL \$YUDFL(\$JCB)

Program Logic: The following work is accomplished:

\$JCB(10) = \$JCB(1)
\$JCB(11) = \$JCB(2)
\$JCB(12) = 'STDbbbbbb'
\$JCB(13) = blanks

Language: OS/360 Assembler.

Author: R. S. Wood

JOSHUA DPSTM-500	Title Subroutine: \$YUDFL	Vol. IX	Page A.3.8
		Rev. 1	Date 6/7

(Blank)

JOSHUA

DPSTM-500

Title Subroutine: \$YUDFN

Vol.
IX

Page
A.3.9

Rev.
1

Date
6/73

Purpose: Define a special set of Record Name Qualifiers in the \$JCB vector. (Names are used by BNAM when searching for data.)

Usage: CALL \$YUDFN (\$JCB {, <q>}₁⁴)

Program Logic: \$JCB(10) thru \$JCB(13) are cleared to blanks and the appropriated number of Record Name Qualifiers, <q> 's, are sequentially added beginning in \$JCB(10).

Language: OS/360 Assembler.

Author: R. S. Wood

JOSHUA DPSTM-500	Title	Subroutine: \$YUDFN	Vol.	Page
			IX	A.3.10
			Rev.	Date
			1	6/

JOSHUA

DPSTM-500

Title	Module: \$YUDPR	Vol.	Page
		IX	A.3.11
Rev.	1	Date	6/73

Purpose: In the execution of the JOSHUA 'PRINT' statement, to establish the required values and space represented by the argument list to the module DDISPLAY and to execute a link to that module.

Attribute: Nonreusable.

Entry Point: \$YUDPR

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
\$MCB	Sixteen double words.	MCB communication Array.
N	One full word.	Number of names in the PRINT statement.
A(1)-A(N)	One double word each.	The names given in the PRINT statement. Each name is a separate argument.

External Reference: ØPEN, LINK, CLØSE

Data Sets: Partitioned data set containing module DDISPLAY.

Program Logic: Fifteen arguments are presented to DDISPLAY. Space represented by these arguments is defined within \$YUDPR (except for \$MCB, whose address is passed to DDISPLAY as received and only as much space as is used by DDISPLAY is defined. The following values are set or constants defined:

1. Arguments A(1) thru (A(N) are loaded into contiguous double words of the second argument to DDISPLAY.
2. The value N is set in the eighth argument to DDISPLAY.
3. The third word of the tenth argument is set to zero.
4. The ninth word of \$MCB is set to zero.
5. The twelfth argument to DDISPLAY is set to one.
6. The fifth word of \$MCB (the communication data set display reference number) is moved to the seventh argument to DDISPLAY.

JOSHUA DPSTM-500	Title Module: \$YUDPR	Vol. IX	Page A.3.12
		Rev. 1	Date 6/7

7. The batch terminal flag is transferred from the seventh word of \$MCB to the ninth argument to DDISPLAY.
8. Each byte of the double word passed as the third argument is defined as hexadecimal '55'.

Registers are then restored since they will be saved by DDISPLAY at the address given \$YUDPR by the routine invoking it.

The data control block for the library containing DDISPLAY is completed by an ØPEN macro, DDISPLAY linked and executed, and a CLØSE macro issued.

Register 14 is saved locally before executing the ØPEN macro. Register 15 is saved between execution of the LINK and CLØSE macro instruction since at that point it contains a condition code returned from DDISPLAY. After the CLØSE, registers 14 and 15 are restored and execution terminated by a branch to the address in register 14.

Language: OS/360 Assembler.

Author: R. C. Haywood

JOSHUA

DPSTM-500

Title Function: \$RAJ

Vol.
IX

Page
A.3.13

Rev.
1

Date
6/73

Purpose:

Generate a double precision Hollerith field suitable for use as a JOSHUA Record Name Qualifier.

Usage:

Value = \$RAJ(code,X)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
Value	R*8	Double precision receiving field.
Code	I*4	Code indicating the FORTRAN type of the X mnemonic. 1 Logical*1 2 Integer*2 3 Logical*4 4 Integer*4 5 Real*4 6 Real*8 7 Complex*8 8 Complex*16

X

Any

Subprograms Used: \$CDINT, HCDINT

Program Logic:

For all FORTRAN types except Integer*2 (code = 2) and Integer*4 (code = 4) the X mnemonic is assumed to contain Hollerith information which is returned left adjusted and packed on the right with blanks. The two Integer types are assumed to contain binary integer information which is converted to a Hollerith before being passed back left adjusted and packed on the right with blanks.

Language:

OS/360 Assembler.

Author:

R. S. Wood

JOSHUA DPSTM-500	Title Function: \$RAJ	Vol. IX	Page A.3.14
		Rev. 1	Date 6/7

Title	Function: \$CDINT and HCDINT	Vol. IX	Page A.3.15
		Rev. 1	Date 6/73

Purpose: Generate a double precision Hollerith field from an integer value.

Usage: Value = \$CDINT(N [,w])
Value = HCDINT(M [,w])

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
Value	R*8	Double precision receiving field.
N	I*4	Integer to be converted.
M	I*2	Integer to be converted.
w	I*4	Controls the: <ul style="list-style-type: none"> width of the receiving field the left or right adjustment the packing of zeroes or blanks

Program Logic: The integer N/M is converted to a Hollerith and returned based on the following table

	<u>w is not Present</u>	<u>$1 \leq w \leq 8$</u>	<u>$-8 \leq w \leq -1$</u>
Receiving Field Width	8	/w/	/w/
Hollerith is Left or Right Adjusted	Left	Right	Right
Packed on the Left With Zeroes or Blanks	-	zeroes	blanks

If w = 0 a field of blanks is returned. If the Hollerith exceeds the receiving field width then a field of asterisks (*) is returned.

Language: OS/360 Assembler.

Authors: R. S. Wood, A. H. Dobyns, R. L. Boyce

Purpose: Provides a transparent linkage to the \$LINK routine from a user's module.

Usage: CALL \$YUDEX([& label,] ,0|1,'routine'
[parameter list]).

Arguments:

<u>Parameter</u>	<u>Description</u>
& label	Optional label from the err condition.
0 1	Key = 0 when no label present and = 1 when label is present.
'routine'	Module name to which control is passed by \$LINK routine.
parameter list	Normal FORTRAN parameter list which is passed intact to 'routine'.

Program Logic: This routine allows a user to pass control to another module from any subprogram in the current module.

- The initial call to \$YUDEX is made immediately after the current module receives control. Save area registers and other addresses are saved for subsequent calls to \$YUDEX.
- On subsequent calls to \$YUDEX the registers are loaded with addresses which make it appear to the \$LINK routine that control was passed to it from the main program of the current module.

Language: OS/360 Assembler.

Author: R. S. Wood

JOSHUA

DPSTM-500

Title Function: \$FLCØ6

Vol. IX	Page A.3.19
Rev. 1	Date 6/73

Purpose: Return the \$JCB error flag value.

Usage: Value = \$FLGØ6(\$JCB)

Parameters: Value. In OS/360 FORTRAN can be considered Logical*4, Logical*1, Integer*4, or Integer*2. If the error flag was on then 'value' will be TRUE or 1 depending upon the Type, Logical or Integer, respectively.

Program Logic: The error flag, \$JCBI(6), is returned to the user and subsequently \$JCBI(6) and \$MCBI(6) are set to zero. The \$MCB address is obtained from the \$JCB.

Language: OS/360 Assembler.

Author: R. S. Wood

JOSHUA DPSTM-500	Title Function: \$FLGØ6	Vol. IX	Page A.3.20
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Subroutine: \$YUDj (j = 98/99)

Vol.

IX

Page

A.3.21

Rev.

1

Date

6/73

Purpose: Clean up after a user's module and issue the STOP number and/or message.

Usage: CALL \$YUD98
CALL \$YUD99(\$LGL\$(1),\$LNAM,n)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
\$LGL\$(1)	L*1	Invariant Qualifier Mask used to get the JOSHUA STOP message from the \$LNAM array.
\$LNAM	R*8	Contains the STOP message in 8-byte groups.
n	I*4	The STOP number.

Program Logic: The linkages needed by the JOSHUA monitor to handle STØP source statements are made. (See the flow diagram.).

Remarks:

- Simple STØP statements -
STØP or
STØP 0
use a CALL \$YUD98.
- Qualified STØP statements
STOP[n]['message']
use a CALL \$YUD99(\$LGL\$(1),\$LNAM,N). The Precompiler will initialize the \$LGL\$ and \$LNAM arrays and will generate the appropriate CALL statement when a STOP is encountered in the source program.

Language: OS/360 Assembler.

Author: R. S. Wood

JOSHUA

DPSTM-500

Title	Subroutines: \$\$1, \$\$2, \$\$3, \$\$4, \$\$5 (ENTRY points in one sub-program.)	Vol.	Page
		IX	A.3.23
Rev.	1	Date	6/73

Purpose: Provide the net cpu cycles expended in each subroutine and print out the results.
(See PROGRAM TIMINGS, Section 1.4.30.)

Usage: CALL \$\$1
CALL \$\$2
CALL \$\$3(\$LGL\$, 'name')
CALL \$\$4(\$LGL\$)
CALL \$\$5

Parameters: \$LGL\$ L*1 Logical constant defined by the precompiler for each subroutine.

'name' R*8 Double precision variable with the name of the subroutine or entry point being timed.

Common Area: \$\$TM

Program Logic: \$\$1 Begins or restarts the timing of user subprograms by setting a logical variable in \$\$TM COMMON to TRUE.

\$\$2 Stops the timing of user subprograms by setting a logical variable in \$\$TM COMMON to .FALSE. and gets a net time for the current subprogram.

\$\$3 Time stamps the entrance into a subprogram.

\$\$4 Time stamps the exit from a subprogram; accumulates the net time and increments a counter by one.

\$\$5 Prints out any accumulated statistics.

Routines Used: CPUT A REAL*8 function which returns the cpu cycles remaining available to this job.

Language: OS/360 FORTRAN IV

Author: R. S. Wood

JOSHUA

DPSTM-500

Title Function: CPUT

Vol.
IX

Page
A.3.27

Rev.
1

Date
6/73

Purpose: Return the remaining CPU time in a job step.

Usage: T = CPUT(X)

Argument:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
X	R*8	Variable which is set to the CPU time remaining in the job step.
T	R*8	This variable also receives the remaining job step CPU time.

Language: OS/360 Assembler

Author: L. F. Zimmerman

Remarks: CPU time is returned in seconds. The argument and the receiving variable will have the same value when using CPUT and therefore may be the same mnemonic.

JOSHUA DPSTM-500	Title Function: CPUT	Vol. IX	Page A.3.28
		Rev. 1	Date 6/73

JOSHUA
DPSTM-500

Title Part B - Data Management

Vol.

IX

Page

E.O.1

Rev.

1

Date

6/73

Contents

- B.1 Introduction to JOSHUA Data Management.
- B.2 Direct Access Storage Management.
- B.3 Basic Named Access Method (BNAM).
- B.4 Data Management Modules.

JOSHUA DPSTM-500	Title Part B - Data Management	Vol. IX	Page B.O.2
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Introduction to JOSHUA Data Management

Vol.
IX

Page
B.1.1

Rev.
1

Date
6/73

Work Done By

J. C. Jensen
F. D. Knight

Report Written By

F. D. Knight

Contents

	<u>Page</u>
1. Introduction	B.1.3

JOSHUA DPSTM-500	Title Introduction to JOSHUA Data Management	Vol. IX	Page B.1.2
		Rev. 1	Date 6/73

(Blank)

1. Introduction

The keystone of the JOSHUA System is the set of data management facilities provided to dynamically manage the physical space allocated to JOSHUA and to allow for the use of named data records^(1,2,3,4) in the creation, retrieval, update, and deletion of specified data. The dynamic management of JOSHUA space consists of allocating requested blocks of direct access space to various Standard, User, Job, Scratch, and Edit data sets and later regaining this space for future allocation. The ability to use named data records is provided by the JOSHUA data manager, BNAM (Basic Named Access Method), which uses a tree structure to associate the logical data to be processed with the physical record(s) where the data is stored. Once this association has been established, the physical data transfer is accomplished by using either the FORTRAN I/O or local BDAM routines. In carrying out its functions of data management, BNAM is characterized by dynamic core allocation for buffer space, the use of a data set hierarchy of search, the maintenance of paths of search for previous records as an aid to future retrieval, and protection of data from unauthorized retrieval and update. In the following sections, direct access space management and BNAM will be discussed along with documentation of the data management programs.

-
- (1) DP-MS-68-92, "Data Management for Multiaccess Computational Systems," J. E. Suich, J. C. Jensen, and H. C. Honeck, presented at the Conference on The Effective Use of Computers in The Nuclear Industry, Knoxville, Tenn., April 21-23, 1969.
 - (2) DP-MS-68-94, "JOSHUA - A Reactor Physics Computational System," H. C. Honeck, J. E. Suich, J. C. Jensen, C. E. Bailey, and J. W. Stewart, presented at the Conference on The Effective Use of Computers in The Nuclear Industry, Knoxville, Tenn., April 21-23, 1969.
 - (3) DP-MS-70-70, "Alphameric Displays in Nuclear Reactor Computations," J. C. Jensen, J. E. Suich, F. D. Knight, H. C. Honeck, presented at the 1970 Winter Meeting, American Nuclear Society, Washington, D.C., November 15-19, 1970.
 - (4) DP-MS-70-82, "The JOSHUA Terminal System," J. C. Jensen, R. L. Boyce, J. E. Suich, and R. C. Haywood, presented at the meeting on New Developments in Reactor Mathematics and Applications, Idaho Falls, Idaho, March 29-31, 1971.

JOSHUA DPSTM-500	Title Introduction to JOSHUA Data Management	Vol. IX	Page B.1.4
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title Direct Access Storage Management

Vol.
IX

Page
B.2.1

Rev.
1

Date
6/73

Work Done By

J. C. Jensen

Report Written By

J. C. Jensen
F. D. Knight

Contents

	<u>Page</u>
1. Introduction	B.2.3
2. Name Catalog Data Sets	B.2.5
3. Tree Catalog and Data Storage Data Sets	B.2.9
4. JOSHUA Data Set Management	B.2.11

JOSHUA DPSTM-500	Title Direct Access Storage Management	Vol. IX	Page B.2.2
		Rev. 1	Date 6/73

(Plank)

JOSHUA DPSTM-500	Title Direct Access Storage Management	Vol. IX	Page B.2.3
		Rev. 1	Date 6/73

1. Introduction

The JOSHUA data base is composed of a number of OS/360 physical data sets that reside on direct access storage with each data set having a fixed block, direct access organization. The data sets are used for JOSHUA data set name catalogs, tree catalogs, and data record storage.

A JOSHUA data set is defined by the first qualifier of a data record name, e.g., the record named ASSEMBLY.MK16.SIZE is a member of the JOSHUA ASSEMBLY data set. Each JOSHUA data set is composed of one or more non-contiguous physical areas called data set extents and a single associated tree catalog, which are allocated to portions of one or more OS/360 physical data sets and catalogued by name in the JOSHUA System. Standard and User data sets and data set extents are allocated upon request, whereas Job and Scratch data sets are dynamically allocated at job execution time and are limited to one extent. Edit data sets are allocated dynamically as a result of an EDIT request for work space and are also limited to one extent. Note that a JOSHUA data set is considered to be a combination of the areas allocated for both data storage and the associated tree catalog.

JOSHUA DPSTM-500	Title Direct Access Storage Management	Vol. IX	Page B.2.4
		Rev. 1	Date 6/73

2. Name Catalog Data Sets

Three separate physical data sets are used to allocate and catalog a JOSHUA data set.

- 1) JOSHUA Data Set Name Catalog (JDSNC).
- 2) JOSHUA Data Set Control Block (JDSCB).
- 3) JOSHUA Catalog Extension Block (JCEB).

The format of each data set is shown in Figures 1, 2 and 3, respectively.

The data set name catalog contains only the names of JOSHUA data sets. Each name occupies eight bytes of the record and all positions that do not contain a name contain the word FREE. Four 784 byte records are provided for this catalog, thus allowing for 392 data set names. The JDSNC is updated whenever a JOSHUA data set is allocated or scratched.

For each data set name there is an associated data set control block and a catalog extension block. The association is determined from the position of the data set name in the JDSNC, i.e., the first physical record in the JDSCB and JCEB data sets will correspond to the first name in the JDSNC.

The JDSCB contains the data necessary for BNAM to store and retrieve data records for a specified JOSHUA data set. There is one 40-byte physical record for each JOSHUA data set name appearing in the JDSNC. These records are updated when a JOSHUA data set is allocated, when the release option is encountered by the SCRATCH module, and when BNAM updates a tree catalog.

The JCEB contains the data necessary for the management of dynamically allocated data set space, e.g., expiration date, data set type, etc. There is one 100-byte physical record for each data set name appearing in the JDSNC. These records are updated when a JOSHUA data set is initially allocated or processed as an old data set, and are referenced only by the SCRATCH module.

If a JOSHUA data set consists of more than one data set extent, then separate JDSNC, JDSCB, and JCEB entries are required for each extent. These extents are linked together as a logical continuum via the tree pages specified within the JDSCB of the first extent.

JOSHUA DPSTM-500	Title Direct Access Storage Management	Vol. IX	Page B.2.6
		Rev. 1	Date 6/73

* The JDSNC data set contains 4 records.

Figure 1. JOSHUA Data Set Name Catalog (JDSNC).

DSN	FC	TR	TB	TL	TN	TS	DR	DB	DL	DS	EX	CR	RESERVED*
1	9	13	15	17	19	21	23	25	27	29	31	33	35 40

DSN** - JOSHUA data set name

FC - File Category

TR - Tree catalog reference number

TB - Tree catalog base (1st record number - 1)

TL - Number of tree records used

TN - Number of cells used in the last used tree record

TS - Number of allocated tree records

DR - Data storage reference number

DB - Data storage base (1st record number - 1)

DL - Number of data records used

DS - Number of allocated data records

EX - Data extent

CR*** - Physical record number of entry in JDSCB data set

* Used for update flag and last core position of JDSCB.

** DSN is stored as an EBCDIC value. All other variables are binary integers.

*** The JDSCB data set contains 392 records.

Figure 2. JOSHUA Data Set Control Block (JDSCB)

JOSHUA DPSTM-500	Title Direct Access Storage Management	Vol.	Page
		IX Rev. 1	B.2.8 Date 6/73

DSN	CD	ED	UNAME	RESERVED*		SF	TF	BS
1	9	13	17	25		95	97	99 100

DSN** - JOSHUA data set name
 CD - Creation date (month/day/year)
 ED - Expiration date (month/day/year)
 UNAME** - User data set name
 SF - Status flag 0 - online, 1 - offline
 TF - Type flag 1 - Standard, 2 - User, 3 - Job, 4 - Scratch,
 5 - Edit
 BS - Block size (bytes)

* Used by the ALLOCATE module for special flags and messages.

** DSN and UNAME are stored as EBCDIC values.
 All other variables are binary integers.

NOTE: The JCEB data set contains 392 records.

Figure 3. JOSHUA Catalog Extension Block (JCEB)

3. Tree Catalog and Data Storage Data Sets

The JOSHUA data base, exclusive of the data set name catalogs and control blocks, consists of allocated areas for tree catalogs and data storage on a number of OS/360 physical data sets. These physical data sets contain records of 784 bytes for tree catalogs and 400 bytes for data storage and are logically allocated to many JOSHUA data sets.

The first record of each physical data set is used to account for the free space within the data set. Each free space element (FSE), occupies 4 bytes of the record. The FSE is composed of two half words; the first half word is the starting location of a block of free space; the second half word is the number of free records in the block. If the data set has no allocated space or if all space is allocated, there will be only one FSE entry as shown below for a data set containing 2800 records:

No space allocated - 2,2799

All space allocated - 2801,0

The FSE's are stored as binary integers and are kept in numerical order based on the first half of the entry. The last valid entry in the record will always be followed by a zero. Since the record sizes are different for the tree catalog data sets and the data storage data sets, the maximum number of FSE entries is 195 and 99 respectively.

JOSHUA DPSTM-500	Title Direct Access Storage Management	Vol. IX	Page B.2.10
		Rev. 1	Date 6/7

(Blank)

4. JOSHUA Data Set Management

JOSHUA data sets are allocated by locating the smallest blocks of available space on specified physical data sets for both the tree catalog and data storage. The first record of each physical data set, the FSE record, is then updated with a new set of free space elements reflecting the allocation changes. In addition, the data set name is added to the name catalog, and the data set control block and catalog extension block are written. The data sets are removed from the data base by removing the name from the catalog and updating the appropriate FSE records to reflect the additional free space. The associated data set control block and catalog extension block are ignored since they can no longer be located when the name has been removed from the name catalog.

Standard and User data sets and data set extents are allocated with a system utility program. The JOSHUA Monitor System (JMS) allocates Job, Scratch, and Edit data sets based on a user supplied space requirement. An active Scratch data set is removed from the data base at the termination of each job. Any inactive Scratch data sets, i.e., those remaining due to abnormal job termination, are removed when a job is initiated. Unused space in Job data sets may be released or the entire data set removed from the system based on user supplied symbolic JCL parameters and input cards. The JMS SCRATCH function may be used to remove any data set from the data base. Job and Edit data sets that have passed their expiration dates or contain no tree entries are removed from the data base with a system utility program (normally on a daily basis).

JOSHUA DPSTM-500	Title Direct Access Storage Management	Vol. IX	Page B.2.12
		Rev. 1	Date 6/73

(Blank)

Work Done By

J. C. Jensen
F. D. Knight

Report Written By

J. C. Jensen
F. D. Knight

Contents

	<u>Page</u>
1. Data Organization	
1.1 General	3.3
1.2 Cell and Page Definition	3.5
1.3 Searching Strategy and Path Definition	3.10
1.4 Storage Considerations	3.15
2. BNAM Communication and Control Blocks	
2.1 General	3.19
2.2 System Communication Control Block	3.19
2.3 In-Core Path-Page Blocks	3.21
2.4 Paths on Disk Control Blocks	3.22
2.5 BNAM Communication Control Block	3.23
2.6 Status Control Block	3.25
2.7 Catalog Control Block	3.25
2.8 System Definition Control Block	3.26
2.9 Auxiliary Control Variables	3.27
2.10 Page Option Control Block	3.28
2.11 Data Set - Category Control Block	3.29
3. BNAM Common Areas	
3.1 General	3.31
3.2 BNAMCM	3.31
3.3 COMNEW	3.32
3.4 RNAME	3.32
3.5 PROTEC	3.33
3.6 STATIS	3.33
4. Program Logic	
4.1 Overview	3.35
4.2 Discussion of BNAM Logic	3.36
4.2.1 General	3.36
4.2.2 Features	3.38
5. BNAM Errors	
5.1 General	3.41
5.2 Error Numbers for Invalid User Requests	3.41
5.3 Loss of Control Abends (JTSEN)	3.43
6. BNAM Statistics	3.45

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.2
		Rev. 1	Date 6/7

(Blank)

1. Data Organization

1.1 General

The information structure associated with using the named data record concept for the storage and retrieval of data is a general tree, with each level of the tree corresponding to a specific qualifier position and the terminal level indicating the physical location of the data. This information structure is programmed using standard binary tree representation⁽⁵⁾ and is illustrated in Figure 4 for the subtree DSNAME.A1. In this figure, each alphanumeric symbol, A1, B1, etc., represents one of the qualifiers or "names" used to reference a named data record as well as representing the nodes of a tree. All nodes at the same level and connected in a circular chain are called "brothers", i.e., C1, C2, C3. The node immediately below a given brother is called a "son", i.e., G1 is a son of C2 and C2 is a "father" of G1. It is noted that in this standard binary tree representation of a general tree, a father points only to the first son and to his next brother. In the figure, the request for the record name, DSNAME.A1.B1.C2.G1 is found to correspond to R4, the fourth named data record in the data set. This may or may not be the fourth physical record depending on the length of named data records 1, 2, and 3. If a record is desired, searching begins at the first node and matching is attempted. This continues at that tree level until a match is made or the brothers exhausted. If a match is found, the subtree of this matched "father" is traversed in a similar manner until either the lowest level is matched (indicating a data pointer) or the name is impossible to locate. If a data pointer is found, it is returned for use in subsequent data transfer provided the requestor is authorized. Otherwise, the requestor is returned the appropriate error number.

(5) "The Art of Computer Programming", Fundamental Algorithms Vol. 1, Chap. 2.3, D. E. Knuth, Addison Wesley Publishing Company, Reading, Mass., 1968.

Figure 4. Information Structure for JOSHUA Named Data Records

1.2 Cell and Page Definition

In order to traverse the tree for a name search, certain information is necessary for each node. This information is the name corresponding to the node, the pointer to the next brother, and the pointer to the first son. This node information or "cell" is illustrated in Figure 5 and consists of a double word for the name and single words for the brother and son pointers.

Figure 5. Cell Contents

The JOSHUA tree catalogs are composed of many cells of this type which are grouped into convenient size blocks or "pages" in the order in which the named data records are created. It is these pages which are located on disk in the allocated tree catalog area. A more detailed breakdown of these cell pointers is given in Figure 6.

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.6
		Rev. 1	Date 6/7

HW1		HW2	
Last Brother Switch	Son Extent	Brother Cell	Brother Page
HW3 Son Cell or Negative List Length in Double Words		HW4 Son Page or Relative Record	

Figure 6. Cell Pointers: Brother, Son

The word that contains the brother pointer is subdivided to contain four items of information. The first halfword contains three items, the last brother switch, the son extent, and the brother cell. The last brother switch (4 bits) contains a 1 for all nodes except when the node is the last brother in a chain in which case it is 0. Whenever the data storage area of a data set is filled, BNAM provides for the use of data extents to chain two or more non-contiguous areas together to allow for more records. The son extent (4 bits) contains a 1 except for a terminal or data node, in which case it gives the extent where the data is located. This use of a son extent pointer to non-contiguous physical areas enables the implementation of a record level protective handling capability for the JOSHUA data base. For protected data, the son extent points to a separate physical data set which contains the protected category of data. When coupled with an authorization table by data category, the protected handling capability becomes operable. The brother cell pointer contained in the last byte and the brother page contained in the second halfword provide the location within the tree catalog for the next brother in the chain. If the current node is the last brother, then the pointers give the location of the first brother due to the circular linked structure.

The third and fourth halfwords respectively give the cell and tree page for locating the first son of the current node, except where the qualifier is terminal as indicated by a negative number in the third halfword. In this case, the third halfword gives the negative of the data record length expressed in double words and the fourth halfword gives the relative record number of the data record. Since the maximum number of double words that can be contained in a halfword is 32767, the maximum data record possible under BNAM is 8 x 32767 or 262136 bytes.

JOSHUA

DPSTM-500

Title Basic Named Access Method (BNAM)

Vol.

IX

Page

B.3.7

Rev. 1

Date

6/73

The first cell of each page is reserved for special flags which identify it out of context and indicate its status. The name field is used for the "data set name" and the brother pointer is used to record (1) the last cell of the page which contains valid information and (2) the relative record number or page number of each page. This information is used by BNAM for proper data set and page number verification.

This data structure for a JOSHUA Named Data Record Catalog with six cells/page is illustrated in Figure 7. The negative entries indicate terminal qualifiers and the R entries give the relative record number of the data. The number of cells/page in the actual system is discussed later.

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. 1X	Page B.3.8
		Rev. 1	Date 6/73

<u>Page</u>	<u>Cell</u>	<u>Name</u>	<u>Brother</u>				<u>Son</u>	
			<u>BSWICH</u>	<u>SEXT</u>	<u>CELL</u>	<u>PAGE</u>	<u>CELL</u>	<u>PAGE</u>
1	1	DSNAME	0	0	6	1		
	2	A1	1	1	3	8	3	1
	3	B1	1	1	6	3	4	1
	4	C1	1	1	3	2	5	1
	5	F1	1	1	6	1	-L1	R1
	6	F2	1	1	2	2	-L2	R2
2	1	DSNAME	0	0	6	2		
	2	F3	0	1	5	1	-L3	R3
	3	C2	1	1	2	3	4	2
	4	G1	1	1	5	2	-L4	R4
	5	G2	1	1	6	2	-L5	R5
	6	G3	0	1	4	2	-L6	R6
3	1	DSNAME	0	0	6	3		
	2	C3	0	1	4	1	3	3
	3	H1	1	1	4	3	-L7	R7
	4	H2	1	1	5	3	-L8	R8
	5	H3	0	1	3	3	-L9	R9
	6	B2	1	1	4	6	2	4
4	1	DSNAME	0	0	6	4		
	2	D1	1	1	6	4	3	4
	3	I1	1	1	4	4	-L10	R10
	4	I2	1	1	5	4	-L11	R11
	5	I3	0	1	3	4	-L12	R12
	6	D2	1	1	5	5	2	5

Figure 7. Data Structure for A JOSHUA Named Data Record Catalog

Page	Cell	Name	Brother				Son	
			BSWICH	SEXT	CELL	PAGE	CELL	PAGE
5	1	DSNAME	0	0	6	5		
	2	J1	1	1	3	5	-L13	R13
	3	J2	1	1	4	5	-L14	R14
	4	J3	0	1	2	5	-L15	R15
	5	D3	0	1	2	4	6	5
	6	K1	1	1	2	6	-L16	R16
6	1	DSNAME	0	0	6	6		
	2	K2	1	1	3	6	-L17	R17
	3	K3	0	1	6	5	-L18	R18
	4	B3	0	1	3	1	5	6
	5	E1	1	1	4	7	6	6
	6	L1	1	1	2	7	-L19	R19
7	1	DSNAME	0	0	6	7		
	2	I2	1	1	3	7	-I20	R20
	3	L3	0	1	6	6	-I21	R21
	4	E2	0	1	5	6	5	7
	5	M1	1	1	6	7	-I22	R22
	6	M2	1	1	2	8	-I23	R23
8	1	DSNAME	0	0	6	8		
	2	M3	0	1	5	7	-I24	R24
		etc.						

