### Voyager Processing Interface

**EDRLOG**
- FOR... does all its own interface

**ENDOAY**
- ECP... open FOR block
- ECL... open library block
- ECLOET... close the end of library block
- EC... close FOR block
- ECL... close library block

**ENGQEN**
- LOL... open library
- LQO... open block
- LQO... close open
- LQ... close library
- LTERM... term block

**WRACCT**
- WCOW... open screen
- WCOC... open CRT
- WCCO... close CRT
- WCCL... close CRT

**MEA48**

### Utilities

<table>
<thead>
<tr>
<th>LREAD (L.WRITE)</th>
<th>READ... in block... LINKED</th>
<th>EOPROL...</th>
<th>RDIR...</th>
<th>ELSER...</th>
<th>DIRSAT...</th>
<th>ETSER...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
III. INITIALIZER

INLOG  Initialize log
BLOCKS  Block data

IV. MAINTENANCE

ALTER   Alter the existing log
REDR   Redo log
RWORK   Rework work done
RCIT    Recite log data
ASSIGN   Assign work to user
CITATE   Cite data
CITARY   Cite data

V. LIST FEATURES

CALLIST  Call the list
GAXLOG   GAX log
CITAMV   CITAMV
DMADV  Domain
EDRAMV  EDMV
INDSIE  Index
LISTAL  List all
LIST   List
LATAN   List data
LEOD   Log data
LENC   Length
LICH   List data
LUXH   List data
RESULT  Result
SHUOT   Show
SHUOR   Show

IV. MAINTENANCE

ALTER   Alter the existing log
REDR   Redo log
RWORK   Rework work done
RCIT    Recite log data
ASSIGN   Assign work to user
CITATE   Cite data
CITARY   Cite data

V. LIST FEATURES

CALLIST  Call the list
GAXLOG   GAX log
CITAMV   CITAMV
DMADV  Domain
EDRAMV  EDMV
INDSIE  Index
LISTAL  List all
LIST   List
LATAN   List data
LEOD   Log data
LENC   Length
LICH   List data
LUXH   List data
RESULT  Result
SHUOT   Show
SHUOR   Show

The document appears to be a manual or guide for an automatic log system with sections on initialization, maintenance, and listing features. The entries seem to be instructions or commands for operating the system.
MSS Automatic Log System

VI. User's Guides

- INTLOG
- ALTBLK
- TH3 update

VII. Appendix

- COMMON BLOCKS
- LOGDAT
- M3S ELOG BLOCK Formats
The Voyager Log production involves five procedures, as diagrammed on the following page. All five procedures interface with the automated log which keeps track of tapes and the status of the production cycle.

A tape is introduced to the Voyager Log processing system through EDRLG. The user inputs the tape volume into the log. EDRLG requests that the next process, ENRSANU, be done for this tape and returns the T5S plate number assigned the new EDR tape.

ENRSANU expects the 8000 EDR tape onto a file on a Library tape (6250 bpi). ENRSANU requests that the next process, ENRGEN, be run for this file, if ENRSANU successfully processes the EDR tape.

ENRGEN reformat the EDR data into encyclopedia tapes with volumes, chapters, and verses. ENRGEN, if successful, requests that
the MERGE and WRKETL processes be done on this work tape. The library files are processed in data, time, order.

The MERGE process merges the work tape data into encyclopedia tapes.

The WRKETL process writes two tapes to be sent to CUT, raw rates and all but raw rates.

These procedures interface with the Log for basically two purposes: 1) to open a Log block or 2) to close a Log block.

An open may be either finding a block to process or allocating and partially filling a new block.

A close updates an old block, mark processes done, WR writes the new block, mark to process and link with the previous old block.

The sources of the interface routines are located in the source library of their respective procedure routines, i.e. EOE, EOL, ECE, ECU
and ECCEOT are with FORSAV.

See the vendor guides for this

Since there are interface subroutines they do not have individual user's guides other than their own documentation.
1. Subroutine EOE

2. Function
   Subroutine EOE (EDR processing – open EDR) locates the next
   EDR block to be processed.

3. Programmer and Date
   Bunice Eng, September 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for EOE is as follows:

   CALL EOE(EDRBLK,RTCODE)

   The arguments in the calling sequence are described below:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDRBLK(32)</td>
<td>I*2</td>
<td>0</td>
<td>EDR block</td>
</tr>
<tr>
<td>RTCODE</td>
<td>I*4</td>
<td>0</td>
<td>Return code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0, EDR block found</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 4, No EDR block found for processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 8, error encountered</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called
   EOE calls Subroutine RDIRET. EOE calls SACC-BATH subroutines
   DRREAD and SACC function subroutine LDR. KEC and KUL.