Figure 7. Data Structure for A JOSHUA Named Data Record Catalog
(continued)

1.3 Searching Strategy and Path Definition

Every reference to a JOSHUA data record, whether initiated by a terminal command or computational module I/O, necessitates a search of a tree to locate or assign the record reference number. These searches must be efficient in the sense that if the references are "sequential", i.e., in the same order as the logical tree structure, most references are resolved with no accesses to disk to retrieve additional pages, and with only a minor amount of computation. On the other hand, if the references are "random", i.e., with no relation between successive references and the logical tree structure, then the user is availing himself of the full retrieval capability of the access method, and some overhead must be paid in terms of disk accesses and computational time. An objective of the design of BNAM was to minimize this overhead while working within a limited amount of core storage.

To attain a high efficiency for sequentially ordered searches, provision has to be made to avoid retracing names through pages previously brought into core storage. In the example given by Figures 4 and 7, the node A1 is a qualifier of every record name and once page 1 has been accessed, node A1 should be saved for all future references. In general, to avoid re-accessing any previous page on a sequential search, it is both necessary and sufficient to retain all nodes visited in traversing the tree on each named data record search. This set of nodes is called a "path".

For example, the path for the retrieval of DSNAMES.A1.B1.C2.G1 would be as shown in Figure 8.

Cell	Name	Brother				Son		
		BSWICH	SEXT	CELL	PAGE	CELL	PAGE	TAG
1	DSNAME	0	1	2	1			1
2	A1	1	1	3	8	3	1	1
3	B1	1	1	6	3	4	1	1
4	C1	1	1	3	2	5	1	0
5	C2	1	1	2	3	4	2	1
6	G1	1	1	5	2	-L4	R4	3

Figure 8. Data Structure of A JOSHUA Path

In this figure there are four entities of interest, the name, the brother pointers, the son pointers, and the tag. All of these have been previously discussed with the exception of the tag. As nodes are encountered during a search, the corresponding cells are moved from the tree page into the path. In the path, the page number and cell location at which a node is to be found, are contained in the cell immediately prior to the node in question, as either the brother or the son pointers. Since a node in a path cannot be identified as having been chained as a son or as a brother from the cell contents alone, each node must be tagged to indicate the relationship. This is accomplished by the use of a one-byte tag which is 0 when there is a brother relationship with the next path node; 1 when there is a son relationship with the next path node; 3 when the node is the end of the path; and 4 when the path would exceed the maximum path length if continued. Thus those nodes tagged with 1's constitute the complete record name. The last qualifier with a tag of one is overridden with a tag of 3 if it is a terminal qualifier.

The use of the path concept and the tree pages previously defined is central to the tree searching strategy of BNAM, which is to match a given name string as far as possible with a path and then utilize in-core pages to continue the search. Thus the efficient search problem is equivalent to the efficient management of core space to contain the paths of several current data set references, along with the tree pages that generated the most recent path nodes.

Since both pages and paths appear in core together, the use of a linked list of space is used to allow for as many variable length paths as possible. A standard size core block for a tree page based on the tree structure and 2314 characteristics was decided upon and these core blocks are used for both paths and pages. The use of a standard size block allows pages as well as path blocks to be read in and out of core as a single block, without cell-by-cell restructuring. Thus core storage at any given time appears as a series of blocks filled with paths and/or pages as the case may be, with paths chained together when they exceed one block of core as shown in Figure 9. The information necessary to distinguish a path from a page, and data set from data set, is contained in two BNAM control blocks to be defined in a later section. Since pages and paths will reside in the same size core blocks, the structure of the two within the blocks must be defined. A page will logically contain N 16-byte cells with the N names grouped contiguously and the N sets of pointers likewise. The only difference between a page and the path as far as structure is concerned is the existence of the tag bytes for a path. These tag bytes could be provided in a separate storage array; but since paths are of variable length and complete

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.12
		Rev. 1	Date 6/73

blocks of storage will not be fully utilized, it is preferable to devote a portion of the path block itself to the tag table. This detailed layout is shown in Figure 10 along with a layout of a page block. Each cell given up to the tag table provides for tagging 16 path nodes; hence 3 cells used for tagging allow a maximum of 48 path nodes/block.

If N-K cells are to be used for tagging, then K cells will be left for containing the path nodes. Thus a path will contain K 16-byte cells plus tag, with the K names grouped contiguously as are the K sets of pointers and K tags.

These path-page blocks are also used to core contain a number of JOSHUA Data Set Control Blocks (JDSCB) in order to reduce the overhead involved with locating JOSHUA data sets. Each block may contain 19 JDSCB entries and can be chained together in a manner analogous to a path.

Figure 9. Storage Layout for In-Core Path-Page Blocks

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.14
		Rev. 1	Date 6/77

Cell		LSWICH	SEXT	CELL	PAGE	CELL	PAGE
PAGE BLOCK	1	DSNAME					
	2	<div>Names</div> <div>2-N</div> <div>Brothers</div> <div>Sons</div>					
	3						
	4						
	.						
	.						
	.						
	.						
	.						
	.						
	.						
	.						
	N						
	Word	1	2	3		4	

Cell	LSWICH				SEXT		CELL			PAGE			CELL		PAGE	
1	Names 1-K								Brothers						Sons	
2																
3																
4																
5																
6																
7																
.																
.																
.																
.																
.																
.																
.																
N																
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10

Figure 10. Page and Path Structure.

1.4 Storage Considerations

In planning the in-core and external storage requirements of the JOSHUA data base, it was necessary to decide on a block size for data, a block size for tree pages, and the maximum number of nodes to be contained in a path block. The original design considerations for JOSHUA storage were based on the IBM 2314 direct access device. With the current use of IBM 3330 disk units, the assumptions used and the decisions reached have remained valid. The following will discuss this decision process.

With the knowledge of the data to be stored, it was assumed that a record size of 400 bytes would be appropriate for most applications at SRL (82% of the records in use are of this size or less) and that no JOSHUA data set would exceed 32K records. The problem then becomes one of determining the number of tree pages required to hold 32K terminal nodes in some reasonable manner.

The tree structure which terminates in 32K nodes for the maximum size data set is specified by the user and exhibits considerable variance between data sets. If 16 qualifiers are assumed, a binary tree (each node has one brother) contains

$$1 + 2 + 2^2 + 2^3 + \dots + 2^{15} \approx 64K$$

nodes and 32K terminal nodes. A balanced t-ary tree of order 5-6 (each node has 4-5 brothers), which splits into 5-6 branches at each node will produce approximately 32K data pointers in some 5-6 generations. Experience to date has indicated that 5-6 qualifiers in a name is a reasonable average for SRL's JOSHUA applications. Such a tree (with an average of 5.6 sons per father, and 6 generations) contains

$$1 + 5.6 + (5.6)^2 + \dots + (5.6)^6 \approx 38K$$

nodes. A tree with many sons per family at the terminal level will also contain ~32K total nodes. Thus it is reasonable to assume that any likely tree structure will contain between 1 and 2 nodes per data pointer, and 2 nodes/data record was chosen as a reasonable upper limit estimate.

A node requires space for an 8-character name and two full words for brother and son pointers - a total of 16 bytes. At 2 nodes/data record, approximately 32 bytes of catalog are required for 400 bytes of data, or 8% of the total disk space.

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol.	Page
		IX	B.3.16
		Rev. 1	Date 6/73

To limit disk accesses for tree pages to 4% of those for data references, the page must contain 25 data references or approximately 50 nodes. The number of cells per page actually used is 49 because this record size makes better utilization of 2314 space. This page size has also been carried forward for 3330 use. With the page size established at 49 16-byte cells per page or 784 bytes, the maximum number of path nodes plus tags that can be contained is given by $784/(16 + 1) \approx 46$ nodes/path block.

If extensive use of random references to the tree are made, many references to the pages will be generated. However, the number of searches in a large tree is insensitive to page size, which may be seen as follows. The binary tree would contain about 16 terminal nodes on each page, since

$$1 + 2 + \dots + 2^4 = 31 \leq 50$$

which would allow the last five names to be found with one access.

However, for an assumed 16 qualifier name, one page per name would be required for the remaining 11. Thus, a random reference to a binary tree would require some twelve pages to be accessed. Increasing the page size to about 100 nodes would only reduce the number of accesses by one because of the rapidity with which 2^n increases. Similar reasoning applied to the t-ary tree of 6 levels of 5.6 names each, leads to an estimate of some 14 accesses per reference. Only the tree structure with many terminal sons per family will benefit on a random search from increased page size, and here the benefit is related only to the last generation search. Therefore not much improvement in disk accesses can be expected from increased page size.

We turn now to core storage requirements. To avoid multiple accesses to tree pages, it is necessary to core contain the longest path through the tree. For the binary tree, this is only 31 nodes; for the t-ary tree, only 5.6 nodes at each of 6 levels or 35 nodes. However, for the unbalanced tree with many brothers in some families, and few in others, the maximum path may be quite long. For example, a data set containing input records to a production program may have a series of case names plus minor auxiliary structure with a resulting path length approximately equal to the number of case names. In order to handle such a path, core storage requirements and/or disk activity to roll-out/roll-in paths and pages will increase. Core storage necessary to provide for a path of 322 nodes has been provided along with the ability to request at execution time the additional core necessary for the handling of larger paths. If a path exceeds 322 nodes and additional space has not been provided, no path is kept beyond the 322nd node. Any accesses beyond this node must "page" again for each subsequent request in this area.

In deciding on the number of in-core blocks to allow, another question that had to be answered was the likely number of data sets which would be active simultaneously. In the batch environment, each job contains its own copy of the data management routines and hence the number of simultaneously active data sets is the determining factor. The normal operation is to have two active data sets, one for reading either input or standard data and the other for the writing and reading of either intermediate or final results. Thus, space sufficient to contain two paths of approximately 140-160 nodes each or 1 path of the maximum 322 nodes will normally accommodate computational modules without excessive roll-out/roll-in activity.

Since all terminals share a single copy of the data management routines, the sum of their active data sets is the determining factor. On a MODIFY operation, three data sets are normally required: the TEMPLATE data set containing format control data, a STANDARD data set for the initial data, and the USER data set for the modified data. However, at any point in time, only two data sets, TEMPLATE plus either USER or STANDARD, are required. Therefore, most operations require two data sets for each of 22 on-line terminals or 44 data sets to be interleaved if all terminals are operating. Assuming that only half these terminals are in use at one time and that the average data set requires 2 path blocks, this amounts to $22 \times 784 \times 2 \approx 34K$ bytes of storage required to support this on-line operation. However, since terminal references occur on the order of one per 1-2 seconds, a high roll-out/roll-in frequency can be tolerated as a substitute for in-core storage since all 44 path blocks can be rolled out and back in within approximately 2 seconds.

In summary, the design estimates for storage requirements are:

Data Sets : Maximum of 32K records of 400 bytes each/data set.

In-Core Blocks : Ten in-core blocks of 784 bytes each = 7840 bytes. These blocks to be used for tree pages, paths, and data set catalogs on a dynamic basis.

Tree Pages : 49 cells at 16 bytes each.

Paths : 46 cells at 16 bytes each + 46 tags at 1 byte each.

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.18
		Rev. 1	Date 6/73

(Blank)

2. BNAM Communication and Control Blocks

2.1 General

In carrying out the operations of storing, retrieving, and deleting named data records, BNAM uses a number of variables and arrays for communication and control. In this section these communication and control blocks and their associated mnemonics are defined:

- a. System Communication Control Block - CCB
- b. In-Core Path-Page Blocks - ARRAY,PGMAN8,PGMAN2
- c. Paths On Disk Control Blocks - PGMAN8,PGMAN2
- d. BNAM Communication Control Block - COMOLD(OLDCOM)
- e. Status Control Block - COMNEW(NEWCOM)
- f. Catalog Control Block - CLOG
- g. System Definition Control Block - DIM
- h. Auxiliary Control Variables

Space Management Request Option - BLOKOP

Path Selection Mode - PTHOPT

Path-Page Retrieval Option - PAGOPT

Match Name Option - MATOPT

Close Option - CLOSOP

- i. Page Option Control Block

2.2 System Communication Control Block

This control block is passed to the data manager upon entry and consists of the following entries used by BNAM:

CCB - System Communication Control Block (16) R*8

- (1) - Job data set name
- (2) - Write data set name
- (3) - Not referenced
- (4) - Environment flag - bytes 1-4

0 - terminal environment

1 - batch environment

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.20
		Rev. 1	Date 6/73

- Error flag - bytes 5-8
 - BNAM error number
- (5) - Mode flag - bytes 5-8
 - 0 - computational mode
 - 1 - functional mode
- (6) - Append flag - bytes 1-2
 - 0 - append as requested
 - 1 - do not append
- Signon flag - bytes 3-4
 - 0 - use in-core Catalog Control Block
 - 1 - use Catalog Control Block from disk
- Protected data authorization flags - bytes 5-6
- Verify flag - bytes 7-8
 - 0 - update is not authorized
 - 1 - update is authorized
- (7) - Not referenced
- (8) - Protective handling flag - bytes 1-2
 - b* - access restricted to non-protected data
 - ∅** - access to protected data, if authorized
- Read flag - bytes 3-4
 - b - ignore read flag
 - ∅ - only retrieve data for the data category given by the read flag
- Write flag - bytes 5-6
 - b - ignore write flag
 - ∅ - only write to the data category given by the write flag

b* - blank
∅** - non-blank

- Category flag - bytes 7-8

b - unprotected data retrieved

Ø - protected data of given category retrieved

(9) - Dynamic scratch data set name

(10)-(13) - Data set search sequence

(14)-(15) - Not referenced

(16) - User badge number - bytes 1-4

2.3 In-Core Path-Page Blocks

As mentioned previously, the in-core space for the paths, pages, and catalog entries is obtained dynamically at run time and is used by the above three entities as the situation demands. This in-core space (ARRAY) is managed dynamically via two control blocks (PGMAN8,PGMAN2) as follows:

Block Number	Array	PGMAN8	PGMAN2	
1	Page, Path or Catalog	DSNAME	Control Information	
2	"	"	"	"
3	"	"	"	"
.				
.				
.				
N	"	"	"	"

The ARRAY Control Block consists of N 784 byte blocks into which tree pages are read, paths are created, and catalog entries are stored. Associated with each 784 byte block are the corresponding entries in the PGMAN8 and PGMAN2 control blocks.

PGMAN8 - DSNAME Control Block - In Core (2,N) R*8

(1,X) - Data set name

(2,X) - Not used

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.22
		Rev. 1	Date 6/73

PGMAN2 - Space Management Control Block - In Core (8,N) I*2

(1,X) - Modification indicator - Page

0 - not modified

1 - modified

- Type path - Path

0 - non-template

1 - template

(2,X) - Block use

0 - Path

1 - Page

2 - Catalog

(3,X) - Least recently used counter; used to free block for other use; not applicable if catalog block.

(4,X) - Data set reference number - Page

(5,X) - Tree base - Page

- Path block sequence number - Path

(6,X) - Forward pointer - Path, Catalog

(7,X) - Backward pointer - Path, Catalog

(8,X) - Location of next in-core space for JDSCB from disk - Catalog.

2.4 Paths On Disk Control Blocks

Since, in general, many page accesses are necessary to form a path block, paths usually contain information of more lasting value and are saved on disk for later retrieval. This disk space is managed by extensions of the PGMAN8 and PGMAN2 control blocks.

PGMAN8 - DSNAME Control Block - Paths on Disk
(2,N+1)-(2,N+18) R*8

(1,X) - Data set name

(2,X) - Not used

PGMAN2 - Space Management Control Block - Paths on Disk
(8,N+1)-(8,N+18) I*2

- (1,X) - First path sequence number on disk
- (2,X) - Type path
 - 0 - non-template
 - 1 - template
- (3,X) - Least recently used counter; used to free block for other data set.
- (4,X) - Data set reference number for path on disk.
- (5,X) - Base for use with path sequence number in writing to disk.
- (6,X)-(8,X) - Not used

2.5 BNAM Communication Control Block

In order for the calling program to communicate with BNAM, there exists a communication array COMTEM which is moved into a local communication control block COMOLD (OLDCOM) for data management use.

COMOLD - BNAM Communication Control Block (16) I*2

- (1) - Functional option requested
 - 2 - Locate the data set reference number and relative record number for retrieval.
 - 3 - Locate the data set reference number and relative record number for creation or update and update the data set JDSCB.
 - 4 - Close all data set JDSCB's and tree pages to disk.
 - 5 - Delete a qualifier from a tree page or a data set from the catalog.
 - 6 - Preorder traverse a subtree for a given name string and return the data set reference number and relative record number of each record.
 - 7 - Bring in JDSCB from disk.
 - 10 - Rename a given qualifier as requested.

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol.	Page
		IX	B.3.24
		Rev.	Date
		1	6/7

11 - Return a list of all brothers at a given qualifier level.

12 - Preorder traverse a subtree for a given name string and return no data pointers.

(2) - Data set reference number of data record.

(3) - Relative record number of data record.

(4) - Number of qualifiers in name string upon entry.

(5) - Number of bytes to be written for functional option 3.

Number of bytes to be read for functional options 2 or 6.

Number of brothers returned for functional option 11.

(6) - Error number.

(7) - Number of qualifiers located.

(8) - Position in name string where a qualifier could not be matched; if retrieving template for display, contains HEX4S upon entry.

(9) - Not currently implemented - set to 16.

(10) - Maximum number of brothers requested - depends on size of user defined array.

(11) - Number of names to which list option is applied - set to 16.

(12) - Version of BNAM.

1 - Version prior to 3-11-71

2 - Version since 3-11-71

(13) - List options

1 - = number of names in (11) above

2 - $\leq$ number of names in (11) above

3 - $\geq$ number of names in (11) above

4 - $<$ number of names in (11) above

5 - $>$ number of names in (11) above

6 - $\neq$ number of names in (11) above

(14)-(16) - Not used

2.6 Status Control Block

As BNAM goes through the tree pages in search of a given qualifier, a local work array COMNEW is used to keep track of the search status.

COMNEW - Status Control Block (12) I*2

- (1) - Current qualifier location in name string.
- (2) - Block in ARRAY where current page is stored.
- (3) - Current cell in current page.
- (4) - Current page number.
- (5) - Block in ARRAY where current path block is stored.
- (6) - Current cell in current path.
- (7) - List length of old data record in double words.
- (8) - Current value of least recently used counter (LRU).
- (9) - Current data extent number.
- (10) - Data extent number where data is to be written.
- (11) - Path block sequence number.
- (12) - Version of BNAM

1 - Version prior to 3-11-71

2 - Version since 3-11-71

In addition to providing current status information, COMNEW is used to indicate requested information such as a new page number which would be placed in COMNEW(4).

2.7 Catalog Control Block

In order to read and write pages and data for the correct data set, it is necessary for BNAM to have available for the data set, specified information as to the data set reference number, the number of allowable records and other pertinent information. BNAM uses the routine JOPEN to access the catalog data sets described in Section B.2 and this information is provided to BNAM as requested on a data set by data set basis via this control block.

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.26
		Rev. 1	Date 6/7

CLOG - Catalog Control Block (22) I*2

- (1)-(4) - JOSHUA data set name.
- (5)-(6) - File category.
- (7) - Tree catalog reference number.
- (8) - Tree catalog base.
- (9) - Last used tree record.
- (10) - Last used cell within last used tree record.
- (11) - Last tree record allocated.
- (12) - Data storage reference number.
- (13) - Data storage base.
- (14) - Last used data record.
- (15) - Last data record allocated.
- (16) - Data extent number.
- (17) - Position in JDSCB data set.
- (18)-(20) - Reserved for JOPEN use.
- (21)-(22) - Not used.

2.8 System Definition Control Block

BNAM has been written such that the size can be varied by changing a few constants. These constants are contained in the System Definition Control Block (DIM). Two of these constants, (1) the number of in-core path-page blocks along with the associated control blocks, and (2) the base record for rolling paths to disk, can be set at run time. Setting the number of in-core path-page blocks allows the user to tailor BNAME to his particular data management problem. Setting the base record for rolling paths to disk provides one of the requirements for JOSHUA to run under MVT.

DIM - System Definition Control Block (16) I*2

- (1) - Cells per tree page - 49.
- (2) - Cells per path block - 46.
- (3) - Maximum qualifiers per named data record - 16.
- (4) - Number of pointers per cell - 4.

- (5) - Bytes per tree page - 784.
- (6) - Number of control blocks for paths on disk - 18.
- (7) - Number of in-core path-page blocks and associated control blocks - 10.
- (8) - Size of COMNEW - 12.
- (9) - Size of COMOLD - 16.
- (10) - Maximum number of path blocks that can be written to disk for one path - 8.
- (11) - Number of words per data block - 100.
- (12) - Base record for paths on disk - 0.
- (13) - Data set reference number for paths on disk - 61.
- (14) - Double words per tree page - 98.
- (15) - Number of catalog blocks allowed in ARRAY - 2.
- (16) - Not used.

2.9 Auxiliary Control Variables

The contents of the in-core ARRAY are managed on a dynamic basis with paths, pages, and catalog blocks being formed, rolled out, and rolled in on a continuing basis. The following four control variables assist this management:

BLOKOP - Space Management Request Option

- 1 - Get a block of core for a page; update LRU for all associated path blocks.
- 2 - Get a block of core for a catalog or work space; chain catalog blocks if necessary.
- 3 - Get a block of core for a path; chain path blocks if necessary; update LRU for all associated path blocks.

PTHOPT - Path Selection Mode

- 1 - Select path in core which best matches requested name string; if none in core, select path on disk and read into core; if none on disk, start new path.
- 2 - If path has more blocks in core, use path block; if no path blocks in core, select next path block from disk.

PAGEOP - Page-Path Retrieval Option

- 1 - Retrieve tree page.
- 2 - Retrieve path block.

MATOPT - Match Name Option

- 1 - Attempt to match name via path.
- 2 - Bypass attempt to match name via path.

CLOSOP - Close Option

- 1 - Close specified data set - do not free space.
- 2 - Close specified data set - free space.
- 3 - Close all modified data sets - do not free space.
- 4 - Close all modified data sets - free space.

2.10 Page Option Control Block

In order to get the PAGE routine to carry out its punitive functions, the PAGE Option Control Block is used.

OPTION - Page Option Control Block (4) I*2

(1) - Page Option

- 1 - Match name string.
- 2 - Traverse a subtree at a given level.
- 3 - Preorder traverse a subtree.

(2) - Brother Option

- 1 - Start with current brother.
- 2 - Start with next brother.

(3) - Not used for page option.

(4) - Valid Data Indicator

- 0 - Valid data record required; used with functional options 6 and 12.
- 1 - Valid data record not required; used with functional options 6, 11, and 12.

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.29
		Rev. 1	Date 6/73

2.11 Data Set - Category Control Block

This control block is read from disk during initialization and contains the data set-category pairs for protective handling control.

ALLCAT - Data Set - Category Control Block (10,2) I*2

(X,1) - Data set reference number

(X,2) - Data category

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.30
		Rev. 1	Date 6/73

(Blank)

3. BNAM Common Areas3.1 General

BNAM uses five named common areas to communicate between routines. They are:

- a. BNAMCM - principal communication area.
- b. COMNEW - starting values for Status Control Block.
- c. RNAME - Real*8 and Integer*2 constants.
- d. PROTEC - communication area for protective handling capability.
- e. STATIS - work area for accumulating data management statistics.

3.2 BNAMCM

<u>Variable</u>	<u>Type</u>	<u>Description</u>
DIM	Integer*2	System Definition Control Block. DIM(16).
TSTRIN	Real*8	Work area to contain desired name string. TSTRIN(18).
COMOLD (OLDCOM)	Integer*2	BNAM Communication Control Block to contain information passed to BNAM. COMOLD(16).
COMNEW (NEWCOM)	Integer*2	Status Control Block. COMNEW(12).
CLOG	Integer*2	Catalog Control Block. CLOG(22).
VSTRIN	Integer*2	Varying qualifier array. VSTRIN(18).
OPTION	Integer*2	Page Option Control Block. OPTION(4).
MATOPT	Integer*2	Match Name Option.
PTHOPT	Integer*2	Path Selection Mode.
SEPPTR	Integer*2	Contains last brother switch, son extent, and brother cell after call to UPKPTR or PKPTR. SEPPTR(3).
TEMP	Integer*2	Work area for cell pointers. TEMP(4).

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. 1	Page 8.3.32
		Rev. 1	Date 6/73

<u>Variable</u>	<u>Type</u>	<u>Description</u>
STORED (STORE)	Integer*2	Work area for trace array. STORED(4,16).
XYZ	Integer*2	Dummy
NOBYTE	Integer*4	Contains number of bytes in data record.

3.3 COMNEW

<u>Variable</u>	<u>Type</u>	<u>Description</u>
COMNEP	Integer*2	Starting values for the Status Control Block.

3.4 RNAME

<u>Variable</u>	<u>Type</u>	<u>Description</u>
STD	Real*8	'STD '
HEX5	Real*8	Z5555555555555555
BLANK	Real*8	' '
\$QUES\$	Real*8	'\$?\$ '
SCRATCH	Real*8	'SCRATCH '
TEMPLT	Real*8	'TEMPLATE'
DELETE	Real*8	'DELETED '
BROPT	Integer*2	Half word in cell for brother cell - 1.
BROPG	Integer*2	Half word in cell for brother page - 2.
SONPT	Integer*2	Half word in cell for son cell - 3.
SONPG	Integer*2	Half word in cell for son page - 4.
QUESTN	Integer*2	'? '
TZERO	Integer*2	Value of 0.

3.5 PROTEC

<u>Variable</u>	<u>Type</u>	<u>Description</u>
PRONAM	Real*8	Work area for reading a list of authorized names for each data set - category entry. PRONAM(120).
ALLCAT	Integer*2	Data Set-Category Control Block. ALLCAT(10,2).
NOPROT	Integer*2	Number of protected data set - category entries to be searched.

3.6 STATIS

<u>Variable</u>	<u>Type</u>	<u>Description</u>
STATS	Integer*2	Array for accumulating data management statistics. STATS(25).

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.34
		Rev. 1	Date 6/77

(Blank)

4. Program Logic

4.1 Overview

The JOSHUA data manager (BNAM) is composed of a series of routines which are described below and interrelated by the general flow chart in Figure 11.

<u>Routine</u>	<u>Purpose</u>
ACCUM	To accumulate data management statistics.
APPEND	To prefix the data set name to a name string as determined by the data set search sequence for retrieval and the designated output data set for update.
AUCHEK	To verify that a given user has authority to retrieve or write data of a particular data category.
BLOCK DATA	To initialize some of the variables in the named common areas BNAMCM, COMNEW, RNAME, PROTEC.
BNAMIN (\$BNAM) (BNAM)	To provide the interface between the data manager and the calling program, and to control the operation of the data manager.
DEPOIN	To delete a named data record, subtree, or data set from the JOSHUA data base; to rename a named data record or subtree in the JOSHUA data base.
GBLOK	To manage the dynamic in-core buffer space for paths, pages, and catalogs.
GPAGE	To make available the tree page requested by a calling routine.
GPATH	To select the previous in-core path which best matches the current name string and to request the retrieval of path blocks from disk when necessary.
JOPEN (JCLOSE)	To locate, retrieve, and update JOSHUA data set control blocks (JDSCB).
JTSEN	To provide an abnormal ending upon detection of an inconsistency in the data manager.
MATNAM	To match a name string by searching a path and/or tree pages.
PAGE	To provide BNAME with the ability to search tree pages to match a name string, to traverse the brothers subordinate to a name string, and to preorder traverse a given subtree.

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.36
		Rev. 1	Date 6/73

Routine

Purpose

PKPTR (UPKPTR)	To realign pointers from the first halfword of a tree cell.
PREAD (PWRITE)	To provide for record level protection under the JOSHUA System.
PTRACE	To search a path block to match a name string.
RDPOIN	To set error flags on a read, list, list brother, or rename request as applicable.
WRPOIN	To modify the tree pages as required to create or update a named data record.

4.2 Discussion of BNAM Logic

4.2.1 General

There exist three entry points for the JOSHUA data manager: BNAMIN for initialization of the number and addresses of the in-core path-page blocks, ARRAY, and the two control blocks, PGMAN8 and PGMAN2; \$BNAM for all version 1 requests; and ENAM for all version 2 requests.

Within the data manager, the entire retrieval procedure is carried out as if the BNAM entry were used, with appropriate interfacing provided for a version 1 request. This interfacing includes the setting up of the requested data set search sequence and providing default options to the BNAM Communication Control Block. The APPEND routine is called to prefix data set names according to the defined search sequence or the defined output data set. Once a data set name is prefixed, a request is made either to JOPEN to retrieve the catalog information in CLOG that pertains to the requested data set or to PWRITE to verify that the data can be written to the requested data category. A successful return from PWRITE also places the required catalog information in CLOG. A branch is then made to the section of coding that carries out the functional option as contained in COMOLD(1). Functional option 2 (READ) issues a call to MATNAM for the purpose of matching the requested name string by searching a path and/or tree pages. A call is then made to RDPOIN to set applicable error numbers. Functional option 3 (WRITE) begins with a call to MATNAM which functions as before. The routine WRPOIN is then called to update the tree pages and the Catalog Control Block in order to reflect the accomplishment of the WRITE request. The WRITE request either adds qualifiers to the tree, relocates the data pointer to a larger area, or changes the data

Figure 11. General Flowchart for BNAM.

length within a previously defined area. Functional option 4 (CLOSE) issues a call to JCLOSE to close all modified tree pages and catalog entries to disk and to retrieve data statistics for subsequent output. Functional option 5 (DELETE) calls MATNAM as before, with a subsequent call to DEPOIN for the placing of HEX5S in the tree to indicate deletion. Functional option 6 (LIST) issues an initial call to MATNAM as before with a subsequent call to PAGE which preorder traverses the subordinate subtree. Each LIST call returns one name string and its associated data pointer, with other name strings of the subtree returned on subsequent calls. On subsequent calls, MATNAM is bypassed and the location within the subtree is obtained from the STORE array passed from the calling program. This STORE array contains the pointers necessary to retrieve the subsequent name string. Functional option 7 (OPEN) calls JOPEN to bring in the JDSCB from disk. Functional option 10 (RENAME) is similar to DELETE except that a given qualifier is replaced with a rename qualifier instead of HEX5S. Functional option 11 (LISTB) issues a call to MATNAM as before with a subsequent call to PAGE to traverse the brothers subordinate to a given name string. The number of brother qualifiers returned on any one call is determined by the size of the BROTHR array passed to BNAM. Functional option 12 (LIST) only differs from functional option 6 (LIST) in that no data pointers are returned. Of the nine functional options, only READ, WRITE, and LIST (for data) return data pointers for subsequent data transfer. For these three options, a verification is made via PREAD, that the requestor is authorized to retrieve or update the data for which the data pointers are being returned. In all cases, the data manager epilog is executed and a return made to the calling program.

4.2.2 Features

In carrying out the job of data management, BNAM provides several features which enhance its capability and increase its efficiency. Two of these, the provision for paths of nodes traversed on previous retrievals and the use of data set extents, have been discussed. In this section several others will be mentioned.

In using data at a terminal, it is necessary to provide both a format record (template) and a data record in order to display the data. Since it is possible that both could come from the same user data set, it was decided to provide a path for both template and data from the same data set.

By the use of functional option 7, a user has a limited ability to look at the output from a job while the problem is still running.

As mentioned earlier, the maximum path length is approximately 322 nodes. However, in some ill-designed data sets, this limit is exceeded. In this case, BNAM ceases to lay in path nodes, marks the tag position to that effect, and continues to search through the tree pages for a match on the name string. This search is wasteful in the sense that beyond 322 nodes the path gives no assistance in retrieval and retrieval time increases. There are two solutions to the situation: (1) design the tree with a more balanced structure and (2) allow more space for in-core blocks. The current number of in-core blocks available for paths in the terminal environment is 7. In the batch environment, the default number is 8, but can be increased by the user at run time.

Record level protection is provided in BNAM via the use of the System Communication Control Block, the son data extent concept, and the PREAD and AUCHEK routines.

BNAM has the capability of searching for qualifiers in a less than normal manner. Given a name string, A.\$?\$.B, BNAM will search over all qualifiers at the second level as indicated by \$?\$ and try to match A and B, working in this manner under functional option 6 (LIST), functional option 11 (LISTB), or functional option 12 (LIST).

As mentioned in Volume I, DPSTM-500, record types make use of templates with one or more variable qualifiers in order to reduce the number of templates required for data display. BNAM handles the variable qualifier within the tree pages by always forcing the variable qualifier at any level of the tree to the end of the brother chain in question. Since a variable qualifier is only used when the actual qualifier is not present, the fact that the variable qualifier if it exists is always the last brother allows this match to be made at the only logical point.

Since the in-core space managed by BNAM dynamically contains path blocks and pages from many data sets, some means must be used to decide which in-core block will contain the next requested path block or page. This problem is resolved by using that in-core block that was least recently used (LRU).

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.40
		Rev. 1	Date 6/73

Paths are chained together with the higher LRUS reserved for the earlier blocks in the chain and the lower LRUS for the later blocks. This hierarchy is maintained continually so that a later path block is never left in core without its immediate predecessor. A series of algorithms for selecting the next block to use is given in the documentation for the GBLOK routine. These algorithms give preference to a path remaining in core at the expense of a tree page due to the fact that a path contains more pertinent information. As blocks are signaled by the LRU algorithms to be freed, the path or page contained in that block must be disposed of. If a tree page has been modified, it is written to disk; if a path is being removed from core, it is written to disk for possible later retrieval.

5. BNAM Errors

5.1 General

There are two classes of errors that are detected within BNAM: (1) those errors that are due to invalid user requests, and (2) those in which BNAM has lost control. The user errors are handled by setting an appropriate error number in COMTEM(6) and returning. The loss of control errors should never occur, but are checked in order to maintain the integrity of the data base in case they do. If these errors occur, the job is abended via a call to routine JTSEN.

5.2 Error Numbers for Invalid User Requests

<u>Error Number</u>	<u>Explanation</u>	<u>BNAM Routine</u>
2	On a read, write, or rename request, a name with only one qualifier was received.	BNAMIN
3	On a write request, the space allocated for the data set tree catalog was full.	WRPOIN
4	On a write request, the space allocated for data record storage was full.	WRPOIN
6	An attempt was made to use the reserved words CATALOG, WORK, or FREE as a data set name.	BNAMIN
7	The requested data set name could not be located for the last entry of the data set search sequence.	JOPEN
8	On a read, list, or list brother request, the data set has no entries.	BNAMIN
9	On a read, delete, list, rename, or list brother request, one or more of the qualifiers could not be located.	DEPOIN, RDPOIN
11	An attempt has been made to retrieve unauthorized data.	APPEND, PREAD
12	On a read, write, delete, list, rename, or list brother request, a name already exists containing fewer qualifiers than the given name.	DEPOIN, WRPOIN RDPOIN

<u>Error Number</u>	<u>Explanation</u>	<u>BNAM Routine</u>
13	On a read, write, list, or list brother request, a name already exists containing more qualifiers than the given name.	RDPOIN,WRPOIN
14	An invalid functional option was received.	BNAMIN
15	The number of bytes requested on a write exceeds 262136.	BNAMIN
21	On a write request, the length of the I/O list is negative.	BNAMIN
22	The number of qualifiers is out of range.	BNAMIN
24	An attempt has been made to create data without proper authorization.	PREAD
26	An inconsistency has been detected between the catalogs in the terminal region and those in the batch region.	BNAMIN,PAGE DEPOIN,WRPOIN
27	On a rename request, a rename qualifier with '?' as the first character is illegal.	BNAMIN,DEPOIN
29	The requested data category does not exist in the specified data set.	PREAD
30	On a write request, the data category for an existing record cannot be changed without an explicit override.	BNAMIN,WRPOIN
33	On a write, delete, or rename request, the user does not have authorization to update the specified data set.	APPEND
36	On a rename request, the rename qualifier already exists in the tree catalog and will give two identical tree roots for the specified data set.	BNAMIN
37	On a list or list brother request, the list parameters are out of range.	BNAMIN
38	The category of a protected template differs from that of the protected data.	PREAD

5.3 Loss of Control Abends (JTSEN)

<u>Explanation</u>	<u>BNAM Routine</u>
A special abend call has been made to BNAM.	BNAMIN
The computational mode cannot exist for the terminal environment.	BNAMIN
The maximum tree record allocated is less than the last tree record used or the maximum data record allocated is less than the last data record used.	BNAMIN
An inconsistency has been detected in the data management catalogs for the computational mode.	BNAMIN
The requested path block is for a path sequence number greater than the maximum allowed.	GBLOCK
The least recently used (LRU) counter has overflowed.	GBLOCK
The selected in-core buffer block is outside of the allowed range.	GBLOCK
An attempt has been made to free a path block with a forward pointer to a lower order path block.	GBLOCK
The data set name of the first path block differs from the corresponding control block name.	GBLOCK
A page and path are assigned to the same in-core block.	GBLOCK
The catalog name for the data set does not match the data set name.	GPAGE
For a Standard, User, or Scratch data set, (1) the data set name on the retrieved page does not agree with the data set name requested, or (2) the page requested exceeds the maximum page number.	GPAGE
The data set name of the first path block differs from the corresponding control block name.	GPATH
A varying qualifier is out of order.	PAGE

JOSHUA DPSTM-500	Title Basic Named Access Method (BNAM)	Vol. IX	Page B.3.44
		Rev. 1	Date 6/73

<u>Explanation</u>	<u>BNAM Routine</u>
The in-core data set name on the first extent of a data set differs from that on some other extent.	JOPEN
The data set name on a page to be written to disk differs from the corresponding control block name.	JOPEN
The node information (qualifier and/or pointers) in a page differs from that in a path when they should be the same.	WRPOIN
An error was detected in getting the JDSCB.	WRPOIN

6. BNAM Statistics

In order to give the user statistical information relative to his JOSHUA runs and to gain information for improving BNAM efficiency, a series of data management statistics are generated by BNAM for each job. Statistics are accumulated during each data management request and at job completion are logged to disk for later analysis. Four of these statistics, the read requests, the write requests, the disk accesses for data management overhead, and the disk accesses for data transfer, are provided to the user at the end of each JOSHUA run. In the following summary of data management statistics, the specific BNAM routine where each is accumulated is shown and all accesses are for disk I/O operations.

<u>Variable</u>	<u>Description</u>	<u>BNAM Routines</u>
STATS(1)	Read requests.	BNAMIN
STATS(2)	Requests for data set search for reads and subtree traversals.	BNAMIN
STATS(3)	Tree page accesses for read requests.	GPAGE
STATS(4)	Data transfer accesses for read requests.	BNAMIN
STATS(5)	Write requests.	BNAMIN
STATS(6)	Tree page accesses for write requests.	GPAGE
STATS(7)	Data transfer accesses for write requests.	GNAMIN
STATS(8)	Delete/rename requests.	BNAMIN
STATS(9)	Tree page accesses for delete/rename requests.	GPAGE
STATS(10)	Requests for vertical or horizontal subtree traversal.	BNAMIN
STATS(11)	Tree page accesses for subtree traversal.	GPAGE
STATS(12)	Name catalog and control block accesses to open JOSHUA data sets.	JOPEN

<u>Variable</u>	<u>Description</u>	<u>BNAM Routine</u>
STATS(13)	Total access for data management overhead. Sum of STATS(3), STATS(6), STATS(9), STATS(11), STATS(12), STATS(14), STATS(15), and STATS(20).	BNAMIN
STATS(14)	Accesses to roll path blocks from core.	GBLOK
STATS(15)	Accesses to read path blocks into core.	GPAGE
STATS(16)	Total data management errors.	BNAMIN
STATS(17)	Name catalog and control block accesses to open JOSHUA data sets for data sets that do not exist.	JOPEN
STATS(18)	Total data management requests.	BNAMIN
STATS(19)	Data transfer access for subtree traversals.	BNAMIN
STATS(20)	Accesses to close control blocks and tree pages to disk.	JOPEN
STATS(21)	Total data transfer accesses as estimated from catalog's record length. Sum of STATS(4), STATS(7), and STATS(19).	BNAMIN
STATS(22)	Not used.	-
STATS(23)	Records that are 1 block (400 bytes) in length.	BNAMIN
STATS(24)	Records that are 2-10 blocks in length.	BNAMIN
STATS(25)	Records that are greater than 10 blocks in length.	BNAMIN

Work Done By

R. C. Haywood
J. C. Jensen
F. D. Knight

Report Written By

R. C. Haywood
J. C. Jensen
F. D. Knight

Contents

	<u>Page</u>
1. Introduction	B.4.3
2. Modules	
2.1 ALLOCATE	B.4.5
2.2 SCRATCH	B.4.9
2.2.1 SCR\$\$\$	B.4.13
2.3 JCATALOG	B.4.19
2.4 CRAM	B.4.23
2.4.1 ACCUM	B.4.24
2.4.2 APPEND	B.4.26
2.4.3 AUCHEK	B.4.29
2.4.4 BLOCK DATA	B.4.31
2.4.5 BNAMIN(\$BNAM) (BNAM)	B.4.33
2.4.6 CRAM	B.4.39
2.4.7 DEPOIN	B.4.41
2.4.8 GBLOK	B.4.45
2.4.9 GPAGE	B.4.49
2.4.10 GPATH	B.4.53
2.4.11 JOPEN(JCLOSE)	B.4.58
2.4.12 JTSEN	B.4.66
2.4.13 MATNAM	B.4.68
2.4.14 PAGE	B.4.70
2.4.15 PKPTR(UPKPTR)	B.4.80
2.4.16 PREAD(PWRITE)	B.4.82
2.4.17 PTRACE	B.4.84
2.4.18 RDPOIN	B.4.89
2.4.19 WRPOIN	B.4.91

JOSHUA DPSTM-500	Title	Data Management Modules	Vol.	Page
			IX	B.4.2
			Rev.	Date
			1	6/73

(Blank)

1. Introduction

This section will describe the modules which compose the JOSHUA data management system. The subprograms contained in the module are described immediately following the module description unless they are of general use, in which case they are described in Section E.2, Subprograms. For arguments that have been discussed under BNAM Communication and Control Blocks, Section B.3.2, only the control block name will be given under the argument description. A flow chart is included for each routine where required.

JOSHUA DPSTM-500	Title Data Management Modules	Vol. IX	Page B.4.4
		Rev. 1	Date 6/73

(Blank)

Purpose: Allocate space within the JOSHUA data base for a JOSHUA data set.

Attribute: Nonreusable.

JOS Main Program: ALOC\$\$

Alternate Entry Point: SUJALL

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
<u>For Entry Point SUJALL:</u>		
CCB	Real*8	System Communication Control Block.
MASKS	Integer*4	Protection Mask.
TR	Integer*2	FT reference number for the tree catalog data set.
LTR	Integer*2	Number of tree catalog records requested.
DR	Integer*2	FT reference number for the data storage data set.
LDR	Integer*2	Number of data records requested.
E	Integer*2	Extent number.
DT	Integer*2	Data Set Type. 1-Standard, 2-User, 3-Job, 4-Scratch, 5-Edit.
EXPD	Integer*4	Expiration date in the form MMDDYY.
SER	Real*8	Up to thirty-two bytes of user data.
F	Integer*2	Flag to indicate the status of the data set on which data storage has been allocated. 0-online, 1-offline.
UERR	Integer*2	Error flag.

For Entry Point ALOC\$\$

CCB	Real*8	CCB
DSPACE	Integer*2	Number of 100-record blocks of data space requested.
MASK	Integer*4	Protection mask.
IERR	Integer*2	On entry: zero indicates new space requested; nonzero indicates space was previously allocated. On exit: an error flag.

JOSHUA DPSTM-500	Title Module: ALLOCATE	Vol. IX	Page B.4.6
		Rev. 1	Date 6/73

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
DTYPE	Integer*2	Same as DT above. When not equal four or five, DTYPE will be forced to three when program is invoked at this entry point.

Subprograms:

<u>Control Section</u>	<u>Entry Point</u>
IBMDAT - Subprogram Section	IBMDAT
IBMJUL - Subprogram Section	IBMJUL
JTJOSDEQ - Subprogram Section	JOSDEQ
JTJOSENQ - Subprogram Section	JOSENQ
JULDAT - Subprogram Section	JULDAT
LINK - Subprogram Section	JTREE
MOVEIT - Subprogram Section	MOVEIT

Data Sets:

JOSHUA data set name catalog (JDSNC)

JOSHUA data set control blocks (JDSCB)

JOSHUA catalog extension blocks (JCEB)

Data sets defined for tree catalog storage.

Data sets defined for data storage.

Program Logic:

- Entry point SUJALL is used to override certain predefined values which are established for entry point ALOC\$\$; it is also the only entry point thru which standard and user data sets may be allocated.
- An enqueue is issued on the catalogs.
- The JDSNC is examined to locate a free catalog block or the previous allocation of an area with the same name and extent number. When the latter is true and the disposition was specified as old, only the expiration date in the JCEB is updated; when disposition was not specified as old, the module immediately terminates with a duplicate entry error. If a free catalog entry is required but not available, termination with an error is effected.

JOSHUA

DPSTM-500

Title Module: ALLOCATE

Vol. IX	Page B.4.7
Rev. 1	Date 6/73

- The free space element records are read to locate space for data storage and the tree catalog. Space is allocated in the smallest accomodating block on the first data set on which space is available. Unavailability of space causes termination with an error condition indicated. The free space element records are adjusted to reflect the allocations.
- The JDSCB and JCEB records, whose positions were implied by the position of the data set name in the JDSNC, are completed.
- The enqueue is lifted before a return is issued.
- The data set reference numbers (FT numbers) of the data sets available for data and tree catalog storage are defined in data statements except for standard data sets which require these to be defined by argument.
- When allocation is made on a data set which is not normally on line, a data definition (DD) statement is placed in the JCEB record.
- The amount of space allocated for the tree catalog is determined by argument through entry point SUJALL and by data set type at ALOC\$\$ which establishes two records per each one hundred data records for job and scratch purposes and one record per data set for edit data sets.

Remarks:

Language:

Fortran IV

Author:

J. C. Jensen

JOSHUA

DPSTM-500

Title

Module: SCRATCH

Vol.
IX

Page
B.4.9

Rev.
1

Date
6/73

Purpose: Remove JOSHUA data sets from the data base or release unused space in job data sets.

Attribute: Nonreusable

JOS Main Program: MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
CCB	Real*8	System Communication Control Block.
NAME	Real*8	WCBARG1 - the second node must contain the indicator <s>. The first node contains a data set name or the key word ALL. The third node may contain the key word RLSE, EXPDT, JCATALOG, or HELPUS.

Subprograms:

<u>Control Section</u>	<u>Entry Point</u>
\$TERR - Subprogram Section	\$TERR
IBMDAT - Subprogram Section	IBMDAT
IBMJUL - Subprogram Section	IBMJUL
JOSDEQ - Subprogram Section	JTJOSDEQ
JOSENQ - Subprogram Section	JTJOSENQ
LINK - Subprogram Section	BNAM, JTREE
MDYJUL - Subprogram Section	MDYJUL
MOVEIT - Subprogram Section	MOVEIT
SCR\$\$\$	SCR\$\$\$
TRNSLT - Subprogram Section	TRNSLT

Data Sets:

JOSHUA data set name catalog (JDSNC)
JOSHUA data set control blocks (JDSCB)
JOSHUA catalog extension blocks (JCEB)

JOSHUA DPSTM-500	Title Module: SCRATCH	Vol. IX	Page B.4.10
		Rev. 1	Date 6/73

Program Logic:

- The module offers the following five options:
 1. Release unused data set space (SCROPT=1).
 2. Scratch a data set unconditionally (SCROPT=3).
 3. Scratch all expired or empty job or edit data sets (SCROPT=4).
 4. Scratch all job or edit data sets for the user (SCROPT=5).
 5. Special purge - attempts to scratch a designated data set despite a full free space element list (SCROPT=6).
- NAME (2) must contain the symbol <s>.
- The default option is to unconditionally scratch (SCROPT=3) the specified data set. Alternatives are specified by argument in NAME as follows:

SCROPT=1 when NAME (3) contains the key word RLSE;

SCROPT=4 when NAME (3) contains the key word EXPDT or JCATALOG;

SCROPT=5 when NAME (1) contains the key word ALL;

SCROPT-6 when NAME (3) contains the key word HELPUS.
- Requests are subject to certain conditions before being passed to SCR\$\$\$ for discharge.
 1. When a special purge (SCROPT=6) is indicated, the write data set must be 9999.
 2. When SCROPT is one, three or six, NAME (1) specifies the data set to be affected. Unless the write data set is 9999 the data set specified must be a job or edit data set.
- Subroutine SCR\$\$\$ is called to satisfy the request.
- A call is made to BNAM (functional request 7) to update the in-core catalogs.

JOSHUA

DPSTM-500

Title

Module: SCRATCH

Vol.

IX

Page

B.4.11

Rev.

1

Date

6/73

Terminating Conditions:

\$TERR is called when the following conditions are encountered:

- a. NAME (2) does not contain the symbol <s>.
- b. The write data set is 9999 but the user is not verified.
- c. Error return from subroutine SCR\$\$\$. The BNAM call is made before terminating.

Language:

Fortran IV

Author:

J. C. Jensen - R. C. Haywood

JOSHUA

DPSTM-500

Title

Module: SCRATCH
Subroutine: SCR\$\$\$

Vol. IX

Page B.4.13

Rev. 1

Date 6/73

Purpose: Remove JOSHUA data sets from the data base or release unused space in a data set.

Usage: Call SCR\$\$\$ (CCB, DSNAME, SCROPT, IERR)

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
CCB	Real*8	CCB.
DSNAME	Real*8	Defines for the appropriate options the name of the data set to be affected.
SCROPT	Integer*2	Subroutine options: <ol style="list-style-type: none">1. Release unused space in data set DSNAME.2. Scratch data set DSNAME if its expiration date has been reached.3. Scratch data set DSNAME.4. Scratch all expired or empty data sets.5. Scratch all job or edit data sets for the user.6. Special purge - equivalent to option 3 but will attempt to scratch the data set despite the FSE record being full.
IERR	Integer*2	Error Flag.

Subprograms:

IBMDAT - Subprogram Section
IBMJUL - Subprogram Section
JOSDEQ - Subprogram Section
JOSENQ - Subprogram Section
JTREE - Subprogram Section
MDYJUL - Subprogram Section
MOVEIT - Subprogram Section

Data Sets:

JOSHUA data set name catalog (JDSNC)
JOSHUA data set control blocks (JDSCB)
JOSHUA extension blocks (JCEB)
All data sets referenced by the JDSCB.

JOSHUA DPSTM-500	Title Module: SCRATCH Subroutine: SCR\$\$\$	Vol. IX	Page B.4.14
		Rev. 1	Date 6/73

Program Logic:

- If DSNAME specifies FREE a return is issued (without error).
- An enqueue is issued on the catalogs and is lifted after all alterations to the catalogs have been made.
- Each entry in the JDSNC is checked. Entries containing the key word FREE are bypassed immediately. When SCROPT is neither four nor five, only those data sets specified by DSNAME are selected; when SCROPT is five the low-order four positions of the data set name must be equal to the user's number. The data set name is inconsequential when SCROPT equals four - all are selected.
- JDSCB and JCEB records associated with a selected data set are read. The data set name stored in each must match the objective data set name or an error return is taken.
- When in the batch, the protection category of the data set must be zero or equal the signon flag.
- When SCROPT is four or five the JCEB record is checked and if a job or edit data set is not indicated, the data set is not scratched.
- When SCROPT is two or four the expiration date in the JCEB is compared to the current date. If the date has not been reached, the data set is not scratched unless, when SCROPT equals four, the data set contains no entries.
- An enqueue is attempted on the objective data set unless it is the write data set and verification has been made. An error return is taken should the enqueue fail unless SCROPT is four or five in which case the status of the data set is left unchanged. The enqueue is lifted after action on the data set is completed.
- The Free Space Element (FSE) list for the physical data sets containing the objective JOSHUA tree and data records is read. (If a data set is not on-line, an error number is set in IERR and processing continued with the next JDSNC entry. Unless another error occurs, this will be the setting of IERR on return). Unless SCROPT is equal to six, an error return is taken if an FSE list is full. The FSE lists are updated. The amount of space being released is added to an already free block immediately preceding or following the released block or a new entry is inserted at the appropriate position in the list. If the final number of entries exceeds the maximum allowed, an error return is taken.
- If SCROPT equals one, the JDSCB is updated to reflect the new amount of space; for all other options the data set name is replaced in the JDSNC by the key word FREE.

JOSHUA

DPSTM-500

Title

Module: SCRATCH
Subroutine: SCR\$\$\$

Vol.
IX

Page
B.4.15

Rev.
1

Date
6/73

Language:

Fortran IV

Author:

J. C. Jensen - R. C. Haywood

JOSHUA

DPSTM-500

Title Module: SCRATCH
 Subroutine: SCR\$\$\$

Vol.
IX

Page
B.4.18

Rev.
1

Date
6/73

(Blank)

JOSHUA

DPSTM-500

Title

Module: JCATALOG

Vol.

IX

Page

B.4.19

Rev.

1

Date

6/73

Purpose: Allocate data set space within the JOSHUA data base. List data from JDSCB, JCEB and FSE records.

Attribute: Nonreusable

JOS Main Program: MAIN

Arguments:

<u>Mnemonic</u>	<u>Type</u>	<u>Description</u>
\$MCB	Real*8	System Communication Control Block. Argument is generated by the precompiler.

Subprograms:¹

<u>Control Section</u>	<u>Entry Point</u>
JTJOSDEQ - Subprogram Section	JOSDEQ
JTJOSENQ - Subprogram Section	JOSENQ
LINK - Subprogram Section	JTREE

External References:

FRECHEK
SCRATCH
SUJALL (see ALLOCATE)

Data Sets:²

JOSHUA data set name catalog (JDSNC)
JOSHUA data set control blocks (JDSCB)
JOSHUAL catalog extension blocks (JCEB)
FT05F001 - SYSIN
FT06F001 - SYSOUT

-
1. Excludes references developed by either the FORTRAN compiler or the JOSHUA precompiler.
 2. Excludes any data sets referenced by modules listed under External References.

JOSHUA DPSTM-500	Title Module: JCATALOG	Vol. IX	Page B.4.20
		Rev. 1	Date 6/73

Program Logic:

- The modules FRECHEK and SCRATCH are executed.
- FT05F001 is read. If any data is present it is assumed to represent the input for a data set allocation (see figure M). For each record, the parameters to the allocate routine are filled in and ALLOCATE executed.
- The edit portion of the module reads the entire name catalog. For each position containing a data set name (viz., it does not contain the word FREE), the corresponding data set control block is read and then written to the SYSOUT data set.

The associated JDSCB record is read and the number of records allocated from each physical data set is totaled.

- A list of the catalog extension blocks is produced.
- The FSE records for each of the physical data sets in the data base are read and written to the SYSOUT data set. The reference numbers for these data sets are defined in the code. The amount of free space in each data set is determined.

The final portion computes the total number of records accumulated for each data set and writes the number of allocated records, the number of free records, and the total number of records for each data set to the SYSOUT data set.

Remarks:

1. The JCATALOG module may be used to obtain the described listings without data set allocation by not using any data cards.
2. JCATALOG is a JOSHUA precompiled module.

Language:

Fortran IV

Author:

J. C. Jensen

JOSHUA

DPSTM-500

Title

Module: JCATALOG

Vol. IX

Page
B.4.21

Rev. 1

Date
6/73

<u>Card Columns</u>	<u>Mnemonics</u>	<u>Format</u>	<u>Description</u>
2 - 9	DNAME	A8	Data set name
11 - 18	MASKS	Z8	Security mask
19 - 22	TR	I4	Tree catalog data set reference number
23 - 27	LTR	I5	Number of required tree records
28 - 31	DR	I4	Data storage data set reference number
32 - 36	LDR	I5	Number of required data records
37 - 38	E	I2	Data extent number
39 - 40	DT	I2	Data set type; 1-Standard, 2-User, 3-Job
41 - 46	EXPD	I6	Expiration date(mo/day/yr)
47 - 78	RES	4A8	User comments
79 - 80	F	I2	Status flag; 0-online, 1-offline

Figure 12. JCATALOG Input Data Card

JOSHUA

DPSTM-500

Title

Module: CRAM

Vol.

IX

Page

B.4.23

Rev.

1

Date

6/73

Purpose: To provide a direct access I/O handler (JTREE) and the capability of handling name data records (BNAM).

Attribute: Serially reusable.