4.3 COMMON Variables
   EOE updates LEDDBK, EDRSER, EDRDSN, and LEDRSL of
   COMMON LOGDAT. IMES(26) of COMMON FERMSG is used to
   write and I/O error message.

   EDRSER = JPL number
   EDRDSN = CRS
Set Return Code, RTCODE = 0

Get to EDR blocks

Read Directory block

If (satellite block @ not found, i.e. RTCODE = 8)

EXIT to RETURN

Read Satellite block

Read EDR Control block

Do Until (all EDR blocks processed, i.e. next EDR @ = 0)

Read EDR block

Get address of next EDR block

If (Entry marked for processing)

Mark processing begun

Mark next entry to be processed in case present processing is not successfully completed

Write EDR block

Update COMMON LOGDAT

IEDRBK
EDRSER
EDRDSN
IEDRSL
HEDRCH

EXIT to RETURN

Od

Write Error message

RTCODE = 4

RETURN

Note: If an I/O error is encountered during DREAD

Write error message

RTCODE = 8

EXIT to RETURN
6.3 Special Notes

6.2 Major Variables and Constants

<table>
<thead>
<tr>
<th>Variable or Constant</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QMSKTP</td>
<td>L#1</td>
<td>Constant = X'80', flags 'to process'</td>
</tr>
<tr>
<td>QMSKPR</td>
<td>L#1</td>
<td>Constant = X'40', flags 'process in progress'</td>
</tr>
<tr>
<td>NSTBLK</td>
<td>I#4</td>
<td>Address of next EDR block</td>
</tr>
<tr>
<td>LASTEN</td>
<td>I#4</td>
<td>Address of last entry in EDR block</td>
</tr>
</tbody>
</table>

6.3 Data Structure and Tables
See Table in appendix for format of EDR block.

6.3 Error Handling
If EOE encounters a serious error a return code of 8 is passed to the calling routine. For errors see Subroutine REDIT.

If EOE does not locate an EDR block to be processed, the following message is written and a *** return code of 4 is passed to the calling routine.

EOE *** NO BLOCKS FOUND MARKED FOR PROCESSING.

If an I/O error is encountered during a DREAD, the following message is printed.

EOE *** I/O ERROR ENCOUNTERED.

Followed by the data passed in COMMON FERMSG, described in SACC DAIO document. A */ return code of 4 is also passed to the calling routine.

6.4 Other
EOE must be compiled under PORTRAN with parm=XL for function LOR to be expanded during compilation time.
1. Subroutine EOL.

2. Function
Subroutine EOL (EDR processing - open Library block) allocates
a new library block.

3. Programmer and Date
Enicce Eng, September 1977

4. Interface

4.1 Calling Sequence
The FORTRAN calling sequence for EOL is as follows:

CALL EOL(LIBBLK,RTCODE)

The arguments in the calling sequence are described below:

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIBBLK(32)</td>
<td>I*2</td>
<td>0</td>
<td>Library block</td>
</tr>
</tbody>
</table>
| RTCODE     | I*4  | 0   | Return Code
            |      |     | = 0, block allocated           |
            |      |     | = 4, block not allocated        |
            |      |     | = 8, error encountered          |

4.2 Subroutines Called
EOL calls Subroutines RDIRET, EWRHGS and EOPROL. EOL also
calls SAVDATA subroutine DREAD.

4.3 COMMON Variables
EOL updates ILIBBK, LIBSER, I1SLOT, INSLOT, LIBDSN, HLIBSQ, HDFEB
and HMXPT of COMMON LOGDAT. IMES(26) of COMMON PERMSG
is used to write an I/O error message.

\[
\text{LIBSER} = M_{12}^{11}X
\]

\[
\text{LIBDSN} = V_{1}^{22}AAXXXX
\]
Set Return Code, RTCODE = 0
Read Directory
If (satellite block address not found; i.e. RTCODE = 3)
   EXIT to RETURN
Get address of next block to be allocated
If (next block to be allocated address exceeds data set)
   Write error message
   RTCODE = 4
   EXIT to RETURN
Read Satellite Block
Read Library Control block
   Save @ of last allocated Library block
   Update COMMON LOGDAT
      LIBSER
      LIBDSN
      HLESER
      HLIBS4
      HFEET
      HMXFT
      I1SLOT
      INSLOT
   Clear and Write new Library block
   Update COMMON LOGDAT
      ILIBBK - present lib. block, new
   Update and Write Directory
RETURN