Entry: CRAM
BNAMIN
\$BNAM
BNAM
JTREE

Component Programs: CRAM
BNAMIN
JTREE

Objective: To provide -

Linkage between any dynamically loaded JOS Module and any entry point within this module (see subroutine 'CRAM');

Data management (see 'BNAMIN');

A direct access organization input/output handler usable directly within this module or dynamically by any JOS Module (see subroutine 'JTREE' in Section D.2).

Language: Assembler, Fortran IV

Authors: F. D. Knight, R. L. Boyce

JOSHUA DPSTM-500	Title Subroutine: ACCUM	Vol. IX	Page B.4.24
		Rev. 1	Date 6/73

Purpose: To accumulate data management statistics.

Usage: CALL ACCUM(CCB,STATS,OPTION), where

CCB $\equiv$ System Communication Control Block

STATS $\equiv$ BNAM Statistics

OPTION $\equiv$ Option flag - ZERO - Normal

ONE - Return Statistics

External References: GETMAIN, FREEMAIN

Program Logic: The addresses of the arguments are received via the argument list. The computational flag (\$CBMONF) is checked and if it is computational (execution of a member is implied), the twenty-five half-word statistics are accumulated and stored in the BNAM statistics table (BST). The return statistics option is then checked and if the statistics are desired, the twenty-five execute member statistics are added to the twenty-five "other" and stored in the statistics fields (STATS) before returning; otherwise, a return is issued.

If the computational mode had not been indicated, the twenty-five statistics fields are accumulated in another area; statistics are set up by function category; and the return statistics option is checked as before.

Referenced by: BNAM

Language: Assembler

Author: W. H. Brotherton, Jr.

JOSHUA DPSTM-500	Title Subroutine: APPEND	Vol. IX	Page B.4.26
		Rev. 1	Date 6/73

Purpose: To prefix the data set name to a name string as determined by the data set search sequence for retrieval and the designated output data set for update.

Usage: Call APPEND(CCB,STRING,J)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block
STRING(1)	Real*8	Name string for desired record.
J	Integer*2	Position pointer for data set search sequence, CCB(J). 10 - First data set 11 - Second data set or blank 12 - Third data set or blank 13 - Fourth data set or blank.

Common Areas: BNAMCM - Section B.3.3.2
RNAME - Section B.3.3.4

Program Logic:

The program checks to determine if the data set search sequence is exhausted in which case a return is issued. If in the batch environment and SCRATCH is the first qualifier in the name string, the dynamic scratch data set name, **HHMMSS, replaces SCRATCH as the first qualifier and no prefixing takes place. If the "Write data set" is '9999' in the functional mode or if the append flag is set, then again no prefixing takes place. If the request is for other than a write, rename, or delete, the data set name as given by CCB(J) is prefixed to the name string unless 'STD' is contained in CCB(J) in which case no prefixing takes place. If a write, rename, or delete is requested and one of the previously stated conditions is not satisfied, the Job data set name is prefixed for the computational mode and the "Write" data set name for the functional mode.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language: Fortran IV

Author: F. D. Knight

JOSHUA DPSTM-500	Title Subroutine: AUCHEK	Vol. IX	Page B.4.29
		Rev. 1	Date 6/73

Purpose: To verify that a given user can retrieve or write data of a particular data category.

Usage: CALL AUCHEK(PMCB(3),I,PRONAM,DUMMY,&N) where,
 PMCB(3) $\equiv$ Two flag bytes
 I $\equiv$ Relative bit of byte to be masked
 PRONAM $\equiv$ Badge area
 DUMMY $\equiv$ Badge for authorization check
 &N $\equiv$ Statement for return 1 control

External References: CALL, JTREE

Data Sets: Data set defined for data set-category and authorization tables.

Program Logic: The addresses of the arguments are received via the argument list (register one). The bit to be masked is set up and masked against the flag position. If the bit is on, the data set has been read before so the next byte is masked to see if the authorization flag has been set. If the authorization flag is on, a return one is set up (indicates user can retrieve or write data); otherwise a return is issued.

If the first masking had indicated that the record has not been read, a Call to JTREE is issued to read the record. A search is then made to see if the badge (DUMMY) is authorized. If the badge is found, access is okayed via setting the bit and taking a return one option; otherwise, a return is issued.

Referenced by: BNAM

Language: Assembler.

Author: W. H. Brotherton, Jr.

Purpose:

To initialize some of the variables in the named common areas BNAMCM, COMNEW, RNAME, PROTEC.

BNAMCM

COMNEW

<u>Variable</u>	<u>Value</u>
DIM(1)	49
DIM(2)	46
DIM(3)	16
DIM(4)	4
DIM(5)	784
DIM(6)	18
DIM(7)	10
DIM(8)	12
DIM(9)	16
DIM(10)	8
DIM(11)	50
DIM(12)	0
DIM(13)	61
DIM(14)	98
DIM(15)	2
CLOG(1)-CLOG(4)	blank
CLOG(5)-CLOG(22)	0

<u>Variable</u>	<u>Value</u>
COMNEP(1)	1
COMNEP(2)	0
COMNEP(3)	2
COMNEP(4)	1
COMNEP(5)	0
COMNEP(6)	1
COMNEP(7)	0
COMNEP(8)	0
COMNEP(9)	1
COMNEP(10)	0
COMNEP(11)	1
COMNEP(12)	0

JOSHUA

DPSTM-500

Title Block Data Routine

Vol. IX Page
B.4.32

Rev. 1 Date
6/73

RNAME

<u>Variable</u>	<u>Value</u>
STD	'STD '
HEX5	Z5555555555555555
BLANK	' '
\$QUES\$	'\$?\$ '
SCRATCH	'SCRATCH '
TEMPLT	'TEMPLATE'
DELETE	'DELETED '
BROPT	1
BROPG	2
SONPT	3
SONPG	4
QUESTN	'? '
TZERO	0

PROTEC

<u>Variable</u>	<u>Value</u>
PRONAM(1)-PRONAM(120)	blank
ALLCAT(1,1)-ALLCAT(10,2)	0

Purpose: To provide the interface between the data manager and the calling program and, to control the operation of the data manager.

Usage: CALL BNAMIN (ARRAY,PGMAN8,PGMAN2)

or

CALL \$BNAM (CCB,COMTEM,STRING,STORE)

or

CALL BNAM (CCB,COMTEM,STRING,STORE,BROTHR,FQUAL)

Arguments:

Parameter	Type	Description
ARRAY (98,1)	Real*8	In-core path - page blocks
PGMAN8 (2,1)	Real*8	DSNAME Control Block
PGMAN2 (8,1)	Integer*2	Space Management Control Block
CCB (1)	Real*8	System Communication Control Block
COMTEM (1)	Integer*4	Communication array between the data manager and the calling program.
STRING (1)	Real*8	Name string for desired record.
STORE (1)	Integer*2	Trace array for subsequent calls for list and list brother.
BROTHR (1)	Real*8	Array for returning requested brothers.
FQUAL (1)	Integer*2	Array for subsequent calls for list and list brother when using the varying qualifier.

Common Areas:

BNAMCM - Section B.3.3.2

COMNEW - Section B.3.3.3

RNAME - Section B.3.3.4

PROTEC - Section B.3.3.5

STATIS - Section B.3.3.6

JOSHUA DPSTM-500	Title Subroutine: BNAMIN	Vol.	Page
		IX	B.4.34
		Rev.	Date
		1	6/73

Subprograms Used:

ACCUM

APPEND

DEPOIN

JOPEN(JCLOSE)

JTREE - Subprogram Section

JTSEN

MATNAM

MOVEIT - Subprogram Section

PAGE

PREAD(PWRITE)

RDPOIN

WRPOIN

Data Sets: Data set defined for data set-category and authorization tables.

Program Logic:

There exist three entry points to this routine. The first call is to BNAMIN to initialize the addresses of the in-core path - page blocks, the DSNAM Control Block, and the Space Management Control Block. In addition, the path base for disk storage and the protective handling array are initialized. The other two entry points are used to retrieve named data records (\$BNAM for Version 1 and BNAM for Version 2). The input arguments are checked for validity and the data sets are searched according to the established data set search sequence. For each data set the routine APPEND is called to prefix the appropriate data set name. The data set Catalog Control Block is returned via a call to JOPEN or PWRITE based on the functional option requested. A branch is then made to the appropriate functional section, which usually consists of a call to MATNAM for matching the given string of qualifiers and a call to _POIN to set flags and/or perform the requested operation. The resulting data set reference number and record number are returned and statistics of retrieval are accumulated.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language: Fortran IV

Author: F. D. Knight

JOSHUA

DPSTM-500

Title

Subroutine: CRAM

Vol.

IX

Page
B.4.39

Rev.

1

Date
6/73

Purpose:

To provide a list of pointers to all entry points within this module.

Program Logic:

The pointer to the entry point address list is set up in general purpose register 1. A return is then issued.

Referenced By:

JOS interface modules

Language:

Assembler

Author:

R. L. Boyce, Jr.

JOSHUA DPSTM-500	Title Subroutine: CRAM	Vol. IX	Page B.4.40
		Rev. 1	Date 6/73

JOSHUA

DPSTM-500

Title

Subroutine: DEPOIN

Vol.

IX

Page

B.4.41

Rev.

1

Date

6/73

Purpose:

To delete a named data record, subtree, or data set from the JOSHUA data base; to rename a named data record or subtree in the JOSHUA data base.

Usage:

Call DEPOIN(CCB,PGMAN8,PGMAN2,PTHPTS,TRACE,NAMES,TAG,OPT)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.
PTHPTS(4,1)	Integer*2	Pointer portion of a path block. Address of PTHPTS in first ARRAY block is passed. BNAM indexes for its own use.
TRACE(1)	Real*8	Names portion of a path block. Address of TRACE in first ARRAY block is passed. BNAM indexes for its own use.
NAMES(1)	Real*8	Names portion of a page block. Address of NAMES in first ARRAY block is passed. BNAM indexes for its own use.
TAG(1)	Logical*1	Tag portion of a path block. Address of TAG in first ARRAY block is passed. BNAM indexes for its own use.
OPT	Integer*2	Delete/Rename option 1 - Delete 2 - Rename

Common Areas:

BNAMCM - Section B.3.3.2

RNAME - Section B.3.3.4

Subprograms Used:

GPAGE

JOPEN

Program Logic:

This routine performs the delete and rename functions of BNAM. These two operations are basically the same operation with the delete being the special case of rename where the new qualifier is HEX5S. If a delete request is received for a name string with one qualifier, this is interpreted to mean that the deletion of the entire data set is desired. This is accomplished by resetting the Catalog Control Block (CLOG) to indicate an empty data set. All paths and pages related to

JOSHUA DPSTM-500	Title Subroutine: DEPOIN	Vol. IX	Page B.4.42
		Rev. 1	Date 6/73

this data set are deleted from the PGMAN8 and PGMAN2 control blocks. If the request is for the delete of a name string of more than one qualifier or for a rename, the logic of the 2 requests is the same. An error check is made to determine if the number of qualifiers requested equals the number matched. If not equal, appropriate error numbers are set. If the number of qualifiers is equal, then the qualifier in both the path cell and associated tree cell are overwritten with HEX5S for a delete or with the rename qualifier for a rename. Any in-core paths for the data set are modified and out-of-core paths which are of a different type are deleted. The program insures that a variable qualifier cannot be renamed and that a rename qualifier cannot be variable.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language: Fortran IV

Author: F. D. Knight

JOSHUA

DPSTM-500

Title Subroutine: GBLOK

Vol. IX	Page B.4.45
Rev. 1	Date 6/73

Purpose: To manage the dynamic in-core space for paths, pages, and catalogs.

Usage: Call GBLOK(CCB,ARRAY,PGMAN8,PGMAN2,BLOKOP)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block.
ARRAY(98,1)	Real*8	In-core path-page blocks.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.
BLOKOP	Integer*2	Space Management Request Option.

Common Area: BNAMCM - Section B.3.3.2

RNAME - Section B.3.3.4

STATIS - Section B.3.3.6

Subprograms Used: JCLOSE
JTREE - Subprogram Section
JTSEN

Data Sets: Data set defined for out-of-core paths.

Program Logic: GBLOK manages the dynamic storage space for paths, pages and catalogs. Whenever a new page must be brought into core, more catalog space obtained, or space for a new path block needed, GBLOK carries out the function of getting the required space by rolling out pages and paths to disk. GBLOK bases its decision of which block to free for the next use on the concept of a least recently used counter or LRU. Each time a request for space is made, GBLOK searches the LRU (PGMAN2(3,I)) of each in-core block and selects (with exceptions) that block with the lowest LRU. If any LRU is zero, that block is immediately selected. Since paths are considered to contain more information than a page, paths are kept in core for some delta time even though their LRU may be less than that of a page. This condition is subordinate to keeping at least one page in core. Once a block is selected to be freed, the disposition of the path (PGMAN2(2,I)=0) or page (PGMAN2(2,I)=1) contained therein must be made. A catalog block once selected is marked in PGMAN2(2,I) with a 2 and is never freed. If a page block is being freed and has not been modified, the block is initialized. If the page has been

JOSHUA DPSTM-500	Title Subroutine: GBLOK	Vol. IX	Page B.4.46
		Rev. 1	Date 6/73

modified, it is updated on disk via a call to JCLOSE before initialization. If a path block is being freed, and it is the only path of this type in core for this data set, the path block must be relocated to disk. If the path block has a sequence number greater than 8, it is lost and control passes to an in-core path termination section. When it is decided that a path block must be relocated to disk, a space in PGMAN2 for the out-of-core paths must be found. If this path type for the same data set already exists, the path block is written to disk by JTREE, with appropriate updating of the PGMAN8 and PGMAN2 control blocks. If space does not exist, the LRU concept is again used to locate a place for the path block. When space has been found for the new page, path, or catalog block, the specific entity determines what is done next. In all cases, the position of the block is returned in COMNEW. If a page block is requested, the page LRU is set and if there exists an associated path, all path block LRUS are updated, always keeping the highest LRUS on the earlier path blocks. If a catalog or path block is requested, this block is chained with forward and backward pointers to any associated blocks. If it is a catalog block request, a return is given after placing CATALOG in PGMAN8. If a path is requested, all associated path LRUS are updated as before. If a request is made for work space, it is only used for the specific request and is available thereafter.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language: Fortran IV

Author: F. D. Knight

Purpose: To make available the tree page requested by a calling routine.

Usage: Call GPAGE(CCB,ARRAY,PGMAN8,PGMAN2,PAGEOP,*)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block
ARRAY(98,1)	Real*8	In-core path-page blocks.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.
PAGEOP	Integer*2	Page-Path Retrieval Option.
*		Return 1 - For a Job data set, a catalog inconsistency was detected.

Common Area: BNAMCM - Section B.3.3.2

RNAME - Section B.3.3.4

STATIS - Section B.3.3.6

Subprograms Used: GBLOK

JTREE - Subprogram Section

JTSEN

Data Sets: Data set defined for out-of-core paths.

Data sets defined for JOSHUA tree catalogs.

Program Logic: GPAGE may be called either to request that a tree page or path block be made available. If the tree page is requested, an in-core search is made to determine if the page is present in ARRAY. If the page is already in core, its block number is set, the LRU is updated and a return is issued. If the tree page is not in core, GBLOK is called to reserve a block of space for the page. The record number and data set reference number are obtained from the Catalog Control Block (CLOG). Checks are made prior to calling JTREE to verify that the tree page and cell requested are valid. A call is then given to JTREE to read the page into the block obtained by GBLOK. After the return from JTREE, the PGMAN2 array is updated. A check is made that the DSNAME in cell 1 of the retrieved page agrees with the DSNAME in the name string. If any of the checks fail in GPAGE, the data manager terminates. If a path is requested, a call is given to GBLOK to get space for the path. A call is then given to JTREE to read the path

JOSHUA DPSTM-500	Title Subroutine: GPAGE	Vol. IX	Page B.4.50
		Rev. 1	Date 6/73

block into core, followed by a return to the calling routine.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language: Fortran IV

Author: F. D. Knight

Purpose:

To select the previous in-core path which best matches the current name string and to request the retrieval of path blocks from disk when necessary.

Usage:

Call GPATH(CCB,ARRAY,PGMAN8,PGMAN2,TAG,TRACE,*)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block.
ARRAY(98,1)	Real*8	In-core path-page blocks.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.
TAG(1)	Logical*1	Tag portion of a path block. Address of TAG in first ARRAY block is passed. BNAM indexes for its own use.
TRACE(1)	Real*8	Names portion of a path block. Address of TRACE in first ARRAY block is passed. BNAM indexes for its own use.

*

Return 1 - No previous path block.

Common Area:

BNAMCM - Section B.3.3.2

RNAME - Section B.3.3.4

Subprograms Used:

GBLOK

GPAGE

JTSEN

MOVEIT - Subprogram Section

Program Logic:

There are two options in GPATH determined by the Path Selection Mode, PTHOPT. When PTHOPT is 1, it is an initial request to find the existing in-core path which best matches the name string. When PTHOPT is 2, the path has already been selected, only the next path block is desired. In selecting the path with PTHOPT equal 1, it is determined whether a template or non-template path is desired. Then the in-core blocks of ARRAY are searched to determine which path best matches the name string. If no suitable paths exist in-core, the out-of-core paths are checked to determine if one exists for the current data set and the desired type. If no suitable path exists on disk, a return 1 is issued to MATNAM which indicates paging is necessary. If the path is found in core, the best path is selected. If the path exists on disk,

JOSHUA DPSTM-500	Title Subroutine: GPATH	Vol. IX	Page B.4.54
		Rev. 1	Date 6/73

the first path block is read in by a call to GPAGE. If PTHOPT is 2, the in-core ARRAY is checked for the next path block. If not in core, it is read from disk by GPAGE.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language: Fortran IV

Author: F. D. Knight

JOSHUA DPSTM-500	Title Subroutine: JOPEN	Vol. IX	Page B.4.58
		Rev. 1	Date 6/73

Purpose: To locate, retrieve, and update JOSHUA Data Set Control Blocks (JDSCB).

Usage: Call JOPEN(CCB,ARRAY,PGMAN8,PGMAN2)
or
Call JCLOSE(CCB,TEMP,CLOSOP)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block.
ARRAY(98,1)	Real*8	In-core path-page blocks.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.
TEMP	Real*8	Name of data set to be closed.
CLOSOP	Integer*2	Close Option.

Common Area: BNAMCM - Section B.3.3.2
STATIS - Section B.3.3.3

Subprograms Used: GBLOK
JTREE - Subprogram Section
JTSEN
MOVEIT - Subprogram Section

Data Sets: JOSHUA Data Set Name Catalog(JDSNC)
JOSHUA Data Set Control Blocks (JDSCB)
Data sets defined for JOSHUA tree catalogs.

Program Logic: The JOPEN entry in this subroutine is used to retrieve a specified JDSCB when required by other BNAM subroutines. The data set name is received in the first double word of the TSTRIN array and the catalog data is returned in the Catalog Control Block, CLOG. In order to reduce the overhead involved with control block retrieval, one or more blocks of BNAM dynamic storage is obtained to retain a number of control blocks in-core (See Figure 12 for in-core JDSCB format). Each block of storage may contain 19 JDSCB entries of 40 bytes each. If the previously returned CLOG array has been modified (the 18th half word is nonzero), the corresponding in-core JDSCB entry is updated before the open operation is performed.

	8 bytes			
1	DSNAME			
9	FC		TR	TB
17	TL*	TN*	TS*	DR
25	DB	DL	DS	EX
33	CR	MF	BP	CP

*The values of MF, BP, and CP for the first data set extent entry are stored in these positions for extents other than the first.

DSNAME** - JOSHUA data set name

FC - File Category

TR - Tree catalog reference number

TB - Tree catalog base (1st record number - 1)

TL - Number of tree records used

TN - Number of cells used in the last used tree record

TS - Number of allocated tree records

DR - Data storage reference number

DB - Data storage base (1st record number - 1)

DL - Number of data records used

DS - Number of allocated data records

EX - Data extent

CR - Physical record number of entry in JDSCB data set

MF - Modification flag (nonzero indicates the entry has been modified)

BP - Block pointer (contains the block number of BNAM dynamic storage used for in-core JDSCB entries)

CP - Catalog pointer (contains the starting double word position of the in-core JDSCB entry within the block of BNAM dynamic storage allocated to in-core JDSCB entries)

** DSNAME is stored as an EBCDIC value. All other variables are binary integers.

Figure 12. In-Core JDSCB Entry

JOSHUA DPSTM-500	Title Subroutine: JOPEN	Vol. IX	Page B.4.60
		Rev. 1	Date 6/73

The JCLOSE entry is used to update the control block data set for JDSCB entries that have been modified. The operation is performed when a specific CLOSE call is received by BNAM, or, when any of the BNAM subroutines, including JOPEN, need to free space to be used for other purposes. Each time a JDSCB is written to disk, any tree pages of the corresponding JOSHUA data set that are core resident and have been modified are also written. Note that all JDSCB entries and tree pages are updated on disk only from this subroutine. The following paragraphs describe the subroutine in more detail for both the JOPEN and JCLOSE entries.

JOPEN

The modification flag for the previous CLOG entry is checked. If the previous entry had been modified, the in-core JDSCB is updated. (The data set name is in the 1st 8 bytes of the CLOG array and the in-core JDSCB position is determined from the BP and CP values in the CLOG array). If the modified entry is not the first extent, the first extent entry is also updated. Note that all tree catalog pointers are kept in the 1st extent entry only. Other extent entries point to the first extent by using the TN and TS positions of the in-core JDSCB entry. If the current request is for the same JDSCB, the subroutine returns.

The in-core JDSCB entries are then searched to locate an entry corresponding to the requested data set name, TSTRIN(1), and extent number, COMNEW(9). If the entry is located in core, it is moved to the CLOG array and the subroutine returns.

If the entry is not located, a position for it must be determined within an in-core catalog block. This position is determined as follows:

- 1 - The next sequential position within the existing catalog blocks is used except when they have been marked as full. Since this position may already be used for another entry, the modification flag is checked. If the entry has been modified, control is transferred to the CLOSE section of the subroutine prior to storing the new entry.
- 2 - If the existing catalog blocks are marked as full, then the first position marked as FREE will be used.
- 3 - If no position is marked as FREE, an additional catalog block may be obtained unless the maximum number of blocks already exist. If an additional catalog block is obtained, the next sequential position is used.
- 4 - If the maximum number of catalog blocks exist and they have been marked as full, the positional pointer for all catalog blocks is reset to the first position and this position in the first catalog block is used. The catalog block is thus no longer marked as full.

JOSHUA

DPSTM-500

Title

Subroutine: JOPEN

Vol.

IX

Page

B.4.61

Rev.

1

Date

6/73

When the catalog block position has been established, the JOSHUA data set name catalog, JDSNC, and the corresponding JDSCB data set are searched to locate the correct data set extent. If it cannot be located, an error flag, ERROR = 7, is set and the subroutine returns. If found, the BP and CP values are stored in the in-core entry, the entry is moved to the CLOG array, the current CP position, PGMAN2(8,BP), is updated to point to the next entry, and the subroutine returns.

JCLOSE

The JCLOSE entry is used to update data set control blocks and tree pages existing on direct access storage, and to free space used for in-core JDSCB entries and tree pages. The value of the CLOSOP parameter determines whether one or all data sets are to be closed, and whether to free the in-core space after the disk data sets have been updated.

The record number of the disk JDSCB entry is obtained from the CR parameter of the in-core entry. The record is then written to disk and the MF parameter is set to zero. The catalog position is marked FREE for CLOSOP equal to 2 or 4. The in-core tree pages are then scanned to determine if any pages corresponding to the updated JDSCB have been modified. If so, they are written to disk and the page modification flag in PGMAN2 is set to zero. The data set reference number and the record number for the tree page are obtained from the PGMAN2 array and the first cell of the page. If CLOSOP is equal to 2 or 4, these blocks are marked as free in the PGMAN2 array. When all of the JDSCB entries have been scanned, the subroutine returns if a JCLOSE entry or branches back to the JOPEN section if control was received from there.

Remarks:

1. If JCLOSE is called by the GBLOK routine during a BNAM write operation, tree pages may be marked as modified before the CLOG array has been updated. This will cause the tree pages on disk to be inconsistent with the disk JDSCB until another CLOSE is issued.
2. JDSCB entries are read from disk even if an in-core entry exists under the following conditions.
 - a) The request is for the "Write" data set and the request is the first request received from a given terminal after a user has "signed-on".
 - b) The request is generated from BNAM functional Option 7.

In each of the above cases, any other blocks of in-core storage that have the same data set name are marked as FREE. These operations are performed since a job in the batch region may have modified the JDSCB entry and the data set tree catalog.

JOSHUA DPSTM-500	Title Subroutine: JOPEN	Vol. IX	Page B.4.62
		Rev. 1	Date 6/73

3. When more than one block of in-core space is obtained for JDSCB storage, the blocks are chained together using the Space Management Control Block.
4. If inconsistent data is detected in the CLOSE section, execution of the entire BNAM program is terminated with a call to the JTSEN subroutine.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language:

Fortran IV

Author:

F. D. Knight, J. C. Jensen

JOSHUA DPSTM-500	Title	Subroutine: JTSEN	Vol. IX	Page B.4.66
			Rev. 1	Date 6/73

Purpose: To provide abnormal ending upon detection of an inconsistency in BNAM.

Usage: Call JTSEN(CCB), where
CCB $\equiv$ System Communication Control Block

External References: LINK, ABEND, DUMP61

Program Logic: The address of the CCB is received via the argument list. The monitor dump request flag is set and a LINK is issued to DUMP61. On return from DUMP61 an ABEND is issued.

Referenced by: BNAM

Language: Assembler

Author: W. H. Brotherton, Jr.

JOSHUA DPSTM-500	Title Subroutine: MATNAM	Vol. IX	Page B.4.68
		Rev. 1	Date 6/73

Purpose: To match a name string by searching a path and/or tree pages.

Usage: Call MATNAM(CCB,BROTHR,ARRAY,PGMAN8,PGMAN2)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block.
BROTHR(1)	Real*8	Array for returning requested brothers.
ARRAY(98,1)	Real*8	In-core path-page blocks.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.

Common Area: BNAMCM - Section B.3.3.2

Subprograms Used: GPATH

PAGE

PTRACE

Program Logic: This routine calls the GPATH routine to locate an existing path. If none exists, a call is given to the PAGE routine to search the tree pages. If a path exists, a call is given to the PTRACE routine to search one path block. If the name string is matched in PTRACE or determined to be invalid, a return is issued. If the end of the path is reached and the name string has not been completely matched, but it cannot be determined as invalid, a call is given to PAGE to search the tree pages. If there exist other path blocks to the path and no match is found in the current path block, a new call is given to GPATH and the process repeated.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language: Fortran IV

Author: F. D. Knight

Purpose: To provide BNAM with the ability to search tree pages to match a name string, to traverse the brothers subordinate to a name string, and to preorder traverse a given subtree.

Usage: Call PAGE(CCB,BROTHR,PGMAN8,PGMAN2,NAMES,PGPTS,TRACE,PTHPTS,TAG,*)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block.
BROTHR(1)	Real*8	Array for returning requested brothers.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.
NAMES(1)	Real*8	Names portion of a page block. Address of NAMES in first ARRAY block is passed. BNAM indexes for its own use.
PGPTS(4,1)	Integer*2	Pointer portion of a page block. Address of PGPTS in first ARRAY block is passed. BNAM indexes for its own use.
TRACE(1)	Real*8	Names portion of a path block. Address of TRACE in first ARRAY block is passed. BNAM indexes for its own use.
PTHPTS(4,1)	Integer*2	Pointer portion of a path block. Address of PTHPTS in first ARRAY block is passed. BNAM indexes for its own use.
TAG(1)	Logical*1	Tag portion of a path block. Address of TAG in first ARRAY block is passed. BNAM indexes for its own use.
*		Return 1 - For a Job data set, a catalog inconsistency was detected.

Subprograms Used:

- GBLOK
- GPAGE
- PKPTR(UPKPTR)
- PREAD

Common Area:

BNAMCM - Section B.3.3.2

RNAME - Section B.3.3.4

Program Logic:

This routine can be called with one of three options. It can search tree pages (1) to locate a name string, (2) to traverse the brothers subordinate to a name string, and (3) to preorder traverse a subtree. In carrying out the requested option, the routine makes use a combination of the following five sections of code: (1) an initialization and startup section, (2) a match section, (3) a get brother section, (4) a traverse subtree section, and (5) a page retrieval section.

In the initialization and startup section for the match option, a path block is obtained if not already present; the startup page is made available; and the path is initialized as required. For the other two options, the tree page and cell of the starting qualifier are obtained from STORE.