Note: if I/O error during DREAD
   Write error message
   RTCODE = 3
   EXIT TO RETURN
6. Special Notes

6.1 Major Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HNXTAV</td>
<td>I*2</td>
<td>Next available block number</td>
</tr>
<tr>
<td>ILIBLS</td>
<td>I*4</td>
<td>Number of the current last Library block.</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Tables in Appendix for format of Directory block and Library block.

6.3 Error Handling

If EOL encounters a serious error, a return code of 8 is passed to the calling routine. See Subroutine RDIFST for serious errors.

If EOL discovers it cannot allocate the next Library block due to a filled log, the following message is written and a return code of 4 is passed to the calling routine.

EOL ***** NEXT AVAILABLE BLOCK NUMBER EXCEEDS XXXXXXXXXXX.

If an I/O error occurs during a DREAD (called subroutine), a return code of 4 is set and the following message is written before a % return is taken.

EOL ***** I/O ERROR ENCOUNTERED.

Followed by data from COMMON FERMFG. The I/O data is described in DAIO documentation.
1. Subroutine ECLEOT

2. Function
   Subroutine ECLEOT (EDP processing -end of tape) declares the present library tape full.

3. Programmer and Date
   Enice Eng, September 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence is as follows:

   CALL ECLEOT(RTCODE)

   The calling argument is described below.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCODE</td>
<td>I*4</td>
<td>0</td>
<td>Return Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0, next vol-ser available</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 8, next vol-ser out of range</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called
   Subroutine ECLEOT calls subroutine EWRITE, RDIRET and BLDSER.
   and SACC-DATA-subroutine DREAD.

4.3 COMMON Variables
   ECLEOT updates LIEBER, HLIBSQ and HFEET of COMMON LOGDAT.
   IMES(26) of COMMON FERMSC is used to write an I/O error message.
5. Procedure

Get to Library Control Block
Read Directory Block
If (Satellite block not found, i.e., RTCODE = 8)
   EXIT to RETURN
Read Satellite Block
Read and Update Library Control block
Increment Tape Volume Number
If (Vol-ser not in range)
   Write error message
   RTCODE = 8
   EXIT to RETURN
Zero file sequence number
Zero amount of feet of tape used
Update COMMON LOGDAT
   HLIBSQ, =0, file sequence
   HFEEET, =0, feet used on tape
   LIBSER, EBCDIC tape vol-ser
   HLBRSER, tape vol-ser

RETURN
6. Special Notes

6.1 Major Variables and/or Constants
None

6.2 Data Structure and Tables
See Table in Appendix for the format of the Library Control block.

6.3 Error Handling
If the next library tape volume-serial exceeds the given range in the library control block, the following message is written and a return code of 8 is passed back to the calling routine.

ECLEOT ****** NEXT VOL-SER EXCEEDS XXXXX.

If an I/O error occurs during the called subroutine DREAD, the following message is written and a return taken.

ECLEOT ****** I/O ERROR ENCOUNTERED.

followed by data from COMMON FERMSSG described in SACC DAIO documentation.
1. Subroutine ECE

2. Function
Subroutine ECE (EDR processing – close EDR) closes the current
EDR block.

3. Programmer and Date
Eunice Eng, September 1977

4. Interface

4.1 Calling Sequence
The FORTRAN calling sequence for ECE is as follows:

CALL ECE(EDRBLK,LIBBLK)

The arguments in the calling sequence are described in the
list below.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDRBLK(32)</td>
<td>I*2</td>
<td>I</td>
<td>EDR block</td>
</tr>
<tr>
<td>LIBBLK(32)</td>
<td>I*2</td>
<td>I</td>
<td>Library block</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

ECE calls subroutine: WRITE.
ECE calls SACC subroutines
KTIME, YDMD, and function-subroutine LGR and -DAIO-subroutine
DREAD.  KMC, KCO, and KCL.