In the match option (NOWOPT=1), a new path block is obtained as required; the path is loaded with the current tree cell; and a comparison is made between the cell name and the current qualifier from the requested name string. If there is an equal comparison; or if the current qualifier is DELETED and the cell name is HEX5S; or if the current qualifier is \$?\$, then the qualifier is considered matched. If no matches are made, and there exist additional brothers in the brother chain, a branch is made to the page retrieval section, where a call is given to GPAGE if another page is required. This sequence continues until either a match is made or the last brother is reached. If the last brother is a ? and the functional option is READ, this is considered a variable qualifier match. If a match is not made on a qualifier and the end of brothers is reached, a return is issued. If a match is made on a given qualifier, the tag in the path is set to one. If the cell points to data, a branch is made to PREAD to verify authorization. If the requestor is authorized, the record pointers are returned. If the requestor is not authorized, the name string is marked as no match and a return is issued. If the cell does not point to data and the name string has remaining qualifiers to be matched, the previous process is repeated for the next qualifier. If the cell does not point to data and the name string has no remaining qualifiers to be matched, a return is issued for error setting by RDPOIN.

The get brother section has two options, (1) retrieve the first brother or (2) retrieve the next brother. If no next brother exists at the current qualifier level, a brother is looked for at the next higher level. This process is repeated until a brother is found or the tree level at first entry is reached. If the functional option is for a list of brothers (NOWOPT=2), the BROTHR array is filled up to its maximum size.

JOSHUA DPSTM-500	Title Subroutine: PAGE	Vol.	Page
		IX	B.4.72
		Rev. 1	Date 6/73

If this section is used as part of the traverse subtree section, one brother is retrieved and control is returned back.

In the traverse subtree option, (NOWOPT)=3), a branch is made to the get brother section where the next brother is retrieved. When control is regained, a test is made against the level of the brother and the subtree option selected. This option states that the level of the qualifier is one of the following relative to a length parameter set by the user: =, <, ≤, >, ≥, ≠. Then a son-to-son trace is made, with a test of the subtree option at each level. This trace terminates either by violating the option or reaching a cell which points to data. If the option is violated, a branch is made to the get brother section and processing continues. If the cell points to data, PREAD is called to verify authorization. If the requestor is authorized, the record pointers are returned. If not authorized, the get brother section is called and processing continues.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language:

Fortran IV

Author:

F. D. Knight

JOSHUA

DPSTM-500

Title

Subroutine: PKPTR

Vol.

IX

Page

B.4.80

Rev.

1

Date

6/73

Purpose: To realign pointers from the first half word of a tree cell.

Usage: Call UPKPTR(PKFLD,UPKFLD)
or
Call PKPTR(PKFLD,UPKFLD)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
PKFLD	Half byte & byte	Binary pointers
UPKFLD	Half words	Binary pointers

Program Logic: For UPKPTR, save applicable registers. Transfer binary values from half byte and byte increments to half word increments. Restore applicable registers and return.

For PKPTR, save applicable registers. Transfer binary values from half word increments to half byte and byte increments.

Referenced by: BNAM

Language: Assembler

Author: R. L. Boyce, Jr.

JOSHUA

DPSTM-500

Title Subroutine: PREAD

Vol. IX	Page B.4.82
Rev. 1	Date 6/73

Purpose: To provide for record level protection under the JOSHUA System.

Usage: Call PREAD(CCB,ARRAY,PGMAN8,PGMAN2)
or
Call PWRITE(CCB,ARRAY,PGMAN8,PGMAN2)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block.
ARRAY(98,1)	Real*8	In-core path-page blocks.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.

Common Areas: BNAMCM - Section B.3.3.2
PROTEC - Section B.3.3.5

Subprograms Used: AUCHEK
JOPEN

Program Logic: There are two entry points to this routine, PREAD for authorization checking on all retrievals and PWRITE for data set specification and authorization checking for all creations and updates. The routine makes use of the information stored in CCB(8) of the System Communication Control Block and the array ALLCAT to carry out its function. For a retrieval, ALLCAT is searched to associate the data set reference number given by matching the name string, with the category of the data stored there. The routine AUCHEK then checks to determine if the requestor is authorized in which case the routine returns. If not authorized, an error return is given. If the data set reference number is not found in ALLCAT, the data set is unprotected and the routine returns. On entry to PWRITE, the category of the data set to which the update is to be made is determined from the write flag, category flag, or protection flag, the selection being the first one that is non-blank when inspected in the above order. For PWRITE, ALLCAT is searched to find the data set reference number that matches the specified data category and for which the requestor is authorized to update.

Terminating Conditions:

BNAM Errors - Section 3.5

Language: Fortran IV

Author: F. D. Knight

JOSHUA DPSTM-500	Title Subroutine: PTRACE	Vol. IX	Page B.4.84
		Rev. 1	Date 6/73

Purpose: To search a path block to match a name string.

Usage: Call PTRACE(CCB,TRACE,PTHPTS,TAG,PGMAN8,PGMAN2,*,*)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block.
TRACE(1)	Real*8	Names portion of a path block. Address of TRACE in selected ARRAY block is passed.
PTHPTS(4,1)	Integer*2	Pointer portion of a path block. Address of PTHPTS in selected ARRAY block is passed.
TAG(1)	Logical*1	Tag portion of a path block. Address of TAG in selected ARRAY block is passed.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.
*		Return 1 - Search of tree pages is not necessary: complete match or error.
*		Return 2 - Search of tree pages is necessary to complete match.

Common Area:

BNAMCM - Section B.3.3.2

RNAME - Section B.3.3.4

STATIS - Section B.3.3.6

Subprograms Used:

PREAD

UPKPTR

Program Logic:

PTRACE is used to search a path block to match a name string and to take the appropriate return branch whenever the name string is matched, found invalid, requires another path block, or requires a search of the tree pages. Starting with the first qualifier in the name string, each qualifier is compared to successive names in the path block until a qualifier match is determined or until it is decided that no match can be made. As each path name is examined, the STORE array is updated so that the son pointers of the STORE array always point to the tree location of the next path name to be examined. When a match is made, the STORE pointers for all previous qualifiers give a name string trace to that point in the string. If a TAG of 3 is found, it indicates that the previous path is terminated and a paging return is required unless the name string is

completely matched. If a TAG of 4 is found, it means that this path cannot be core contained and paging will be required without forming a path. If a TAG of 1 is found with no match, it indicates that the old and new path digress and paging is required. Only if a TAG is 0 and there is no match can searching continue in the path. A name string qualifier of 'DELETED' and a path name of HEX5 is considered a match and, for a READ request, any name string qualifier is considered to match `?XXXXXXX` where X is any character. Also the presence of '\$?\$\$' in the name string will give a match. In returning from PTRACE, there are several possibilities that exist: (1) the last brother has been compared with no match; PTRACE sets the tree page and cell pointers in COMNEW to the tree location of this last brother and issues a RETURN 1; (2) the last qualifier matched points to data; the data set reference number and record number are set; the tree page and cell pointers in COMNEW are set to the tree location of the last matched qualifier, and a RETURN 1 is issued; (3) the name string is matched and does not point to data; the tree page and cell pointers in COMNEW are set to the tree location of the last path name used and a RETURN 1 is issued; (4) it cannot be determined whether a match exists; pointers are set and a RETURN 2 is issued; (5) if a new path block is needed, a normal RETURN is issued.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language: Fortran IV

Author: F. D. Knight

JOSHUA

DPSTM-500

Title

Subroutine: RDPOIN

Vol.

IX

Page

B.4.89

Rev.

1

Date

6/73

Purpose:

To set error flags on a read, list, list brother, or rename request as applicable.

Usage:

Call RDPOIN

Common Area:

BNAMCM - Section B.3.3.2

Program Logic:

The number of names in the name string is compared with the number of names matched. If the two are equal and the last qualifier points to data, ERROR is set to zero. In all other cases, an appropriate ERROR is set and a return issued.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language:

Fortran IV

Author:

F. D. Knight

Purpose: To modify the tree pages as required to create or update a named data record.

Usage: Call WRPOIN(CCB,PGMAN8,PGMAN2,PGPTS,TAG,PTHPTS,TRACE,NAMES,ARRAY)

Arguments:

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
CCB(1)	Real*8	System Communication Control Block.
PGMAN8(2,1)	Real*8	DSNAME Control Block.
PGMAN2(8,1)	Integer*2	Space Management Control Block.
PGPTS(4,1)	Integer*2	Pointer portion of a page block. Address of PGPTS in first ARRAY block is passed. BNAM indexes for its own use.
TAG(1)	Logical*1	Tag portion of a path block. Address of TAG in first ARRAY block is passed. BNAM indexes for its own use.
PTHPTS(4,1)	Integer*2	Pointer portion of a path block. Address of PTHPTS in first ARRAY block is passed. BNAM indexes for its own use.
TRACE(1)	Real*8	Names portion of a path block. Address of TRACE in first ARRAY block is passed. BNAM indexes for its own use.
NAMES(1)	Real*8	Names portion of a page block. Address of NAMES in first ARRAY block is passed. BNAM indexes for its own use.
ARRAY(98)	Real*8	In-core path-page blocks.

Common Area: BNAMCM - Section B.3.3.2

RNAME - Section B.3.3.4

Subprograms Used:

GBLOK

GPAGE

JOPEN

PKPTR(UPKPTR)

PREAD(PWRITE)

Program Logic:

WRPOIN is the routine which modifies the tree pages as required to write a named data record. The requested number of bytes is converted to a request for a number of fixed size blocks. If the named record previously existed and the number of blocks requested is less than or equal the number previously used, the record length is updated, other paths of the data set are modified, and a return is issued.

If the named data record previously existed, and the number of blocks must be increased, a check is made for available data space. If the available space exists, the data pointers are adjusted to point to the new area. Then the record length is updated, other paths of the data set are modified, and a return is issued.

If the existing named data record is the last record in the data set, this space is recovered and used for the larger record.

If the named data record does not exist, then qualifiers must be added to the tree. If enough tree space or data space is not available, an error number is set. When a qualifier is added to a tree, the first new qualifier is added to an existing brother chain while the remaining qualifiers are added as sons to that brother. If, when adding a qualifier to an existing brother chain, the last existing brother is a variable qualifier, the variable qualifier is placed at the end of the brother chain and the new qualifier inserted before it. If the last brother is not a variable qualifier, the new qualifier is simply added at the end of the chain. If this qualifier is the terminal qualifier for the named data record, the record length is stored, the relative record number is stored, other paths of the data set are modified, and a return is issued. If there are sons to this added brother qualifier, they are chained to the brother in a son relationship. The chaining is done in the following manner: the son pointers for the brother node are set to point to the new son node, and the brother pointers of the son are set to point to itself. This process is repeated until the name string is complete. Then the record length and relative record number are stored in the tree page; the other paths of this data set are modified; and a return is issued.

Terminating Conditions:

BNAM Errors - Section B.3.5

Language:

Fortran IV

Author:

F. D. Knight

JOSHUA

DPSTM-500

Title Part C - JOSHUA Monitor

Vol. IX	Page C.O. 1
Rev. 1	Date 6/73

Contents

- C.1 General Overview of the JOSHUA Monitor
- C.2 Linkage
- C.3 Program Segments
- C.4 Edit Facilities
- C.5 Data Protection Facilities

JOSHUA

DPSTM-500

Title

Part C - Joshua Monitor

Vol.
IX

Page
C.O.2

Rev.
1

Date
6/73

(Blank)

JOSHUA

DPSTM-500

Title

General Overview of the JOSHUA Monitor

Vol.	IX	Page	C.1.1
Rev.	1	Date	6/73

Work Done By

R. L. Boyce, Jr.
W. H. Brotherton, Jr.

Report Written By

R. L. Boyce, Jr.
W. H. Brotherton, Jr.

Contents

	<u>Page</u>
1. Introduction	C.1.3
2. JOSHUA Operating System	C.1.5
2.1 Processing	C.1.5
2.1.1 Controlling Task Creation (JTMON)	C.1.5
2.1.1.1 Region Organization and Initialization	C.1.7
2.1.1.1.1 Resident Load Modules	C.1.10
2.1.1.1.2 Control Blocks	C.1.10
2.1.1.1.3 JOSHUA Control Data Set	C.1.12
2.1.1.1.4 Terminal Input/Output	C.1.13
2.1.1.2 Processing Logic	C.1.14
2.2 System Functions	C.1.17

JOSHUA
DPSTM-500

Title

General Overview of the JOSHUA Monitor

Vol.

IX

Page

C-1.2

Rev.

1

Date

6/73

(Blank)

1. Introduction

The JOSHUA Operating System (JOS), is an interfacing system between the user and the Operating System/360 (OS/360). The principal function of the JOSHUA System is to facilitate execution of problems by preservation of conveniently reusable data. The basic environment of the JOS, as shown in Figure 1, is that of user-oriented I/O devices and large volume direct access storage interconnected by modules designed to give program control capability.

The direct access devices are used to store large amounts of data and programs "on-line". The data is stored in direct organization data sets permitting storage and retrieval by relative record addressing while the programs are stored in a partitioned organized data set.

Through the program control capability of the JOSHUA Operating System the appropriate data and programs can be obtained for computational usage. The input and output developed for or by the computational usage of the system makes use of various devices depending on whether the user is communicating through the use of the on-line communication system or the batch mode of operation.

In the online system alphanumeric display terminals with cathode ray tubes for graphic capability are available for display and update of data; whereas, in the Batch Mode, the printer and reader devices are used.

The following subsections are designed to break the JOSHUA Operating System down into its components and to give the facilities, concepts, logic, and terminology related to each.

FIGURE 1. Basic Environment of JOS.

2. JOSHUA Operating System

2.1 Processing

2.1.1 Controlling Task Creation (JTMON)

When in execution, the JOSHUA Operating System is under control of the Operating System/360 Multiprogramming with a Variable Number of Tasks (MVT). It is initiated as a job step and is assigned a region of main core memory for its exclusive use. Figure 2 outlines the step-logic of the JOSHUA Operating System.

As shown in Figure 2, the JOSHUA Operating System is put into execution by OS/360; that is, OS/360 initiates a job step for the creation of the executing task (a priority-controlled program capable of independent execution).

The JOSHUA monitor task, JTMON, begins execution by providing the standard OS/360 linkage conventions and invoking the region organization and initialization module (JTINIT).

FIGURE 2 JOB STEP LOGIC

2.1.1.1 - Region Organization and Initialization

During the JOS region initialization, the basic systems control areas are initialized, all resident modules are positioned in the job step region, and the major group of basic control areas for each terminal is initialized. (See Figure 3 for organization of core storage). The FORTRAN input/output interface module (JIBCOM) and the JOS Data Manager module (CRAM) receives control during this time for interface and additional initialization. Upon receiving control, JIBCOM issues an OS LOAD to bring in JIBCOM1 (a load module which contains the FORTRAN input/output and other service routines). JIBCOM then passes control to the main subprogram of JIBCOM1. This subprogram contains an address list of all the entry points in the JIBCOM1 module and passes the pointer to this list back to JIBCOM. JIBCOM then stores these addresses locally for subsequent use. Control is also passed to DEFINE, which performs the initialization of all FORTRAN direct access data sets. The region initialization module gets the specified buffer space and passes the address and amount to the initialization section (BNAMIN) of CRAM. (See Figure 4 for a chart description of the initialization linkage).

FIGURE 3 Core Storage After Region Organization

FIGURE 4. Basic Linkage

JOSHUA DPSTM-500	Title General Overview of the JOSHUA Monitor. JOSHUA Operating System	Vol. IX	Page C.1.10
		Rev. 1	Date 6/

2.1.1.1.1 - Resident Load Modules

The Resident Load Modules are programs loaded into the terminal region at initialization time and remain there for the duration of terminal execution. They consist of Fortran I/O interface modules, the JOSHUA Data Manager, and other modules such as Display Control and the Paging Manager. These particular load modules are of two types. The Display Control and the Paging Manager are reenterable. A reenterable module can be executed by more than one task at a time and cannot be modified by itself or by any other module during execution; that is, a task may begin executing a reenterable module before a previous task has finished executing it. THE JOSHUA Data Manager and the Fortran I/O interface modules are controlled as serially reusable modules. A serially reusable module will be executed by only one task at a time.

2.1.1.1.2. - Control Blocks

The JOSHUA control consists of two logical groups:

- (a) JOSHUA region control group.
- (b) JOSHUA terminal control group.

The primary block within the region control group is the Region Control Block (RCB), which contains the addresses of all other major blocks within the group. One of these major blocks is the Region Control List (RCL), which contains the primary address to each of the terminal control groups. Another major block is the resident module list (RES), the addresses to the primary resident modules. The primary block within the terminal group is the Monitor Control Block (MCB), which contains the addresses of all other major blocks within the group, and the address of the Region Control Block (RCB). One of the major blocks in the terminal group is the Communication Control Block (CCB), which contains data and flags associated with the terminal. The CCB also contains the address of the MCB. The CCB is passed to all modules and system routines associated with a terminal, giving each the capability to locate any other control area within the region. (See Figure 5 for example of control block linkage).

FIGURE 5. Control Blocks Linkage

JOSHUA DPSTM-500	Title General Overview of the JOSHUA Monitor JOSHUA Operating System	Vol. IX	Page C.1.12
		Rev. 1	Date 6/73

2.1.1.1.3 - JOSHUA Control Data Sets

There are two types of JOSHUA control data sheets, the communication data set and the queue data set. The communication data set is divided into blocks, with a block being assigned for the exclusive use of each terminal control group. It is used for intermediate storage of small logical segments of data formatted for display at the associated terminal. The queue data set is used for intermediate storage of accounting data and serves as a queue for jobs in a hold status. These data sets are opened at initialization time and are available for use by all sub-tasks.

2.1.1.1.4 - Terminal Input/Output

Each terminal control group input/output section is organized according to the terminal type. The terminal types consist of the following:

- (a) OS/2260 local.
- (b) OS/2260 remote.
- (c) Terminal simulator (Card Reader and Printer).
- (d) Batch (Card Reader and Printer).

The system is designed to control any combination of local, remote or simulator terminals within the same job step. For the batch mode of operation, only one terminal control is allowed. All computational modules and many systems functions are executed in the batch mode. These input/output managers, if needed, are set up as resident modules and remain available for use for the duration of the job step.

JOSHUA DPSTM-500	Title General Overview of the JOSHUA Monitor. JOSHUA Operating System	Vol. IX	Page C.I.14
		Rev. 1	Date 6/73

2.1.1.2 - Processing Logic

The monitor maintains control of all processing in the JOSHUA region by creating and deleting subtasks for each terminal allocated to the job step. An analyses is made of each terminal control block group for its status and triggers execution of the next module in line. A monitor control block address list located in the Region Control List (RCL) area is scanned by the monitor in order to transfer its control through all the terminal control block groups. The sequence of module execution is determined by the function list (a logical list of subfunctions) associated with the terminal.

A function list is placed in the Function Control Table (FCT) each time a user makes a request from the terminal. A subfunction consists of a module name, system management flags, system resource flags and argument codes. The design of the function list is shown in Figure 6. Although the monitor normally controls the execution of these modules sequentially, a subfunction is capable of altering the sequence or substituting subfunction or switching to an entirely different function list.

If any subfunction completes abnormally, the function is terminated and the ABend function is placed in the FCT. The purpose of the ABend function is to perform cleanup as applicable and to inform the user of the problem. Prior to creating a subtask the monitor assigns any reusable resources requested via the system resource flags. If any of the requested resources is assigned to another subtask the subfunction execution is delayed. After completing the creation of all subtasks that are ready, the monitor waits for completion of one or more subtasks. Upon regaining control, all completed subtasks are deleted and serially reusable resources are released. The terminal control block groups are again scanned and flags set in preparation to the execution of the next subfunction. This process is repeated until the job termination flag is set by the End Job function. See Figure 7 for the basic logic of JTMON with respect to a two-terminal environment.

FUNCTION CONTROL TABLE (FCT)			
HEX	DEC	SUB NAME	FLAGS AND CODES (HEX)
0	0	MODULE 1	MMMM RRRR N CCCCCC
10	16		
.	.		
F0	240		

SYSTEM MANAGEMENT FLAGS

SYSTEM RESOURCE FLAGS

NUMBER OF ARGUMENTS

ARGUMENT CODES

FIGURE 6. Design of Function Control Table

MCB LIST	
0	A (MCB1)
4	A (MCB2)

MCB1 FCT		
0	NAME 1	FLAGS
16	NAME 2	FLAGS

MCB2 FCT		
0	NAME A	FLAGS
16	NAME B	FLAGS

FIGURE 7. Basic Logic of JTMON with Respect to a Two-Terminal Environment.

2.2 - System Functions

The JOSHUA Operating System (JOS) provides the necessary facilities to manipulate data, to initiate job execution (terminal only), and to do other tasks via user function requests. These "function requests" are entered on the "select frame" of the appropriate input device. A "function request" invokes a particular JOS module that will accomplish a specific task such as displaying the data in the JOSHUA Named Data Record, logging a sign-off from terminal, etc. Function requests may require "arguments" such as the name of a record to be displayed or the name of a job to be executed. The following is a list of the functions (with description) used in the JOSHUA Operating System.

- *AB - Set up abnormal ending.
- CO - Copy an existing record to another record.
- CD - Create or modify a record.
- CT - Create a template (list descriptor and format) for a given record type.
- DB - Delete a specified set of records or subtrees from the tree catalog.
- DC - Display the data set name catalog.
- DD - Display a record or group of records.
- DE - Delete a record or subtree from the tree catalog.
- DT - Display a template.
- DU - Unload a JOSHUA Data set.
- ED - Initiate text editor.
- *EJ - Terminate a batch "session" (run).
- EM - Execute a module.
- EN - Terminate a terminal "session".
- EP - Execute a procedure.
- EX - Execute a job.
- FM - Specify an SRP reactor facemap.
- IP - Interrupt a procedure.
- LB - Display the modes at a specified level of a data set tree catalog.
- LI - Display the names of the records in a data set or a subtree of the data set.
- LO - Load a JOSHUA Data set.
- LS - Perform least square calculation
- MA - Maintain a JOSHUA data set.
- MD - Modify a record.
- ME - Merge a JOSHUA data set.
- MT - Modify a template.
- PD - Punch a JOSHUA record.
- PR - Print specified display frames.
- *RA - Request abnormal ending.
- *RD - Request monitor dump.
- RE - Rename a node in data set tree catalog.
- RM - Specify an SRP reactor facemap.
- RP - Restart a procedure.
- SC - Release all or portion of space allocated to a data set.
- *SE - Select the requested function.
- SI - Set up terminal sign-on.
- SJ - Establish a jobname for execution of a module or locate an existing jobname and place it in the system directory.
- SK - Skip a procedure step.
- *SN - Set up batch sign-on.
- SP - Stop a procedure.
- ST - Display the status of a job.
- SU - Set user data set for read hierarchy.

JOSHUA DPSTM-500	Title General Overview of the JOSHUA Monitor JOSHUA Operating System	Vol. IX	Page C.1.18
		Rev. 1	Date 6/73

SW - Set up update data set.
*TE - Test module or function.
*TS - Test module or function.
VA - Perform requested vary operation.
VQ - Display the JCL for all jobs in the holdqueue.
*VV - View protection violations.
*XB - Release holdqueue JCL to HASP.
*XR - Execute within JOSHUA Terminal region.

* For system use only.

JOSHUA

DPSTM-500

Title Linkage

Vol. Page
IX C.2.1

Rev. Date
1 6/73

Work Done By

R. L. Boyce, Jr.
W. H. Brotherton, Jr.

Report Written By

R. L. Boyce, Jr.
W. H. Brotherton, Jr.

Contents

1. Linkage

Page

C.2.3

JOSHUA DPSTM-500	Title Linkage	Vol. IX	Page C.2.2
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

Linkage

Vol.

IX

Page

C.2.4

Rev.

1

Date

6/73

(Blank)

1. Linkage

\$LINK is an interface routine which must be included in all JOS modules which reference the resident modules JIBCOM, JIBCOM1, and CRAM (see Processing Section). \$LINK contains entry point names identical to the entries in JIBCOM1 and CRAM. When a JOS module passes control to one of these entry points, \$LINK resets the entry address to the corresponding entry point within JIBCOM1 or CRAM. The first reference to one of the entry points in \$LINK corresponding to an entry point in JIBCOM1 or CRAM results in control being passed to JIBCOM or CRAM which returns the pointer to a list of actual addresses in JIBCOM1 or CRAM.

A JOSHUA precompiled program references a common area named \$JCOMA which contains three arrays: \$JCB, \$NAMES, and \$BNAMI. Since certain initializations are required in the \$BNAMI array, \$JCOMA has been established as a block data routine in the JOSHUA subroutine library and is included in the load module during linkage editing.

Each JOSHUA precompiled main program executes the module JIBINIT to initialize JIBCOM1 and to transfer the CCB which it receives as an argument to the array \$JCB.

See Figure 1 for Linkage description.

- * - Transfer to appropriate entry within link.
- ** - Pointer reset from corresponding address in address list. If first time to \$LINK the address list is obtained from JIBCOM.
- *** - Transfer to actual I/O routine within JIBCOM1.

Figure 1. Basic Linkage

JOSHUA DPSTM-500	Title Linkage	Vol. IX	Page C.2.6
		Rev. 1	Date 6/73

(Blank)

Work Done By

R. L. Boyce, Jr.
W. H. Brotherton, Jr.

Report Written By

R. L. Boyce, Jr.
W. H. Brotherton, Jr.

Contents

	<u>Page</u>
1. Introduction	C.3.3
2. Control Blocks	C.3.5
2.1 Accounting Control Block	C.3.5
2.2 Accounting Control Table	C.3.7
2.3 Argument List	C.3.11
2.4 Communication Control Block	C.3.13
2.5 Communication Control List	C.3.17
2.6 Communication Control Table	C.3.19
2.7 Communication Line Group Address List	C.3.21
2.8 Data Definition Table	C.3.23
2.9 Data Manager (BNAM) Statistics Table	C.3.25
2.10 Dynamic Allocation Table	C.3.27
2.11 Function Control List	C.3.29
2.12 Function Control Table	C.3.31
2.13 Function Timing Work Area	C.3.35
2.14 General Card Control	C.3.37
2.15 Graphic Parameters Address List	C.3.39
2.16 Monitor Control Block	C.3.43
2.17 Monitor Control Flags	C.3.47
2.18 O/S 360 Parameters Block	C.3.49
2.19 O/S 360 Parameters List	C.3.51
2.20 Parameter Control Block	C.3.53
2.21 Queue Control Block	C.3.57
2.22 Reader Control Table	C.3.59
2.23 Region Control Block	C.3.61
2.24 Region Control Flags	C.3.65
2.25 Region Control List	C.3.67
2.26 Resident Module List	C.3.69
2.27 Text Editor Control Block	C.3.71
2.27.1 Text Editor Control Block (Editor)	C.3.71
2.27.2 Text Editor Control Block (Monitor)	C.3.73
2.28 User Identification Block	C.3.75
2.29 Work Control Block	C.3.77
2.30 Work Extension Table	C.3.81
3. Source Code Blocks	C.3.85
3.1 FREEBUG	C.3.85
3.2 GETBUG	C.3.86
3.3 NORMEXIT	C.3.87
3.4 REGUSAGE	C.3.88
3.5 SAVER	C.3.89
3.6 SAVES	C.3.90

JOSHUA DPSTM-500	Title Program Segments	Vol. IX	Page C.3.2
		Rev. 1	Date 6/73

(Blank)

1. Introduction

This section contains reference information about the contents and format of the major program segments used by more than one component of the JOSHUA Operating System (JOS). A diagram of each segment is followed by description of its fields.

The system control blocks are the primary means for communicating information among the major parts of JOS. The information is stored in the control blocks in a compact, readily accessible form. The addresses maintained in the control blocks permit any component of JOS to locate other control blocks.

The source code blocks are segments of code applicable to many programs and are retrieved from a library at assembly time.

Usage of the program segments is discussed as applicable within the description of each JOS module and subroutine. See Part E for descriptions of modules and subroutines.

JOSHUA DPSTM-500	Title Program Segments	Vol. IX	Page C.3.4
		Rev. 1	Date 6/73

(Blank)

JOSHUA
PSTM-500

Title CONTROL BLOCK:
 ACCOUNTING CONTROL BLOCK

Vol. IX	Page C.3.5
Rev. 1	Date 6/73

ACCOUNTING CONTROL BLOCK (ACB)

```
*****
*0(0)                                *2(2)                                *
*  ACBSTART                          *  ACBEND                          *
*   FIRST RECORD NUMBER              *   LAST RECORD ALLOCATED          *
*****
*4(4)                                *
*  ACBREC                            *
*                                CURRENT RECORD NUMBER                    *
*****
*8(8)                                *10(A)                                * 11(B)                                *
*  ACBLAST                          *  ACBFLAGS                          *
*   LAST ACTIVE RECORD              *   FLAGS                          *   RESERVED                      *
*****
*12(C)                                *
*
*                                RESERVED                                *
*****
```

ACCOUNTING CONTROL BLOCK (ACB)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	ACBSTART	FIRST RECORD NUMBER OF THE QUEUE DATA SET ALLOCATED TO THE ACCOUNTING DATA SET
2(2)	2	ACBEND	LAST RECORD NUMBER OF THE QUEUE DATA SET ALLOCATED TO THE ACCOUNTING DATA SET
4(4)	4	ACBREC	CURRENT RECORD NUMBER DURING INPUT/OUTPUT PROCESSING
8(8)	2	ACBLAST	LAST ACTIVE RECORD NUMBER OF THE ACCOUNTING DATA SET
10(A)	1	ACBFLAGS	ACCOUNTING DATA SET FLAGS
		BIT NAME	
		ACBTREQ	1... .. TRANSFER OF DATA REQUESTED
			.1... .. RESERVED
			..1. RESERVED
			...1 RESERVED
			 1... RESERVED
			1.. RESERVED
			1. RESERVED
			1 TEST SYSTEM ACCOUNTING QUEUE
11(B)	5		RESERVED

JOSHUA
DPSTM-500

Title CONTROL BLOCK:
ACCOUNTING CONTROL TABLE

Vol.	Page
IX	C.3.7
Rev.	Date
1	6/73

ACCOUNTING CONTROL TABLE (ACT)

```
*****
*010)                                *2(2)
*  ACTEAGE                          *  ACTUCS
*  USER BADGE NUMBER (BINARY)      *  USER DATA SET (BINARY)
*****
*4(4)
*  ACTCC                             USER COST CODE *CCCCSS* (COST CODE & SUB CODE)
*****
*  *10(A)
*  *  ACTCATCN
*  *    DATE CN
*****
*12(C)
*  ACTCATCF
*  CATE OFF
*****
*  *14(E)
*  *  ACTIMCN
*  *    TIME ON
*****
*  *17(11)
*  *  ACTIMCFF
*  *    TIME CFF
*****
*20(14)
*  ACTCSPLS
*  NUMBER OF DISPLAYS
*****
*24(18)
*  ACTABTYP
*  TABLE TYPE
*****
*  *25(19)
*  *  ACTFCT
*****
*  FUNCTION REQUESTS STATISTICS
*****
*  *22(16)
*  *  ACTUNIT
*  *    TERMINAL ID * PAGES PRINTED
*****
*  *23(17)
*  *  ACTPRTPG
*****
*80(50)
*  ACTWCRK
*****
*  COUPLE WORD WORK AREA
*****
*****
```

ACCOUNTING CONTROL TABLE (ACT)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	ACTBADGE	USER BADGE NUMBER (BINARY)
2(2)	2	ACTUCS	USER DATA SET (BINARY)
4(4)	6	ACTCC	USER COST CODE OF THE FORM CCCCSS WHERE: CCCC IS THE CODE SS IS THE SUB CODE
10(A)	2	ACTDATON	DATE TERMINAL SESSION STARTED AS PACKED DECIMAL OF THE FORM YDDD WHERE: Y IS THE YEAR DDD IS THE DAY
12(C)	2	ACTDATOF	DATE TERMINAL SESSION ENDED AS PACKED DECIMAL OF THE FORM YDDD WHERE: Y IS THE YEAR DDD IS THE DAY
14(E)	3	ACTIMON	TIME TERMINAL SESSION STARTED AS PACKED DECIMAL OF THE FORM HHMMSS WHERE: HH IS THE HOUR MM IS THE MINUTE SS IS THE SECOND
17(11)	3	ACTIMOFF	TIME TERMINAL SESSION ENDED AS PACKED DECIMAL OF THE FORM HHMMSS WHERE: HH IS THE HOUR MM IS THE MINUTE SS IS THE SECOND
20(14)	2	ACTDSPLS	NUMBER OF DISPLAYS DURING THE TERMINAL SESSION
22(16)	1	ACTUNIT	TERMINAL IDENTIFICATION NUMBER (BINARY)
23(14)	1	ACTPRTPG	NUMBER OF DISPLAYS PRINTED DURING TERMINAL SESSION

ACCOUNTING CONTROL TABLE (ACT) CONT'D

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
--------	-------	------------	--------------------------------------

24(18)	1	ACTABTYP BIT NAME	ACCOUNTING CONTROL TABLE TYPE
--------	---	----------------------	-------------------------------

1...		RESERVED
.1..		RESERVED
..1.		RESERVED
...1		RESERVED
....	1...	RESERVED
....	.1..	RESERVED

ACTVLTN		.1.	SIGN ON VIOLATION
ACTPROTC		...1	DATA PROTECTION SESSION

25(19)		ACTFCT	FUNCTION REQUEST STATISTICS (CATAGORIES 0(0) THROUGH 10(A)
--------	--	--------	---

CATAGORY 0(0)

25(19)	1	ACTECTNR	NUMBER OF FUNCTION REQUESTS (BINARY)
26(1A)	2	ACTECTWT	FUNCTION WAIT TIME (BINARY IN SECCNDS)
28(1C)	2	ACTECTET	FUNCTION ELASPED TIME (BINARY IN SECCNDS)

CATAGORY 1(1) THROUGH 10(A)

30(1E)	50		MULTIPLE CATEGORYS IN 5-BYTE INCREMENTS OF THE SAME FORM AS IN CATAGORY 0(0)
--------	----	--	---

80(50)	8	ACTWORK	DOUBLE WORD WORK AREA
--------	---	---------	-----------------------

JOSHUA DPSTM-500	Title CONTROL BLOCK: ACCOUNTING CONTROL TABLE	Vol. IX	Page C.3.10
		Rev. 1	Date 6/7

(Blank)

JOSHUA DPSTM-500	Title CONTROL BLOCK: ARGUMENT LIST	Vol. IX	Page C.3.11
		Rev. 1	Date 6/73

ARGUMENT LIST (ARG)

```

*****
*C(0)
*  ARGCCB          ADDRESS CF
*                  COMMUNICATION CONTROL BLOCK
*****
*4(4)
*  ARGVARG          ADDRESSES CF
*                  VARIABLE ARGUMENTS
*****
63(3F)*

```


JOSHUA DPSTM-500	Title CONTROL BLOCK: ARGUMENT LIST	Vol. IX	Page C.3.12
		Rev. 1	Date 6/73

ARGUMENT LIST (ARG)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
-----	-----	-----	-----
0(0)	4	ARGCCB	ADDRESS OF COMMUNICATION CONTROL BLOCK
8(8)	60		ADDRESSES OF VARIABLE ARGUMENTS

COMMUNICATION CONTROL BLOCK (CCB)

```

*****
*0(0)
* CCBJCB
*
* ACTIVE JOB DATA SET NAME
*****
*8(8)
* CCBUSER
*
* ACTIVE USER DATA SET NAME
*****
*16(10)
* CCBSYS
*
* DATA SET REFERENCE NUMBER FOR 'SYSOUT'
*****
*20(14)
* CCBERR
*
* ERROR FLAG
*****
*24(18)
* CCBMODE
*
* MOVE FLAG
*****
*28(10)
* CCBERRN
*
* ERROR NUMBER
*****
*32(20)
* CCBNERR
*
* NUMBER OF ERRORS
*****
*36(24)
* CCBMONF
*
* MONITOR FLAG
*****
*40(28)
* CCBAPEND
* APPEND FLAG
*****
*42(2A)
* CCBSINCN
* SIGN ON FLAG
*****
*44(2C)
* CCBSECCB
*
* CCBVERIFY
* VERIFY FLAG
*****
*48(30)
* CCBTIMER
*
* TIMER FLAG
*****
*52(34)
* CCBCTLF $CBGTIME
*
*****

```

JOSHUA DPSTM-500	Title CONTROL BLOCK: COMMUNICATION CONTROL BLOCK	Vol. IX	Page C.3.14
		Rev. 1	Date 6/7

COMMUNICATION CONTROL BLOCK (CCB) CONT'D.

```

*****
*56(38)                                *58(3A)                                *
*  CCBSFLGS  $CBSF                      *  $CERF                                *
*                                           *                                           *
*                                           *                                           *
*  CCBWF                      *  $CBCF                                *
*                                           *                                           *
*****
*64(40)                                *
*  CCBSCTH                      *
*                ACTIVE SCRATCH DATA SET NAME                      *
*****
*72(48)                                *
*  CCBCS1                      *
*                FIRST INPUT DATA SET                              *
*****
*80(50)                                *
*  CCBCS2                      *
*                SECOND INPUT DATA SET                              *
*****
*88(58)                                *
*  CCBCS3                      *
*                THIRD INPUT DATA SET                               *
*****
*96(60)                                *
*  CCBCS4                      *
*                FOURTH INPUT DATA SET                             *
*****
*104(68)                               *
*  CCBCCB                      *
*                ADDRESS OF                                         *
*                THIS COMMUNICATION CONTROL BLOCK                  *
*****
*108(6C)                               *
*  CCSTAE                      *
*                ADDRESS OF                                         *
*                'STAE' EXIT ROUTINE                               *
*****
*112(70)                               *
*  CCBCC                      *
*                USER COST CODE                                    *
*****
*120(78)                               *
*  CCBADCE                      *
*                USER BADGE NUMBER                                 *
*****
*124(7C)                               *
*  CCBMCB                      *
*                ADDRESS OF                                         *
*                MONITOR CONTROL BLOCK                             *
*****

```

COMMUNICATION CONTROL BLOCK (CCB)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	8	CCBJOB	ACTIVE JOB DATA SET NAME.
8(8)	8	CCBUSER	ACTIVE USER DATA SET NAME.
16(10)	4	CCBSYS	DATA SET REFERENCE NUMBER FOR 'SYSOUT'.
20(14)	4	CCBERR	ERROR FLAG: 0 - NO ERROR. 1 - ERROR.
24(18)	4	CCBMODE	MODE FLAG: 0 - 2260 TERMINALS AND/OR SIMULATOR. 1 - BATCH.
28(1C)	4	CCBERRN	ERROR NUMBER
32(20)	4	CCBNERR	NUMBER OF ERRORS
36(24)	4	CCBMONF	MONITOR FLAG 0 - COMPUTATIONAL. 1 - FUNCTION.
40(28)	2	CCBAPEND	APPEND FLAG 0 - STANDARD APPEND OF DATA SET NAME. 1 - NO APPEND OF DATA SET NAME.
42(2A)	2	CCBSINCN	SIGN ON FLAG 0 - DATA MANAGER INITIALIZED. 1 - DATA MANAGER NOT INITIALIZED.
44(2C)	2	CCBSECOB	
46(2E)	2	CCBVERIFY	VERIFY FLAG 0 - DATA SET UPDATE AUTHORIZED. 1 - DATA SET UPDATE NOT AUTHORIZED.
48(30)	4	CCBTIMER	TIMER FLAG 0 - TIMING OF ROUTINES NOT REQUESTED. 1 - TIMING OF ROUTINES REQUESTED.

JOSHUA DPSTM-500	Title CONTROL BLOCK: COMMUNICATION CONTROL BLOCK	Vol. IX	Page C.3.16
		Rev. 1	Date 6/73

COMMUNICATION CONTROL BLOCK (CCB) CONT'D.

52(34)	4	CCBCTLF CCBGTIME	ZERO (2260 TERMINALS AND/OR SIMULATOR) ESTIMATED JOB CPU TIME (BATCH)
56(38)	2	CCBSF	SIGN ON FLAG
58(3A)	2	CCBRF	READ FLAG
60(3C)	2	CCBWF	WRITE FLAG
62(3E)	2	CCBCF	CATAGORY FLAG
64(40)	8	CCBSCRTH	ACTIVE SCRATCH DATA SET NAME
72(48)	8	CCBDS1	FIRST INPUT DATA SET
80(50)	8	CCBDS2	SECCND INPUT DATA SET
88(58)	8	CCBDS3	THIRD INPUT DATA SET
96(60)	8	CCBDS4	FOURTH INPUT DATA SET
104(68)	4	CCBCCB	ADDRESS OF THE SYSTEM COMMUNICATION CONTROL BLOCK.
108(6C)	4	CCBSTAE	ADDRESS OF THE 'STAE' EXIT ROUTINE
112(70)	8	CCBCC	USER COST CODE
120(78)	4	CCBBADGE	USER BADGE NUMBER
124(7C)	4	CCBMCB	ADDRESS OF MONITOR CONTROL BLOCK

JOSHUA
DPSTM-500

Title

CONTROL BLOCK:
COMMUNICATION CONTROL LIST

Vol.

IX

Page

C.3.17

Rev.

1

Date

6/73

COMMUNICATION CONTROL LIST (CCL)

```
*****
*0(0)          *          *2(2)          *
* CCLUNITB      *          * CCLSTART      *
*   TERMINAL ID * RESERVED *   STARTING RECORD NUMBER   *
*****
*4(4)          *          *
* CCLEND        *          *
*   LAST RECORD NUMBER   *          *
*****
```

JOSHUA DPSTM-500	Title CONTROL BLOCK: COMMUNICATION CONTROL LIST	Vol. IX	Page C.3.18
		Rev. 1	Date 6/7

COMMUNICATION CONTROL LIST (CCL)

<u>OFFSET</u>	<u>BYTES</u>	<u>FIELD NAME</u>	<u>FIELD DESCRIPTION, CONTENTS, MEANING</u>
0(0)	1	CCLUNITB	TERMINAL IDENTIFICATION NUMBER (NOT USED)
1(1)	1		RESERVED
2(2)	2	CCLSTART	FIRST RECORD NUMBER OF COMMUNICATION DATA SET ALLOCATED TO SPECIFIED TERMINAL
4(4)	2	CCELEND	LAST RECORD NUMBER OF COMMUNICATION DATA SET ALLOCATED TO SPECIFIED TERMINAL
6(6)			REPEAT AS ABOVE FOR ALL TERMINALS ALLOCATED TO THE JOB

COMMUNICATION CONTROL TABLE (CCT)

```

*****
*0(0)                                *2(2)                                *
* CCTSTART                          * CCTEND                                *
* FIRST RECORD NUMBER              * LAST RECORD NUMBER                *
*****
*4(4)                                *
* CCTRECRD                          *
* CURRENT RECORD NUMBER              *
*****
*8(8)                                *10(A)                                *
* CCTLAST                          * CCTPAGE                            *
* LAST ACTIVE RECORD NUMBER        * CURRENT PAGE NUMBER              *
*****
*12(C)          *13(D)          *14(E)                                *
* CCTOPTN        *              * CCTDF                                *
* I/O FLAGS      * RESERVED    * DISPLAY FLAG                        *
*****
*16(10)                                *
* RESERVED                                *
*****
*20(14)                                *
* CCTPCT          ADDRESS OF          *
* PAGING CONTROL TABLE              *
*****
*24(18)                                *
* CCTPCTN          ADDRESS OF          *
* END OF PAGING CONTROL TABLE        *
*****
*28(1C)                                *
* RESERVED                                *
*****

```


JOSHUA DPSTM-500	Title CONTROL BLOCK: COMMUNICATION CONTROL TABLE	Vol. IX	Page C.3.20
		Rev. 1	Date 6/73

COMMUNICATION CONTROL TABLE (CCT)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	CCTSTART	FIRST RECORD NUMBER
2(2)	2	CCTEND	LAST RECORD NUMBER
4(4)	4	CCTRECRD	CURRENT RECORD NUMBER
8(8)	2	CCTLAST	LAST ACTIVE RECORD NUMBER
10(A)	2	CCTPAGE	CURRENT PAGE NUMBER
12(C)	1	CCTOPTN	INPUT/OUTPUT FLAGS
		BIT NAME	
		CCTRSTRT	1... .. RESTART AT FIRST PAGE
		CCTRRDWR	.1.. ... READ OR WRITE SAME PAGE
		CCTEOD	..1. ... END OF DATA
		CCTEOD	...1 END OF FILE
		CCTINSERT	 1... INSERT AFTER REQUESTED PAGE
		CCTMODFY	1.. UPDATE PAGE
		CCTDELTE	1. DELETE REQUESTED PAGE
			1 RESERVED
13(D)	1		RESERVED
14(E)	2	CCTDF	DISPLAY FLAG
16(10)	4		RESERVED
20(14)	4	CCTPCT	ADDRESS OF PAGING CONTROL TABLE
24(18)	4	CCTPCTN	ADDRESS OF END OF PAGING CONTROL TABLE
28(1C)	4		RESERVED

COMMUNICATION LINE GROUP ADDRESS LIST (CAL)

```

*****
*0(0)          *2(2)
* CALLNGTH    * CALIDENT
*   LENGTH    *
*****
*               CONTROL BLOCK IDENTIFICATION
*
*
*****
*8(8)
* CALISTC      ADDRESS OF
*               COMMUNICATION LINE TERMINAL LIST
*****
*12(C)
*
*               RESERVED
*****
*16(10)
*
*               RESERVED
*****
*20(14)
* CALDCBC      ADDRESS OF
*               COMMUNICATION LINE GROUP DATA CONTROL BLOCK
*****
*24(18)
* CALDECBC     ADDRESS OF
*               COMMUNICATION LINE GROUP DATA EVENT CONTROL BLOCK
*****
*28(1C)
*
*               RESERVED
*****
*32(20)
*               DCB (STARTING AT BYTE 20(14)) FOR
*               COMMUNICATION LINE GROUP
*****
*64(40)
*               DECB FOR
*               COMMUNICATION LINE GROUP
*****
*104(68)
*               TERMINAL LIST FOR
*               COMMUNICATION LINE GROUP
*****

```

COMMUNICATION LINE GROUP ADDRESS LIST (CAL)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	CALLNGTH	LENGTH OF CONTROL BLOCK
2(2)	6	CALIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	4	CALISTC	ADDRESS OF TERMINAL LIST FOR COMMUNICATION LINE GROUP
12(C)	8		RESERVED
20(14)	4	CALDCBC	ADDRESS OF DATA CONTROL BLOCK FOR COMMUNICATION LINE GROUP
24(18)	4	CALDECBC	ADDRESS OF DATA EVENT CONTROL BLOCK FOR COMMUNICATION LINE GROUP
28(1C)	4		RESERVED
32(20)	32		DATA CONTROL BLOCK (STARTING AT BYTE 20(14)) FOR COMMUNICATION LINE GROUP
64(40)	40		DATA EVENT CONTROL BLOCK FOR COMMUNICATION
104(68)	3		TERMINAL LIST FOR COMMUNICATION LINE GROUP

DATA DEFINITION TABLE (DDT)

```

*****
*0(0)
* DDTDDN
*
* DATA DEFINITION NAME
*
*****
*8(8) *9(9)
* DDTUNIT8 * DDTUNIT(E)
* TERMINAL ID(8) * TERMINAL IDENTIFICATION (EBCDIC)
*****
*
***
*
***

```

JOSHUA

DPSTM-500

Title CONTROL BLOCK:
DATA DEFINITION TABLE

Vol. IX	Page C.3.24
Rev. 1	Date 6/73

DATA DEFINITION TABLE (DDT)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	8	DDTDDN	DATA DEFINITION NAME
8(8)	1	DDTUNITB	TERMINAL IDENTIFICATION NUMBER (BINARY)
9(9)	3	DDTUNITE	TERMINAL IDENTIFICATION NUMBER (EBCDIC)
12(C)			REPEAT AS ABOVE FOR ALL TERMINAL TYPE DD CARDS IN THE JOB STREAM.

DATA MANAGER (BNAM) STATISTICS TABLE (BST)

```

*****
*0(0)          *2(2)
* BSTLNTH      * BSTIDENT
*   LENGTH     *
*****
*               CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
* BSTEMTOT
*               TOTAL FOR 'EM' FUNCTION
*****
*108(6C)
* BSTCTTCL
*               TOTAL FOR FUNCTIONS OTHER THAN 'EM'
*****
*208(D0)
* BSTCATNC
*               NUMBER OF FUNCTION REQUESTS
*****
*212(D4)
* BSTFCTCT
*               TOTAL FOR FUNCTION
*****

```

JOSHUA

DPSTM-500

Title

CONTROL BLOCK:
DATA MANAGER (BNAM) STATISTICS TABLE

Vol.

IX

Page

C.3.26

Rev.

1

Date

6/75

DATA MANAGER (BNAM) STATISTICS TABLE (BST)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	BSTLNTH	LENGTH OF CONTROL BLOCK
2(2)	6	BSTIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	100	BSTEMTOT	TOTALS FOR 'EM' FUNCTION
108(6C)	100	BSTOTTOL	TOTALS FOR OTHER THAN 'EM' FUNCTIONS
208(D0)	4	BSTCATNO	NUMBER OF FUNCTION REQUESTS
212(D4)	12	BSTFCTCT	TOTALS FOR FUNCTION
224(E0)	160		REPEAT 'BSTCATNO' AND 'BSTFCTCT' FOR EACH ADDITIONAL CATEGORY

```

*****
*0(0)
* DATDUNAM
*          DDNAME
*****
*8(8)
* DATSTAT
*          STATUS FLAGS
*****
*9(9)
* DATDSORG
*          DATA SET ORGANIZATION
*****
*10(A)
* DATUNITB
*          TERMINAL IDENTIFICATION (BINARY)
*****
*11(B)
*
*          RESERVED
*****
*12(C)
* DATCAL
*          ADDRESS OF
*          DYNAMIC ALLOCATION LIST
*****
*16(10)
* DATDSN
*          ADDRESS OF
*          DATA SET NAME
*****
*20(14)
* DATDCB
*          ADDRESS OF
*          DATA CONTROL BLOCK
*****
*24(18)
* DATDSNAM
*          DATA SET NAME
*****
*68(44)
*
*          RESERVED
*****
*****71(47)X
*****
*
```


OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
--------	-------	------------	--------------------------------------

DYNAMIC ALLOCATION TABLE (DAT)

0(0)	8	DATDDNAM	DATA DEFINITION NAME.
8(8)	1	DATSTAT	STATUS FLAGS.
		BIT NAME	
		DATLAST	1... LAST TABLE ENTRY
		DATINUSE	.1.. TABLE ENTRY IN USE
		DATSPECL	..1. SPECIAL HANDLING REQUIRED
			...1 RESERVED
			 1... RESERVED
			1.. RESERVED
			1. RESERVED
			1 RESERVED
9(9)	1	DATDSORG	DATA SET ORGANIZATION.
		BIT NAME	
		DATISORG	1... (IS) INDEXED SEQUENTIAL.
			.1.. (PS) PHYSICAL SEQUENTIAL.
			..1. (DA) DIRECT.
			...X XX.. RESERVED.
			1. (PO) PARTITIONED.
			1 UNMOVABLE (LOCATION DEPENDENT).
10(A)	1	DATUNITB	TERMINAL IDENTIFICATION
11(B)	1		RESERVED
12(C)	4	DATDAL	ADDRESS OF DYNAMIC ALLOCATION LIST
16(10)	4	DATDSN	ADDRESS OF DATA SET NAME
20(14)	4	DATDCB	ADDRESS OF DATA CONTROL BLOCK
24(18)	44	DATDSNAM	DATA SET NAME
68(44)	4		RESERVED

FUNCTION CONTROL LIST (FCL)

```

*****
*0(0)
* FCLSTART
* FIRST FUNCTION LIST RECORD NUMBER
*****
*2(2)
* FCLEND
* LAST FUNCTION LIST RECORD NUMBER
*****
*4(4)
* FCLFCTNM
* FUNCTION NAME CODE
*****
*6(6)
* FCLFLAG
* FUNCTION FLAGS
*****
*7(7)
* FCLRECNO
* FUNCTION LIST RECORD NUMBER
*****
7(7)*
*****

```

JOSHUA

DPSTM-500

Title CONTROL BLOCK:
FUNCTION CONTROL LIST

Vol. IX

Page
C.3.30

Rev.
1

Date
6/73

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
--------	-------	------------	--------------------------------------

FUNCTION CONTROL LIST (FCL)

0(0)	2	FCLSTART	FIRST FUNCTION LIST RECORD NUMBER (NOT USED)
2(2)	2	FCLEND	LAST FUNCTION LIST RECORD NUMBER (NOT USED)
4(4)	2	FCLFCTNM	FUNCTION NAME CODE
6(6)	1	FCLFLAG	FUNCTION FLAGS
		BIT NAME	
		XXXX	FUNCTION CATEGORY CODE
		 1...	RESERVED
		i..	TERMINAL MODE ONLY
		1.	BATCH MODE ONLY
		1	FUNCTION OFFLINE
7(7)	1	FCLRECNO	FUNCTION LIST RECORD NUMBER (NOT USED)

JOSHUA

DPSTM-500

Title

CONTROL BLOCK:
FUNCTION CONTROL TABLE

Vol.

IX

Page

C.3.31

Rev.

1

Date

6/73

FUNCTION CONTROL TABLE (FCT)

```

*****
*0(0)
* FCTMODUL
*
*          NAME OF SUB-FUNCTION
*****
*8(8)      *9(9)      *10(A)      *11(B)
* FCTMFLG1  * FCTMFLG2  * FCTRSRC1  * FCTRSRC2
*  SYSTEM MANAGEMENT FLAGS      *  SYSTEM RESOURCE FLAGS
*****
*12(C)
* FCTMCODE
*
*          ARGUMENT CODES
*****

```

FUNCTION CONTROL TABLE (FCT)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	8	FCTMODUL	NAME OF SUB-FUNCTION
8(8)	1	FCTMFLG1 BIT NAME	SYSTEM MANAGEMENT FLAG 1
		FCTLINK	1... .. EXECUTE SUB-FUNCTION UNDER CONTROL OF MONITOR TASK
			.1.. .. RESERVED
			..1. .. RESERVED
		FCTJOBER	...1 TERMINATE JOB UPON ABNORMAL COMPLETION OF SUB-FUNCTION
			 1... RESERVED
			1.. RESERVED
		FCTLOGIC	1. LOGIC FLAG 1
		FCTBLARG	1 ARGUMENTS REQUESTED
9(9)	1	FCTMFLG2 BIT NAME	SYSTEM MANAGEMENT FLAG 2
		FCTMWRPG	1... .. BYPASS OUTPUT TO COMMUNI- CATION DATA SET
		FCTMRDPG	.1.. .. BYPASS INPUT FROM COMMUNI- CATION DATA SET
		FCTMBKPG	..1. BYPASS RELEASE OF DISPLAY BUFFER
		FCTALIGN	...1 ALIGN DATA TO/FROM COMMUNI- CATION DATA SET
			 1... RESERVED
			1.. RESERVED
		FCTLOGC2	1. LOGIC FLAG 2
			1 RESERVED
10(A)	1	FCTRSRC1 BIT NAME	SYSTEM RESOURCE MANAGEMENT FLAG RESOURCE
		FCTBNAM	1... .. JOSHUA DATA MANAGER
		FCTINRDR	.1.. .. INTERNAL READER
		FCTACTLG	..1. JOSHUA ACCOUNTING LOG
			...1 RESERVED
			 1... RESERVED
			1.. RESERVED
			1. RESERVED
			1 RESERVED

FUNCTION CONTROL TABLE (FCT) CONT'D.

11(B)	1	FCTRSRC2	SYSTEM RESOURCE USAGE FLAG
		BIT NAME	RESOURCE
		-----	-----
		FCTLNREQ	XXXX SUB-FUNCTION LENGTH (BLKS)
			 1... RESERVED
			1.. RESERVED
			1. RESERVED
			1 FORTRAN I/O (IBCOM)
12(C)	4	FCTMCODE	ARGUMENT CODES
		BYTE 1	
		FCT2ALL	111. ALL STANDARD ARGUMENTS
		FCT2PROC	1111 PROCEDURE ARGUMENTS
		FCT2CDL	.XXX NUMBER OF ARGUMENTS (1-7)
		FCT2CD1	 XXXX CODE FOR ARGUMENT 1 (0-F)
		BYTE 2	
		FCT2CD2	XXXX CODE FOR ARGUMENT 2 (0-F)
		FCT2CD3	 XXXX CODE FOR ARGUMENT 3 (0-F)
		BYTE 3	
		FCT2CD4	XXXX CODE FOR ARGUMENT 4 (0-F)
		FCT2CD5	 XXXX CODE FOR ARGUMENT 5 (0-F)
		BYTE 4	
		FCT2CD6	XXXX CODE FOR ARGUMENT 6 (0-F)
		FCT2CD7	 XXXX CODE FOR ARGUMENT 7 (0-F)

JOSHUA DPSTM-500	Title CONTROL BLOCK: FUNCTION CONTROL TABLE	Vol.	Page
		IX	C.3.34
		Rev.	Date
		1	6/75

(Blank)

FUNCTION TIMING WORK AREA (FWK)

```

*****
*0(0)          *2(2)
*  FWKLNTH      *  FWKIDENT
*    LENGTH      *
*****
*
*                CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
*  FWKCODE
*
*                CURRENT FUNCTION CATEGORY CODE
*****
*12(C)
*  FWKBET
*
*                BEGINNING ELAPSED TIME
*****
*16(10)
*  FWKBWT
*
*                BEGINNING WAIT TIME
*****
*20(14)
*  FWKET
*
*                ELAPSED TIME FIELDS
*****
*64(40)
*  FWKWT
*
*                WAIT TIME FIELDS
*****
*108(6C)
*  FWKNR
*
*                FUNCTION REQUEST FIELDS
*****

```


FUNCTION TIMING WORK AREA (FWK)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	FWKLNTH	LENGTH OF CONTROL BLOCK
2(6)	6	FWKIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	4	FWKCODE	CURRENT FUNCTION CATEGORY CODE
12(C)	4	FWKBET	BEGINNING ELAPSED TIME OF CURRENT FUNCTION
16(10)	4	FWKBWT	BEGINNING WAIT TIME OF CURRENT FUNCTION
20(14)	44	FWKET	ELAPSED TIME ACCUMULATION FIELDS (ONE FIELD FOR EACH CATEGORY)
64(40)	44	FWKWT	WAIT TIME ACCUMULATION FIELDS (ONE FIELD FOR EACH CATEGORY)
108(6C)	11	FWKNR	FUNCTION REQUEST ACCUMULATION FIELDS (ONE FIELD FOR EACH CATEGORY)
119(77)	1		RESERVED

GENERAL CARD CONTROL (GCC)

```

*****
*0(0)
* GCCTAPE
* TAPE REQUEST
*****
*8(8)
* GCCINPT
* INPUT REQUEST
*****
*16(10)
* GCCDSOFT
* DATA SET DEFAULTS
*****
*48(30)
* GCCSCTH
* NAME TO BE SCRATCHED
*****
*56(38)
* GCCSCCDE
* AUTHORIZATION CODE
*****
*64(40)
* GCCVLSE
* VOL-SER INFO
*****
*104(68)
* GCCJOBVS
* JOB CARD SAVE
*****
*128(80)
* GCCDDPG
* FIRST DD PAGE SAVE
*****
*130(82)
* GCCIPPG
* FIRST INPUT CARD SAVE
*****
*132(84)
* GCCDDBPS
* BYPASS REQUEST
*****
*136(88)
* GCCHCQUE
* HOLDQUE REQUEST
*****

```

JOSHUA

DPSTM-500

Title CONTROL BLOCK:
GENERAL CARD CONTROL

Vol. IX

Page C.3.38

Rev. 1

Date 6/73

GENERAL CARD CONTROL (GCC)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	8	GCCTAPE	TAPE REQUEST
8(8)	8	GCCINPT	INPUT REQUEST
16(10)	32	GCCDSOFT	DATA SET DEFAULTS
48(30)	8	GCCSCTH	NAME TO BE SCRATCHED
56(38)	8	GCCSCCDE	AUTHORIZATION CODE
64(40)	40	GCCVLSEK	VOL-SER INFO
104(68)	24	GCCJOBVS	JOB CARD SAVE
128(80)	2	GCCDDPG	FIRST DD PAGE SAVE
130(82)	2	GCCIPIPG	FIRST INPUT CARD SAVE
132(84)	4	GCCDDBPS	BYPASS REQUEST
136(88)	4	GCCHDQUE	HOLDQUE REQUEST

GRAPHIC PARAMETERS ADDRESS LIST (GAL)

```

*****
*0(0)          *2(2)
*  GALLNGTH    *  GALIDENT
*    LENGTH    *
*****
*
*                CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
*  GALGACB      ADDRESS OF
*                GRAPHIC ATTENTION CONTROL BLOCK
*****
*12(C)
*  GALECBG      ADDRESS OF
*                ECB FOR ATTENTION HANDLING
*****
*16(10)
*  GALSPAR      ADDRESS OF
*                SET PROGRAM ATTENTION ROUTINE LIST
*****
*20(14)
*  GALDCBG      ADDRESS OF
*                GRAPHIC DATA CONTROL BLOCK
*****
*24(18)
*  GALDECBG     ADDRESS OF
*                DATA EVENT CONTROL BLOCK
*****
*28(1C)
*
*                RESERVED
*****
*32(20)
*
*                GRAPHIC ATTENTION CONTROL BLOCK (GACB)
*****
*88(58)
*
*                COMAREA - EXTENSION TO (GACB)
*****

```

JOSHUA DPSTM-500	Title CONTROL BLOCK: GRAPHIC PARAMETERS ADDRESS LIST	Vol. IX	Page C.3.40
		Rev. 1	Date 6/73

GRAPHIC PARAMETERS ADDRESS LIST (GAL) CONT'D.

*104(68) *
* *
* EVENT CONTROL BLOCK FOR ATTENTION HANDLING *

*108(6C) *
* *
* SET PROGRAM ATTENTION ROUTINE (SPAR) LIST *

*116(74) *
* *
* GRAPHIC DATA CONTROL BLOCK *

*168(A8) *
* *
* DATA EVENT CONTROL BLOCK *

GRAPHIC PARAMETERS ADDRESS LIST (GAL)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	GALLNGTH	LENGTH OF CONTROL BLOCK
2(2)	6	GALIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	4	GALGACB	ADDRESS OF GRAPHIC ATTENTION CONTROL BLOCK (GACB).
12(C)64	4	GALECBG	ADDRESS OF EVENT CONTROL BLOCK FOR ATTENTION HANDLING.
16(10)	4	GALSPAR	ADDRESS OF SET PROGRAM ATTENTION ROUTINE LIST.
20(14)	4	GALDCBG	ADDRESS OF GRAPHIC DATA CONTROL BLOCK.
24(18)	4	GALDECB	DATA EVENT CONTROL BLOCK.
28(1C)	4		RESERVED.
32(20)	56		GRAPHIC ATTENTION CONTROL BLOCK
88(58)	16		COMAREA - EXTENSION TO (GACB).
104(68)	4		EVENT CONTROL BLOCK FOR ATTENTION HANDLING.
108(6C)	8		SET PROGRAM ATTENTION ROUTINE LIST.
116(74)	52		GRAPHIC DATA CONTROL BLOCK.
168(A8)	32		DATA EVENT CONTROL BLOCK.

JOSHUA DPSTM-500	Title CONTROL BLOCK: GRAPHIC PARAMETERS ADDRESS LIST	Vol. IX	Page C.3.42
		Rev. 1	Date 6/73

(Blank)

MONITOR CONTROL BLOCK (MCB)

```

*****
*0(0)          *2(2)
* MCBLENGTH    * MCBIDENT
*      LENGTH   *
*****
*                                CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
* MCB$CB        ADDRESS OF
*                COMMUNICATION CONTROL BLOCK (CURRENT)
*****
*12(C)
* MCBCCB        ADDRESS OF
*                COMMUNICATION CONTROL BLOCK (FUTURE)
*****
*16(10)
* MCBMCF        ADDRESS OF
*                MONITOR CONTROL FLAGS
*****
*20(14)
* MCBARG        ADDRESS OF
*                SUB-FUNCTIONS ARGUMENT LIST
*****
*24(18)
* MCBOSL        ADDRESS OF
*                OS/360 PARAMETERS ADDRESS LIST
*****
*28(1C)
* MCBGAL        ADDRESS OF
*                OS/2260 (LOCAL) PARAMETERS ADDRESS LIST
*****
* MCBFCT        ADDRESS OF
*                FUNCTION CONTROL TABLE
*****
*32(20)
*****
*36(24)
* MCBCFCT       ADDRESS OF
*                CURRENT SUB-FUNCTION CONTROL TABLE
*****
*40(28)
* MCBBUFFG      ADDRESS OF
*                TERMINAL COMMUNICATION BUFFER
*****

```


JOSHUA DPSTM-500	Title CONTROL BLOCK: MONITOR CONTROL BLOCK	Vol. IX	Page C.3.44
		Rev. 1	Date 6/73

MONITOR CONTROL BLOCK (MCB) CONT'D.

```

*****
*44(2C)
* MCBBUFIN          ADDRESS OF
*                   TERMINAL INPUT BUFFER
*****
*48(30)
* MCBWCB            ADDRESS OF
*                   WORK CONTROL BLOCK
*****
*52(34)
* MCBRCB            ADDRESS OF
*                   REGION CONTROL BLOCK
*****
*56(38)
* MCBCT             ADDRESS OF
*                   COMMUNICATION CONTROL TABLE
*****
*60(3C)
* MCBDCBLB          ADDRESS OF
*                   DCB FOR MONITOR SYSTEM PROGRAM LIBRARY
*****
*64(40)
* MCBFWK            ADDRESS OF
*                   FUNCTION TIMING WORK AREA
*****
*68(44)
* MCBDSP10          ADDRESS OF
*                   TERMINAL COMMUNICATION INPUT/OUTPUT MANAGER
*****

```

MONITOR CONTROL BLOCK (MCB)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	MCBLNGTH	LENGTH OF CONTROL BLOCK
2(2)	6	MCBIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	4	MCB\$CB	ADDRESS OF COMMUNICATION CONTROL BLOCK (CURRENT)
12(12)	4	MCBCCB	ADDRESS OF COMMUNICATION CONTROL BLOCK (FUTURE)
16(16)	4	MCBMCF	ADDRESS OF MONITOR CONTROL FLAGS
20(20)	4	MCBARG	ADDRESS OF SUB-FUNCTION ARGUMENT LIST
24(24)	4	MCBOSL	ADDRESS OF OS/360 PARAMETERS ADDRESS LIST
28(28)	4	MCBGAL	ADDRESS OF OS/2260 (LOCAL) PARAMETERS ADDRESS LIST
32(32)	4	MCBFCT	ADDRESS OF FUNCTION CONTROL TABLE
36(36)	4	MCBCFCT	ADDRESS OF CURRENT SUB-FUNCTION CONTROL TABLE
40(40)	4	MCBBUFFG	ADDRESS OF TERMINAL COMMUNICATION BUFFER
44(44)	4	MCBBUFIN	ADDRESS OF TERMINAL INPUT BUFFER
48(48)	4	MCBWCB	ADDRESS OF WORK CONTROL BLOCK
52(52)	4	MCBRCB	ADDRESS OF REGION CONTROL BLOCK
56(56)	4	MCBCCT	ADDRESS OF COMMUNICATION CONTROL TABLE
60(60)	4	MCBDCBLB	ADDRESS OF DCB FOR MONITOR SYSTEM PROGRAM LIBRARY (MUST BE ZERO - THIS SPACE WILL BE RELEASED AFTER REMOVAL OF ALL REFERENCES TO THIS FIELD)
64(64)	4	MCBFWK	ADDRESS OF FUNCTION TIMING WORK AREA
68(68)	4	MCBDSPID	ADDRESS OF APPLICABLE TERMINAL COMMUNICATION INPUT/OUTPUT MANAGER

JOSHUA DPSTM-500	Title CONTROL BLOCK: MONITOR CONTROL BLOCK	Vol. IX	Page C.3.46
		Rev. 1	Date 6/73

(Blank)

MONITOR CONTROL FLAGS

```

*****
*0(0)          *2(2)
*  MCFLNGTH    *  MCFIDENT
*    LENGTH    *
*****
*                  CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
*  MCFSTATG
*                STATUS OF TERMINAL
*****
*9(9)
*  MCFREQ
*                MONITOR REQUEST
*****
*10(A)
*
*                RESERVED
*****

```

MONITOR CONTROL FLAGS (MCF)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	MCFLNGTH	LENGTH OF CONTROL BLOCK
2(2)	6	MCFIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	1	MCFSTATG	STATUS OF TERMINAL
		BIT NAME	
		MCFCNTRL	1... CONTROL TERMINAL
		MCFACTV	.1.. TERMINAL ACTIVE
		MCFOFFRQ	..1. TERMINAL OFFLINE REQUEST
		MCFOFFLN	...1 TERMINAL OFFLINE
			 1... RESERVED
			1.. RESERVED
			1. RESERVED
		MCFCCLASS	1 TERMINAL AUTHORIZED FOR ACCESS OF CLASSIFIED DATA
9(9)	1	MCFREQ	REQUEST FOR MONITOR TO:
		BIT NAME	
		MCFRSETF	1... RESET CURRENT SUB-FUNCTION POINTER
		MCFRSELF	.1.. SET UP NEW FUNCTION
			..1. RESERVED
		MCFDUMP	...1 TERMINATE WITH DUMP
			 1... RESERVED
		MCFLINK	1.. EXECUTE SUB-FUNCTION UNDER MONITOR TASK (LINK)
			1. RESERVED (PREVIOUSLY USED AS USER ENQ REQUEST)
			1 RESERVED (PREVIOUSLY USED AS USER DEQ REQUEST)
10(A)	6		RESERVED

JOSHUA OPSTM-500	Title CONTROL BLOCK: OS/360 PARAMETERS BLOCK	Vol. IX	Page C.3.49
		Rev. 1	Date 6/73

OS/360 PARAMETERS BLOCK (OSB) -----

```

*****
*0(0)          *2(2)
*  OSBLNGTH    *  OSBIDENT
*    LENGTH    *
*****
*                                CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
*  OSBDCBQ      ADDRESS OF DCB FOR
*                JOSHUA MONITOR SYSTEM QUEUE DATA SET
*****
*12(C)
*  OSBDCBC      ADDRESS OF DCB FOR
*                JOSHUA MONITOR SYSTEM COMMUNICATION DATA SET
*****
*16(10)
*                DCB (STARTING AT BYTE 16(10)) FOR
*                JOSHUA MONITOR SYSTEM QUEUE DATA SET
*****
*104(68)
*                DCB (STARTING AT BYTE 16(10)) FOR
*                JOSHUA MONITOR SYSTEM COMMUNICATION DATA SET  )*
*****

```

JOSHUA DPSTM-500	Title CONTROL BLOCK: OS/360 PARAMETERS BLOCK	Vol. IX	Page C.3.50
		Rev. 1	Date 6/73

OS/360 PARAMETERS BLOCK (OSB)

<u>OFFSET</u>	<u>BYTES</u>	<u>FIELD NAME</u>	<u>FIELD DESCRIPTION, CONTENTS, MEANING</u>
0(0)	2	RCFLNGTH	LENGTH OF CONTROL BLOCK
2(2)	6	RCFIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	4	OSBDCBQ	ADDRESS OF DCB FOR JOSHUA MONITOR SYSTEM QUEUE DATA SET
12(C)	4	OSBDCBC	ADDRESS OF DCB FOR JOSHUA MONITOR SYSTEM COMMUNICATION DATA SET
16(10)	88		DCB (FIRST 16 UNUSED BYTES NOT INCLUDED) JOSHUA MONITOR SYSTEM QUEUE DATA SET
104(68)	88		DCB (FIRST 16 UNUSED BYTES NOT INCLUDED) JOSHUA MONITOR SYSTEM COMMUNICATION DATA SET

OS/360 PARAMETERS LIST (OSL)

```

*****
*0(0)          *2(2)
*  USLLNGTH    *  OSLIDENT
*    LENGTH    *
*****
*                                CONTROL BLOCK IDENTIFIER
*
*****
*8(8)
*  OSLDECB     ADDRESS OF DECB
*                FOR COMMUNICATION DATA SET I/O
*****
*12(C)
*  OSLDECBQ    ADDRESS OF DECB
*                FOR QUEUE DATA SET I/O
*****
*16(10)
*
*                DECB FOR COMMUNICATION DATA SET I/O
*****
*44(2C)
*
*                DECB FOR QUEUE DATA SET I/O
*****
71(47)*
*****

```


JOSHUA

DPSTM-500

Title CONTROL BLOCK:
OS/360 PARAMETERS LIST

Vol. IX	Page C.3.52
Rev. 1	Date 6/73

OS/360 PARAMETERS LIST (OSL)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	PCLENGTH	LENGTH OF CONTROL BLOCK
2(2)	6	PCIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	4	OSLDECR0	ADDRESS OF DECB FOR COMMUNICATION DATA SET INPUT/OUTPUT.
12(C)	4	OSLDECBQ	ADDRESS OF DECB FOR QUEUE DATA SET INPUT/OUTPUT.
16(10)	28		DECB FOR COMMUNICATION DATA SET INPUT/OUTPUT.
44(2C)	28		DECB FOR QUEUE DATA SET INPUT/OUTPUT.

PARAMETER CONTROL BLCK (PCB)

```

*****
*0(0)
* PCBUSER
*
* USER DATA SET NAME
*****
*8(8)
* PCBJOB
*
* JOB DATA SET NAME
*****
*16(10)
* PCBMEMBR
*
* NAME OF MODULE TO BE EXECUTED
*****
*24(18)
* PCBGTIME
*
* ESTIMATED CPU TIME
*****
*32(20)
* PCBJSPCE
*
* NUMBER OF BLCKS REQUESTED FOR JOB DATA SET
*****
*40(28)
* PCBSSPCE
*
* NUMBER OF BLCKS REQUESTED FOR SCRATCH DATA SET
*****
*48(30)
* PCBRUFNC
*
* NUMBER OF DATA MANAGER BUFFERS
*****
*56(38)
* PCERLSE
*
* JOB DATA SET RELEASE OPTION
*****
*64(40)
* PCBDISP
*
* JOB DATA SET DISPOSITION OPTION
*****
*72(48)
* PCBTIMER
*
* TIMING OPTION
*****

```

JOSHUA DPSTM-500	Title CONTROL BLOCK: PARAMETER CONTROL BLOCK	Vol. IX	Page C.3.54
		Rev. 1	Date 6/73

PARAMETER CONTROL BLOCK (PCB) CONT'D.

```

*****
*80(50)
*  PCBADGE
*                BADGE NUMBER
*****
*88(58)
*  PCBET
*                ESTIMATED ELAPSED TIME
*****
*96(60)
*  PCBPATH
*                DATA MANAGER PATH NUMBER
*****
*104(68)
*  PCBDS1
*                FIRST READ DATA SET
*****
*112(70)
*  PCBDS2
*                SECOND READ DATA SET
*****
*120(78)
*  PCBDS3
*                THIRD READ DATA SET
*****
*128(80)
*  PCBDS4
*                FOURTH READ DATA SET
*****

```

PARAMETER CONTROL BLOCK (PCB)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	8	PCBUSER	USER DATA SET NAME.
8(8)	8	PCBJOB	JOB DATA SET NAME.
16(10)	8	PCBMEMB	NAME OF MODULE TO BE EXECUTED.
24(18)	8	PCBGTIME	ESTIMATED CPL TIME.
32(20)	8	PCBJSPCE	NUMBER OF BLOCKS REQUESTED FOR JOB DATA SET
40(28)	8	PCBSSPCE	NUMBER OF BLOCKS REQUESTED FOR SCRATCH DATA SET
48(30)	8	PCBRUFNO	NUMBER OF BUFFERS REQUESTED FOR DATA MANAGER USE
56(38)	8	PCERLSE	JOB DATA SET RELEASE OPTION. YES - RELEASE UNUSED JOB DATA SET SPACE AT END OF JCB. NO - RETAIN ALL JOB DATA SET SPACE.
64(40)	8	PCBDISP	JCB DATA SET DISPOSITION OPTION: NEW - REQUESTED JCB SPACE TO BE ALLOCATED AND JOB NAME ASSIGNED. OLD - JOB DATA SET ALREADY EXISTS. TMP - REQUESTED JCB SPACE TO BE ALLOCATED AND TEMPORARY JCB NAME ASSIGNED. JOB SPACE TO BE DEALLOCATED AT END OF JOB.
72(48)	8	PCBTIMER	TIMIN OPTION: YES - ACCUMULATE EXECUTION TIME OF ALL APPLICABLE ROUTINES. NO - BYPASS TIMING

JOSHUA

DPSTM-500

Title CONTROL BLOCK:
PARAMETER CONTROL BLOCK

Vol. IX	Page C.3.56
Rev. 1	Date 6/73

PARAMETER CONTROL BLOCK (PCB) CONT'D.

80(50)	8	PCBBADGE	BADGE NUMBER (FROM JOB CARD).
88(58)	8	PCBET	ESTIMATED ELAPSED TIME OF JOB.
96(60)	8	PCBPATH	PATH NUMBER USED BY DATA MANAGER.
104(68)	8	PCBDS1	FIRST INPUT DATA SET ACCESSED.
112(70)	8	PCBDS2	SECOND INPUT DATA SET ACCESSED.
120(78)	8	PCBDS3	THIRD INPUT DATA SET ACCESSED.
128(80)	8	PCBDS4	FOURTH INPUT DATA SET ACCESSED.

QUEUE CONTROL BLOCK (QCB)

QUEUE DATA SET

NAME	EQUIVALENT	FIELD DESCRIPTION, CONTENTS, MEANING
----	-----	-----
QCBQUE	164	BLOCK SIZE
QCBQCB	0	CATALOG RECORD NUMBER (NOT USED)
QCBFLAG	1	FLAG RECORD NUMBER
QCBFCL	2	FIRST RECORD OF FUNCTION CONTROL LIST (FCL)
QCBFCL2	5	SECND RECCRD OF FUNCTION CONTROL LIST (FCL)
QCBACB	3	RECORD NUMBER OF ACCOUNTING CONTROL BLOCK
QCBRCT	4	RECORD NUMBER OF READER CONTROL TABLE
QCBASAV	20	STARTING RECORD NUMBER FOR SAVING ACCOUNTING DATA
QCBASAVN	100	LAST RECORD NUMBER FOR SAVING ACCOUNTING DATA

BUFFER DATA SET

QCBPGQUE	960	BLOCK SIZE
QCBPG	10	STARTING RECORD NUMBER FOR SAVING BUFFERS
QCBPGPR	310	STARTING RECCRD NUMBER FOR SAVING PAGES FOR SUBSEQUENT PRINTING
QCBPGPRL	10	MAXIMUM NUMBER OF RECORDS ALLOWED PER TERMINAL

JOSHUA DPSTM-500	Title CONTROL BLOCK: QUEUE CONTROL BLOCK	Vol. IX	Page C.3.58
		Rev. 1	Date 6/73

(Blank)

JOSHUA

DPSTM-500

Title

CONTROL BLOCK:
READER CONTROL TABLE

Vol.
IX

Page
C.3.59

Rev.
1

Date
6/73

READER CONTROL TABLE (RCT)

```
*****
*0(0)                *2(2)                *
* RCTSTART           * RCTEND              *
*   FIRST RECORD NUMBER      *   LAST RECORD NUMBER      *
*****
*4(4)                *
* RCTCURF            *
*           CURRENT RECORD NUMBER          *
*****
*8(8)                *
* RCTLAST            *
*   LAST ACTIVE RECORD      *
*****
```


JOSHUA DPSTM-500	Title CONTROL BLOCK: READER CONTROL TABLE	Vol. IX	Page C.3.60
		Rev. 1	Date 6/73

READER CONTROL TABLE (RCT)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	RCTSTART	FIRST RECORD NUMBER OF THE QUEUE DATA SET ALLOCATED TO THE READER HOLDQUE
2(2)	2	RCTEND	LAST RECORD NUMBER OF THE QUEUE DATA SET ALLOCATED TO THE READER HOLDQUE
4(4)	4	RCTCURF	CURRENT RECORD NUMBER DURING INPUT/OUTPUT PROCESSING
8(8)	2	RCTLAST	LAST ACTIVE RECORD NUMBER OF THE READER HOLDQUE

REGION CONTROL BLOCK (RCB)

```

*****
*0(0)          *2(2)
* RCBNGTH      * RCBIDENT
* LENGTH      *
*****
*                CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
* RCBXCB      ADDRESS OF
*                EXECUTION CONTROL BLOCK
*****
*12(C)
* RCBRES      ADDRESS OF
*                RESIDENT MODULE LIST
*****
*16(10)
* RCBRCL      ADDRESS OF
*                REGION CONTROL LISTS
*****
*20(14)
* RCRCF      ADDRESS OF
*                REGION CONTROL FLAGS
*****
*24(18)
* RCBFCL      ADDRESS OF
*                FUNCTION CONTROL LIST
*****
*28(1C)
* RCBOSB      ADDRESS OF
*                OS/360 PARAMETERS BLOCK
*****
*32(20)
* RCBMSG      ADDRESS OF
*                MESSAGE FOR ALL TERMINALS
*****
*36(24)
* RCBPCB      ADDRESS OF
*                PARM CONTROL BLOCK
*****
*40(28)
* RCBDAT      ADDRESS OF
*                DYNAMIC ALLOCATION TABLE
*****

```

JOSHUA DPSTM-500	Title CONTROL BLOCK: REGION CONTROL BLOCK	Vol. IX	Page C.3.62
		Rev. 1	Date 6/73

REGION CONTROL BLOCK (RCB) CONT'D.

```

*****
*44(2C)                                     *
*  RCBCDAT           ADDRESS OF           *
*                   CURRENT DYNAMIC ALLOCATION TABLE ENTRY *
*****
*48(30)                                     *
*  RCBST            ADDRESS OF           *
*                   DATA MANAGER STATISTICS TABLE        *
*****
*52(34)                                     *
*                   NOT USED                               *
*****
  
```

REGION CONTROL BLOCK (RCB)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	RCBLNGTH	LENGTH OF CONTROL BLOCK
2(2)	6	RCBIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	4	RCBXCB	ADDRESS OF EXECUTION CONTROL BLOCK
12(C)	4	RCBRES	ADDRESS OF RESIDENT MODULE LIST
16(10)	4	RCBRCL	ADDRESS OF REGION CONTROL LIST
20(14)	4	RCBRCF	ADDRESS OF REGION CONTROL FLAGS
24(18)	4	RCBFCL	ADDRESS OF FUNCTION CONTROL LIST
28(1C)	4	RCROSE	ADDRESS OF OS/360 PARAMETERS BLOCK
32(20)	4	RCBMSG	ADDRESS OF MESSAGE FOR ALL TERMINALS
36(24)	4	RCBPCB	ADDRESS OF PARM CONTROL BLOCK
40(28)	4	RCBDAT	ADDRESS OF DYNAMIC ALLOCATION TABLE
44(2C)	4	RCBCDA	ADDRESS OF CURRENT DYNAMIC ALLOCATION TABLE ENTRY
48(30)	4	RCBBST	ADDRESS OF DATA MANAGER STATISTICS TABLE
52(34)	4		RESERVED

JOSHUA
DPSTM-500

Title CONTROL BLOCK:
REGION CONTROL BLOCK

Vol.
IX

Page
C.3.64

Rev.
1

Date
6/73

(Blank)

```
*****  
*0(0)                *2(2)  
*   RCFLNGTH          *   RCFIDENT  
*     LENGTH          *  
*****  
*                   CONTROL BLOCK IDENTIFIER  
*  
*  
*****  
*8(8)                *9(9)  
*   RCFKSRCL          *  
* RESOURCE FLAGS      *           RESERVED  
*****  
*12(C)               *15(F)*  
*   RCFSTAT           *  
*                   STATUS OF TERMINALS  
*****
```

JOSHUA DPSTM-500	Title CONTROL BLOCK: REGION CONTROL FLAGS	Vol. IX	Page C.3.66
		Rev. 1	Date 6/73

REGION CONTROL FLAGS (RCF)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	RCFLNGTH	LENGTH OF CONTROL BLOCK
2(2)	6	RCFIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	1	RCFRSRC1	SERIALY REUSABLE RESOURCE FLAGS
		1... ..	JOSHUA DATA MANAGER
		.1... ..	SIMULATED CARD READER
		..1.	JOSHUA MONITOR SYSTEM
			ACCOUNTING LOG
		...1	RESERVED
		 1...	RESERVED
		1..	RESERVED
		1.	RESERVED
		1	RESERVED
9(9)	3		RESERVED
12(C)	4	RCFSTAT	STATUS OF TERMINALS
		BYTE NAME	
12(C)		RCFALLOC	NUMBER OF TERMINALS ALLOCATED
13(D)		RCFACTV	NUMBER OF TERMINALS ACTIVE (NOT USED)
14(E)		RCFOFLIN	NUMBER OF TERMINALS OFFLINE (NOT USED)
15(F)			RESERVED

REGION CONTROL LIST (RCL)

```

*****
*0(0)          *2(2)
* RCLLNTH      * RCLIDENT
*   LENGTH     *
*****
*               CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
* RCLMAXT
*               MAXIMUM NUMBER OF TERMINALS
*****
*10(A)
* RCLNDX
*               RELATIVE POSITION IN ADDRESS LIST
*****
*12(C)
*
*               RESERVED
*****
*36(24)
* RCLWAITL     ADDRESSES OF
*               ALL MONITOR EVENT CONTROL BLOCKS
*****
*48(30)
* RCLCBL       ADDRESSES OF
*               TERMINAL OPERATION EVENT CONTROL BLOCKS
*****
*160(A0)
* RCLMCBL      ADDRESSES OF
*               TERMINAL OPERATION MONITOR CONTROL BLOCKS
*****
*272(110)
* RCLTCBL      ADDRESSES OF
*               TERMINAL OPERATION TASK CONTROL BLOCKS
*****
*384(180)
* RCLCBL       ADDRESSES OF
*               TERMINAL OPERATION EVENT CONTROL BLOCKS
*****
*496(1F0)
* RCLSPEC      SPECIAL EVENT CONTROL BLOCKS
*****

```


REGION CONTROL LIST (RCL)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	RCLNGTH	LENGTH OF CONTROL BLOCK
2(2)	6	RCLIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	2	RCLMAXT	MAXIMUM NUMBER OF TERMINALS
10(A)	2	RCLNDX	RELATIVE POSITION IN ADDRESS LIST
12(C)	24		RESERVED
36(24)		RCBWAITL	START OF MONITOR SYSTEM EVENT CONTROL BLOCKS LIST
36(24)	12	RCLSECBL	RESERVED FOR ADDRESSES OF SPECIAL EVENT CONTROL BLOCKS
48(30)	112	RCLECBL	ADDRESSES OF TERMINAL OPERATION EVENT CONTROL BLOCKS (ONE PER TERMINAL)
160(A0)	112	RCLMCBL	ADDRESSES OF TERMINAL OPERATION MONITOR CONTROL BLOCKS (ONE PER TERMINAL)
272(110)	112	RCLTCBL	ADDRESSES OF TERMINAL OPERATION TASK CONTROL BLOCKS (ONE PER TERMINAL)
384(180)	112	RCLECB	TERMINAL OPERATION EVENT CONTROL BLOCKS (ONE PER TERMINAL). EACH ECB CONTROLS THE MAJOR EVENT FOR THE ASSOCIATED TERMINAL
496(1F0)	12	RCLSECB	SPECIAL EVENT CONTROL BLOCKS

NOTE: RCLECBL, RCBMCBL, RCBTCBL AND RCBECEB ARE ALWAYS EQUAL IN LENGTH (4 TIMES RCBMAXT). THE SAME RELATIVE POSITION OF EACH REPRESENT THE SAME TERMINAL

JOSHUA DPSTM-500	Title	CONTROL BLOCK: RESIDENT MODULE LIST	Vol.	Page
			IX	C.3.69
			Rev.	Date
			1	6/73

RESIDENT MODULE LIST (RES)

```

*****
*0(0)          *2(2)
*  RESLNTH      *  RESIDENT
*    LENGTH      *
*****
*
*                CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
*  RESGRAM      ADDRESS OF
*                DATA MANAGER INTERFACE ROUTINE
*****
*12(C)
*  RESDSPLY     ADDRESS OF
*                TERMINALS DISPLAY MANAGER
*****
*16(10)
*  RESDSPLW     ADDRESS OF
*                OS/2260 (LOCAL) ATTENTION ROUTINE
*****
*20(14)
*  RESPAGER     ADDRESS OF
*                COMMUNICATION QUEUE PAGING MANAGER
*****
*24(18)
*  RESRPAGE     ADDRESS OF
*                COMMUNICATION QUEUE INPUT  MANAGER
*****
*28(1C)
*  RESWPAGE     ADDRESS OF
*                COMMUNICATION QUEUE OUTPUT MANAGER
*****
*32(20)
*  RESBLARG     ADDRESS OF
*                ARGUMENT LIST MANAGER
*****
*36(24)
*  RESDAIO      ADDRESS OF
*                DIRECT ACCESS INPUT/OUTPUT MANAGER
*****

```

RESIDENT MODULE LIST (RES)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	RESLNTH	LENGTH OF CONTROL BLOCK
2(2)	6	RESIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	4	RESCRAM	ADDRESS OF DATA MANAGER INTERFACE ROUTINE
12(C)	4	RESDSPLY	ADDRESS OF TERMINALS DISPLAY MANAGER
16(10)	4	RESDSPLW	ADDRESS OF OS/2260 (LOCAL) ATTENTION ROUTINE
20(14)	4	RESPAGER	ADDRESS OF COMMUNICATION QUEUE PAGING MANAGER
24(18)	4	RESRPAGE	ADDRESS OF COMMUNICATION QUEUE INPUT MANAGER
28(1C)	4	RESWPAGE	ADDRESS OF COMMUNICATION QUEUE OUTPUT MANAGER
32(20)	4	RESBLARG	ADDRESS OF ARGUMENT LIST MANAGER
36(24)	4	RESDAIO	ADDRESS OF DIRECT ACCESS INPUT/OUTPUT MANAGER

DSECT TX

PURPOSE : CONTAINS TEXT EDITOR CONTROL INFORMATION.

USAGE : SETUP LOCAL TO EACH EDITOR MODULE.
VARIABLES ZX THRU WRK20 ARE KEEP IN A JOSHUA AREA SUPPLIED TO EACH EDITOR ROUTINE WHILE VARIABLES RX AND ZP ARE STORED IN THE SECOND RECORD OF THE USER'S WORKSPACE.

VARIABLES :

MNEMONIC	TYPE	DESCRIPTION
----------	------	-------------

ZX	L*1	DIMENSIONED ZX(6). CONTAINS THE FOLLOWING: ZX(1) = TRUE IF USER'S WORK SPACE IS A JOSHUA USER DATASET. ZX(2) = TRUE IF USER'S WORK SPACE IS ALLOCATED. ZX(3) = TRUE IF THE GET DATASET IS ALLOCATEABLE. ZX(4) = TRUE IF THE WRITE DATASET IS ALLOCATEABLE. ZX(5) WORKING FIELD - EDITO. ZX(6) WORKING FIELD - EDITW EDITY EDITO.
FCODE	I*2	FUNCTION CODE FOR A JOSHUA SYSTEM ACTION: FCODE = 0 NO SYSTEM ACTION; FCODE = 1 DISPLAY THE TUBE BUFFER; FCODE = 2 SET WRITE ON THE USER'S WORK SPACE DATASET; FCODE = 4 NO SYSTEM ACTION; FCODE = 8 NO SYSTEM ACTION; FCODE = 16 NO SYSTEM ACTION; FCODE = 32 START THE INTERNAL READER ON THE COMMUNICATIONS DATASET; FCODE = 64 EITHER ALLOCATE OR DEALLOCATE TO THE EDITOR AN OS CATALOGED DATASET. FCODE = 128 GO BACK TO SELECT FRAME.
LMTCD	I*4	NUMBER OF PAGES AVAILABLE IN THE COMMUNICATIONS DATASET.
RCD77	I*4	BT77F001 BDAM RECORD NUMBER FOR USER'S TERMINAL.
CSECT	R*8	NAME OF MODULE BEING EXECUTED.
\$X	L*1	DIMENSIONED \$X(4). CONTAINS THE FOLLOWING: \$X(1) = TRUE IF IN THE 'R' MODE, AND ALSO WORKING FIELD - EDITO; \$X(2) WORKING FIELD - EDITX, EDITZ; \$X(3) = TRUE IF IN THE 'Q' MODE; \$X(4) NOT CURRENTLY USED.
XRNAM	L*1	NOT CURRENTLY USED.
XSIZE	I*1	NOT CURRENTLY USED.
MAXIO	I*2	MAXIMUM NUMBER OF TEXT EDITOR PAGES AVAILABLE.

JOSHUA DPSTM-500	Title CONTROL BLOCK: TEXT EDITOR CONTROL BLOCK (EDITOR)	Vol. IX	Page C.3.72
		Rev. 1	Date 6/73

NXTCS	I*4	INDEX OF NEXT TEXT EDITOR FUNCTION TO BE ATTACHED.
NXTBR	I*4	INTERNAL BRANCH INDEX TO BE USED BY THE NEXT TEXT EDIT MODULE ATTACHED.
LUNIT	I*4	LOGICAL UNIT NUMBER OF THE JOSHUA DATASET BEING USED AS THE USER'S TEXT EDITOR WORK SPACE.
LRCRD	I*4	RDAM RECORD NUMBER OF THE FIRST RECORD IN THE USER'S WORK SPACE.
LSTPG	I*4	NEXT AVAILABLE TEXT EDITOR PAGE NUMBER.
PRECS	I*4	CODE USED TO DETERMINE NEXT MODULE TO GET CONTROL.
WRK16	I*4	WORKING FIELD.
WRK17	I*4	WORKING FIELD.
WRK18	I*4	WORKING FIELD.
WRK19	I*4	WORKING FIELD.
WRK20	I*4	WORKING FIELD.
BX	L*1	DIMENSIONED BX(200). CONTAINS THE LOGICAL RECORD NUMBER OF THE CORRESPONDING TEXT EDITOR PAGE.
ZP	L*1	DIMENSIONED ZR(200). CONTAINS A LOGICAL TRUE IF THE PAGE IS CURRENTLY BEING USED BY THE TEXT EDITOR.

AUTHOR : R S WOOD

JOSHUA DPSTM-500	Title CONTROL BLOCK: TEXT EDITOR CONTROL BLOCK (MONITOR)	Vol. IX	Page C.3.73
		Rev. 1	Date 6/73

TEXT EDITOR CONTROL BLOCK (TEX)

```

*****
*0(0)
*  TEXTPAGE
*
*          PAGE REQUEST
*****
*4(4)
*  TEXREQ
*
*          EDITOR REQUEST CODE
*****
*8(8)
*  TEXPGLMT
*
*          NUMBER OF AVAILABLE PAGES
*****
*12(C)
*  TEXDSPSV
*
*          FIRST RECORD NUMBER FOR SAVING BUFFERS
*****
*16(10)
*  TEXMODUL
*
*          MODULE NAME TO BE EXECUTED
*****
*24(18)
*
*          RESERVED
*****
*76(4C)
*  TEXUNITE
*
*          TERMINAL IDENTIFICATION (EBCDIC)
*****

```

JOSHUA DPSTM-500	Title CONTROL BLOCK: TEXT EDITOR CONTROL BLOCK (MONITOR)	Vol. IX	Page C.3.74
		Rev. 1	Date 6/73

```

*****
*           CONTROL BLOCK:           *
*           TEXT EDITOR CONTROL BLOCK *
*                                     *
*                                     *
*****

```

TEXT EDITOR CONTROL BLOCK (TEX)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	4	TEXPAGE	PAGE REQUEST
4(4)	4	TEXREQ	EDITOR REQUEST CODE
		BYTE 1	RESERVED
		BYTE 2	RESERVED
		BYTE 3	RESERVED
		BYTE 4	
		BIT NAME	
		TEXEXIT	1... EXIT FROM EDITING MODE
		TEXDADR	.1.. DIRECT ACCESS DYNAMIC REQ
		TEXEXEC	..1. SUBMIT JOB STREAM FOR EXEC
			...1 RESERVED
			 1... RESERVED
			1.. RESERVED
		TEXSETWR	1. VERIFY UPDATE AUTHORIZATION
		TEXDSPLY	1 DISPLAY BUFFER
8(8)	4	TEXPCLMT	NUMBER OF AVAILABLE PAGES
12(C)	4	TEXDSPSV	FIRST RECORD NUMBER OF DATA SET USED FOR SAVING BUFFERS
16(10)	4	TEXMODUL	NAME OF MODULE TO BE EXECUTED
24(18)	52		RESERVED
76(4C)	4	TEXUNITE	TERMINAL IDENTIFICATION (EBCDIC)

JOSHUA DPSTM-500	Title CONTROL BLOCK: USER IDENTIFICATION BLOCK	Vol. IX	Page C.3.75
		Rev. 1	Date 6/73

USER IDENTIFICATION BLOCK (UIB)

```

*****
*0(0)
*  UIBFILID
*
*          FILE IDENTIFICATION
*
*          *****
*          *6(6)
*          *  UIBBADGE
*          *    USER BADGE NUMBER
*
*          *****
*          *10(A)
*          *  UIBUID
*          *    USER IDENTIFICATION
*
*          *****
*
*
*
*****

```


JOSHUA DPSTM-500	Title CONTROL BLOCK: USER IDENTIFICATION BLOCK	Vol. IX	Page C.3.76
		Rev. 1	Date 6/73

USER IDENTIFICATION BLOCK (UIB)

<u>OFFSET</u>	<u>BYTES</u>	<u>FIELD NAME</u>	<u>FIELD DESCRIPTION, CONTENTS, MEANING</u>
0(0)	6	UIBFILID	FILE IDENTIFICATION
6(6)	4	UIBBADGE	USER BADGE NUMBER
10(A)	6	UIBUID	USER IDENTIFICATION
16(10)			REPEAT 'UIBBADGE' AND 'UIBUID' FOR EACH ADDITIONAL USER

WORK CONTROL BLOCK (WCB)

```
*****
*0(0)          *2(2)
* WCBLENGTH    * WCBIDENT
*      LENGTH  *
*****
*                                CONTROL BLOCK IDENTIFIER
*
*
*****
*8(8)
* WCBFCTNM
*                                FUNCTION REQUEST CODE
*****
*16(10)
* WCBWXT        ADDRESS OF
*                                WORK EXTENSION TABLE
*****
*20(14)
* WCBECB        SUB-FUNCTION COMPLETION CODE
*****
*24(18)
* WCBWORK       WORK AREA
*****
*32(20)         *33(21)
* WCBUNITB      * WCBUNITE
* TERMINAL ID (8) * TERMINAL IDENTIFICATION (EBCDIC)
*****
*36(24)         *38(26)
* WCBFSTR       * WCBFPRGC
* 'SIGN IN' FUNCTION CODE      * 'EXEC PROC' FUNCTION CODE
*****
*40(28)
* WCBARG1       FUNCTION ARGUMENT 1 (ALIGNED)
*****
*184(188)
* WCBARG2       FUNCTION ARGUMENT 2 (ALIGNED)
*****
*328(148)
* WCBARGS1      FUNCTION ARGUMENT 1 SAVE AREA
*****
*400(190)
* WCBARGS2      FUNCTION ARGUMENT 2 SAVE AREA
*****
```

JOSHUA DPSTM-500	Title CONTROL BLOCK: WORK CONTROL BLOCK	Vol. IX	Page C.3.78
		Rev. 1	Date 6/73

WORK CONTROL BLOCK (WCB) CONT'D

```

*****
*472(1D8)
* WCBFNM
*
* FUNCTION REQUEST CODE SAVE AREA
*****
*480(1E0)
* WCBBLNKS
*
* BLANKS
*****
*488(1E8)
* WCBUSER
*
* USER DATA SET NAME
*****
*492(1EC) *493(1ED) *494(1EE)
* WCBRSRC * WCBSPREQ * WCBDSPLS
* RESOURCE FLAGS * RESOURCE USAGE * NUMBER OF DISPLAYS
*****
*496(1F0) *497(1F1) *498(1F2) *499(1F3)
* WCBPRTPG * WCBPRTSV * WCBICLRQ * WCBICLDW
*DISPLAYS PRINTED* DISPLAYS SAVED * INVALID 'SIGN IN' REQUESTS
*****
*500(1F4)
* WCBMSG
* ADDRESS OF
* MESSAGE FOR THIS TERMINAL
*****
*504(1F8)
* WCBNQDN
*
* NAME OF DATA SET AUTHORIZED FOR UPDATING
*****

```

WORK CONTROL BLOCK (WCB)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	2	WCBLENGTH	LENGTH OF CONTROL BLOCK
2(2)	6	WCBIDENT	IDENTIFICATION OF CONTROL BLOCK
8(8)	2	WCBFCTNM	FUNCTION REQUEST CODE
10(A)	6		RESERVED
16(10)	4	WCBWXT	ADDRESS OF WORK EXTENSION TABLE
20(14)	4	WCBECB	SUB-FUNCTION COMPLETION CODE
24(18)	8	WCBWORK	WORK AREA (DOUBLE WORD)
32(20)	1	WCBUNITB	TERMINAL IDENTIFICATION (BINARY)
33(21)	3	WCBUNITC	TERMINAL IDENTIFICATION (EBCDIC)
36(24)	2	WCBFSTRT	'SIGN IN' FUNCTION CODE
38(26)	2	WCBFPROC	'EXECUTE PROCEDURE' FUNCTION CODE
40(28)	144	WCBARG1	FUNCTION ARGUMENT 1 (ALIGNED ON DOUBLE WORD BOUNDARIES)
184(88)	144	WCBARG2	FUNCTION ARGUMENT 2 (ALIGNED ON DOUBLE WORD BOUNDARIES)
328(148)	72	WCBARGS1	FUNCTION ARGUMENT 1 SAVE AREA
400(190)	72	WCBARGS2	FUNCTION ARGUMENT 2 SAVE AREA

JOSHUA DPSTM-500	Title CONTROL BLOCK: WORK CONTROL BLOCK	Vol. IX	Page C.3.80
		Rev. 1	Date 6/73

WORK CONTROL BLOCK (WCB) CONT'D

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
472(1D8)	8	WCBFNM	FUNCTION REQUEST CODE SAVE AREA
480(1E0)	8	WCBBLANKS	BLANK FIELD (DOUBLE WORD BOUNDARY)
488(1E8)	4	WCBUSER	USER DATA SET NAME
492(1EC)	1	WCBRSRC	REUSABLE RESOURCES ASSIGNED TO SUB-FUNCTION
493(1ED)	1	WCBSPREQ	
494(1EE)	2	WCBDSPLS	NUMBER OF DISPLAYS DURING TERMINAL SESSION
496(1F0)	1	WCBPRTPG	NUMBER OF DISPLAYS PRINTED
497(1F1)	1	WCBPRTSV	NUMBER OF DISPLAYS BEING SAVED FOR PRINT
498(1F2)	1	WCBICLRQ	NUMBER OF INVALID 'SIGN IN' REQUESTS PER 'SIGN IN'
499(1F3)	1	WCBICLDW	NUMBER OF INVALID 'SIGN IN' REQUESTS PER TERMINAL
500(1F4)	4	WCBMSG	ADDRESS OF MESSAGE FOR THIS TERMINAL
504(1F8)	8	WCBNQDQN	NAME OF DATA SET AUTHORIZED FOR UPDATE

WORK EXTENSION TABLE (WXT)

```

*****
*U(0)
* WXTWORK1
* 20 WORD WORK AREA
*****
*80(50)
* WXTWORK2
* 64 HALF WORD WORK AREA
*****
*208(D0)
* WXTWORK3
* 328 CHARATER WORK AREA
*****
*536(218)
* WXTWORK4
* 512 CHARATER WORK AREA
*****
*1048(418)
* WXTZERO
* INTEGER ZERO
*****
*1052(41C)
* WXTUNE
* INTEGER ONE
*****
*1056(420)
* WXTNTGR
* INTEGER FIELD
*****
*1060(424)
* WXTDSRF5
* RESERVED
*****
*1064(428)
* WXTDSRF6
* RELATIVE RECORD NUMBER OF TERMINAL
*****
*1068(42C)
* RESERVED
*****

```

JOSHUA DPSTM-500	Title CONTROL BLOCK: WORK EXTENSION TABLE	Vol. IX	Page C.3.82
		Rev. 1	Date 6/73

WORK EXTENSION TABLE (WXT) CONT'D.

```

*****
*1072(430)
* WXTNAME
*          PROCEDURE NAME SAVE AREA
*****
*1088(440)
* WXTFLAG
*          PROCEDURE FLAGS
*****
*1092(444)
* WXTSTEP
*          PROCEDURE STEP NUMBER
*****
*1096(448)
* WXTSAVE
*          PROCEDURE SAVE AREA
*****

```

WORK EXTENSION TABLE (WXT)

OFFSET	BYTES	FIELD NAME	FIELD DESCRIPTION, CONTENTS, MEANING
0(0)	80	WXTWORK1	20 WORD WORK AREA
80(50)	128	WXTWORK2	64 HALF WORD WORK AREA
208(00)	328	WXTWORK3	328 BYTE WORK AREA
536(218)	512	WXTWORK4	512 BYTE WORK AREA
1048(418)	4	WXTZERO	INTEGER ZERO (0)
1052(41C)	4	WXTONE	INTEGER ONE (1)
1056(420)	4	WXTNTR	NUMBER OF QUALIFIERS IN ARGUMENT 1 FIELD
1060(424)	4		RESERVED
1064(428)	4	WXTDSRF6	RELATIVE RECORD NUMBER OF TERMINAL
1068(42C)	4		RESERVED
1072(430)	16	WXTNAME	PROCEDURE NAME SAVE AREA
1088(440)	4	WXTFLAG	PROCEDURE FLAG 0 - PROCEDURE NOT ACTIVE 1 - PROCEDURE TO BE EXECUTED 2 - RETURNED FROM PROCEDURE
1092(444)	4	WXTSTEP	PROCEDURE STEP NUMBER 0 - PROCEDURE NOT IN USE X - STEP NUMBER
1096(448)	152	WXTSAVE	PROCEDURE SAVE AREA

JOSHUA DPSTM-500	Title CONTROL BLOCK: WORK EXTENSION TABLE	Vol. IX	Page C.3.84
		Rev. 1	Date 6/7

(Blank)

5

— — — — —

```

*****
*          FREEBUFFG
*
***  RELEASE THE DISPLAY BUFFER
      L      R1,MCBBUFFG          -> TERMINAL DISPLAY BUFFER
      LTR    R1,R1                IS THER A Cisplay BUFFER?
      BZ     FREEBUFN             NC - BRANCH
      L      R0,FBUFARG           SUBPOOL AND BUFFER LENGTH
      FREEMAIN R,LV=(R0),A=(R1)  RELEASE Cisplay BUFFER
      XC     MCBBUFFG(4),MCBBUFFG SET DISPLAY BUFFER ADDRESS TO ZERO
      B      FREEBUFN
FBLFARG  DS      OF
DC       AL1(MCBSP),AL3(BCLBUFL)  SUBPOOL & BUFFER LENGTH
FREEBUFN DS      OH
*
*****

```

JOSHUA DPSTM-500	Title	SOURCE CODE BLOCKS: GETBUFFG	Vol.	Page
			IX	C.3.86
			Rev.	Date
			1	6/7

GETBUFFG

```

*****
*          GETBUFFG
*
*** GET TERMINAL DISPLAY BUFFER
    L      R1,MCBBUFFG      -> DISPLAY BUFFER
    LTR    R1,R1            IS DISPLAY BUFFER AVAILABLE?
    BNZ    GETBUFN          YES - BRANCH
    L      R0,GBUFARG        SUBPOOL AND BUFFER LENGTH
    GETMAIN R,LV=(R0)        GET MAIN AREA FOR DISPLAY BUFFER
    ST     R1,MCBBUFFG      SAVE ADDRESS OF DISPLAY BUFFER
    MVI    000(R1),C' '      BLANK
    MVC    001(080,R1),000(R1)      1ST &
    MVC    880(080,R1),000(R1)      LAST LINES
    B      BLNKBUFG
GBUFARG DS    OF
DC      AL1(MCBSP),AL3(BCLBUFL) SUBPOOL & BUFFER LENGTH
GETBUFN DS    OH
* BLANK TERMINAL DISPLAY BUFFER
BLNKBUFG DS    OH
    L      R1,MCBBUFFG      -> TERMINAL DISPLAY BUFFER
    MVI    080(R1),C' '      BLANK
    MVC    081(159,R1),080(R1)
    MVC    240(240,R1),080(R1)      DISPLAY
    MVC    480(240,R1),080(R1)
    MVC    720(160,R1),080(R1)      BUFFER
*
*****

```

— — — — —

```

*****
*          NORMEXIT          *
*** SET UP NORMAL EXIT      ***
NORMEXIT DS      OF          *
          SR      R15,R15    SET NORMAL COMPLETION CODE      *
          B       RETURN     TO RESTORE REGISTERS AND RETURN  *
*                               *
*****

```

JOSHUA DPSTM-500	Title SOURCE CODE BLOCKS: REGUSAGE	Vol. IX	Page C.3.88
		Rev. 1	Date 6/73

REGUSAGE

```

*****
*          REGUSAGE
*
*****
*  REGISTER USAGE
R0      EQU    0      STANDARD GENERAL REGISTER USAGE
R1      EQU    1      STANDARD GENERAL REGISTER USAGE
R14     EQU    14     STANDARD GENERAL REGISTER USAGE
R15     EQU    15     STANCARC GENERAL REGISTER USAGE
*
MCB      EQU    2      -> MONITOR CONTROL BLOCK      (MCB)
*
*****

```

SAVER

```

*****
*          SAVER                                     *
*
*          EJECT                                     *
*****
*** ENTRY                                           ***
* REENTRANT LINKAGE                                *
*   USING *,12                                     *
*   STM 14,12,12(13)      SAVE REGISTERS           *
*   LR  12,15             LOAD BASE REGISTER       *
*   LR  14,13             SAVE PCINTER TO PREVIOUS SAVE AREA *
*   L   0,GARG             SET UP ARGUMENT FOR GETMAIN      A*
*   GETMAIN R,LV=(0)       GET MAIN STORAGE AREA FOR SAVE AREA*
*   LR  13,1              -> TO CURRENT SAVE AREA          *
*   LM  0,1,20(14)        RESTORE PARAMETER REGISTERS      *
*   B   S13               *
*   DS   OF               *
GARG DC AL1(16),AL3(72)   SUBPOOL AND LENGTH FOR GETMAIN *
RETURN DS OH              *
*   LR  1,13              -> CURRENT SAVE AREA FOR FREEMAIN *
*   L   13,4(13)          -> PREVIOUS SAVE AREA            *
*   L   0,GARG             SUBPOOL AND LENGTH FOR FREEMAIN *
*   FREEMAIN R,LV=(0),A=(1) FREE THE SAVE AREA           *
*   L   14,12(13)         RETURN ADDRESS               *
*   LM  2,12,28(13)       RESTORE REGISTERS             *
*   BR  14                RETURN TO PREVIOUS MODULE      *
S13 ST 13,8(14)           SAVE ADDR OF CURRENT SAVE AREA *
*   ST  14,4(13)          SAVE ADDR OF PREVIOUS SAVE AREA *
*   SPACE 1              *
*****
* REGISTER CONTENTS UPON ENTRY                       *
* REGISTER      DESCRIPTION                          *
*   1          -> ARGUMENT LIST (CCB)                *
*   WHERE: CCB -> COMMUNICATION CONTROL BLOCK        *
*   SPACE 1                                           *
*   USING CCBDSCT,CCB      ESTABLISH ADDRESSABILITY OF (CCB) *
*   USING MCBDSCT,MCB      ESTABLISH ADDRESSABILITY OF (MCB) *
*   SPACE 1                                           *
* SET UP POINTER TO MONITOR CONTROL BLOCK (MCB)      *
*   L   CCB,0(R1)          -> COMMUNICATION CONTROL BLOCK (CCB) *
*   L   MCB,CCBMCB         -> MONITOR CONTROL BLOCK (MCB)      *
*
*****

```

JOSHUA DPSTM-500	Title SOURCE CODE BLOCKS: SAVES	Vol. IX	Page C.3.90
		Rev. 1	Date 6/73

SAVES

```

*****
*          SAVES                                     *
*          EJECT                                     *
*****
*** ENTRY                                           *
*  STANDARD LINKAGE                                *
*      USING *,12          ESTABLISH ADDRESS ABILITY *
*      STM  14,12,12(13)   SAVE REGISTERS           *
*      LR   12,15          LOAD BASE REGISTER        *
*      LR   14,13          SAVE ADDRESS OF PREVIOUS SAVE AREA *
*      RAL  13,NULLOCC1    *
*      DC   3F'C'          *
*      DS   15F            *
RETURN  L    13,4(0,13)    LOAD ADDRESS OF PREVIOUS SAVE AREA *
        L    14,12(0,13)   LOAD RETURN ADDRESS           *
        LM   2,12,28(13)   RESTORE REGISTERS             *
        PR   14            RETURN TO PREVIOUS MODULE      *
NULLOCC1 ST  13,8(0,14)    SAVE ADDRESS OF CURRENT SAVE AREA *
        ST   14,4(0,13)    SAVE ADDRESS OF PREVIOUS SAVE AREA *
*
*****

```

JOSHUA

DPSTM-500

Title Edit Facilities

Vol.
IX

Page
C.4.1

Rev.

Date
6/73

Work Done By

R. S. Wood

Report Written By

R. S. Wood

Contents

Page

1. Introduction

C.4.3

JOSHUA

DPSTM-500

Title Edit Facilities

Vol.
IX

Page
C.4.2

Rev.
1

Date
6/73

(Blank)

1. Introduction

Edit facilities are provided to terminal users via the EDITOR function, which operates as a subsystem under control of the JOSHUA Operating System. This subsystem gives users the ability to create, modify, and delete card images which are kept in sequential or partitioned OS data sets. In using the EDITOR, the user "GETS" card images from an OS library, works out of separate workspace contained in a JOSHUA record, and at completion "WRITES" the modified card images back to an OS data set. It is the EDITOR user's responsibility to identify the "GET" and "WRITE" libraries and also the JOSHUA record used for editing. The following are the major EDITOR commands available:

- (1) Add card images;
- (2) Copy card images from one place to another;
- (3) Delete card images;
- (4) Get card images from an OS library;
- (5) Look for various card images;
- (6) Move card images from one place to another;
- (7) Query the file for card images containing a specified field;
- (8) Update card images in place;
- (9) Write card images back to an OS file;
- (10) Execute a batch job from the JOSHUA record;
- (11) Perform cleanups on OS partitioned data set libraries; also print and/or punch the cards images in the OS libraries.

JOSHUA

DPSTM-500

Title Edit Facilities

Vol.
IX

Page
C.4.4

Rev.
1

Date
6/73

(Blank)