4.3 COMMON Variables

IMES(26) of COMMON FERMSG is used to write an I/O error
message.
Read EDR block from Log

If (EDR block read ≠ EDR block passed in calling sequence)
  Write error message
  Call ABEND(300)

Update EDR block
  Record number of records
  Record number of errors
  Mark Disposition and Completion code
    Locate next to last recorded entry (see EOE)
    Mark processing done
    Record date of production
    If (I/O error, i.e. no. of records = 0)
      Mark completion code = ZOF
      Write EDR block
      Write error message
      EXIT to RETURN
    Clear Completion code
    If (no. of errors ≠ 0)
      Mark completion code = Z01
    Fi
  Mark slot to be removed
  Write success message
  Record Library block @
  Write EDR block to EDR Log
  Read EDR block just written (pass ECL the most recent version)

RETURN

Note: If I/O error encountered during DREAD
  Write Error Message
  EXIT to RETURN
6. Special Notes

6.1 Major Variable and Constant

<table>
<thead>
<tr>
<th>Variable or Constant</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QERDON</td>
<td>L*1</td>
<td>Constant = X'20', flags production done</td>
</tr>
<tr>
<td>HOLDEN(32)</td>
<td>I*2</td>
<td>'Old' EDR block</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Table

See Table in Appendix for the format of an EDR block.

6.3 Error Handling

Should an I/O error occur during a DREAD, an error message is written and a return from ECE is taken. The message is as follows:

**BCE ***** I/O ERROR ENCOUNTERED.**

Followed by data passed in IMES926) of COMMON FERMSC. The data is described in DAIO documentation.

6.4 Other

ECE must be compiled under FORTRANH with parm=XL for function subroutine LOR to be expanded during compilation time.
6.3 Error Handling

Should an I/O error occur during a DREAD, an error message is written and a return from ECE is taken. The message is as follows:

ECE ******* I/O ERROR ENCOUNTERED.

Followed by data passed in IMES(26) of COMMON FERMMSG. The data is described in DAIO documentation.

If the EDR block in the log is different to the block passed in the calling sequence the following message is written and a user abend 300 taken.

******************************************************
THE EDR BLOCK BEING PROCESSED (BLOCK/XXXX) HAS BEEN TEMPERED.
ABENDING FROM SUBROUTINE ECE.
******************************************************

If an I/O error is detected elsewhere NRCEDR (number of records in EDR processing) of COMMON LOGDAT is equated to zero. If ECE detected a zero in NRCEDR, the following message is written:

ECE ******* ERROR PROCESSING EDR (BLOCK NUMBER:XXXX). EDR HAS ZERO RECORDS.

If any errors occurred in EDR processing, the following message is written:

ECE ******* ERROR PROCESSING EDR (BLOCK NUMBER:XXXX). EDR ERROR COUNT NOT ZERO.

If processing is successful the following message is printed:

ECE ******* PROCESSING OF EDR (BLOCK NUMBER:XXXX) HAS BEEN SUCCESSFULLY COMPLETED. PLEASE REMOVE TAPE FROM SLOT XXXXX. UPDATE-PDS AND LOG AFTER REMOVAL.
6.4 Other

ECE must be compiled under FORTRAN with path XL for function subroutine LOR to be expanded during compilation time.
1. Subroutine ECL

2. Function
   Subroutine ECL (EDR processing - close Library) closes the present library block.

3. Programmer and Date
   Bunice Eng, September 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for ECL is as follows:

   CALL ECL(EDRBLK,LIBBLK)

   The arguments in the calling sequence are described in the following list:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDRBLK(32)</td>
<td>I*2</td>
<td>I</td>
<td>EDR block to be updated</td>
</tr>
<tr>
<td>LIBBLK(32)</td>
<td>I*2</td>
<td>I</td>
<td>Library block</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called
   ECL calls Subroutine WRITE and SACC DAEC subroutine DDBEG, KLC.

4.3 COMMON Variables
   IMES(26) of COMMON FERMSG is used to write an I/O error message.
Test last EDR entry for success, i.e. E1 or E6
If(not successful)
   Write error message
   EXIT to RETURN
Test for altered Library block
   If (Library block tampered during processing)
      Write error message
      Call ABEND (300)
Update Library block
   Mark entry for processing
   Increment entry number
   Record @ of EDR block
   FDSC start
   FDSC end
   Library tape serial
   Increment and record file sequence
   @ of 1st Ency. Attr. for satellite
Write Updated library block
Update previous Library block
   If (established, forward link)
Write updated previous Library block
Update Library Control block.
   If (first library block)
      Record @ of present Library block as 1st library block
   Fi
   Record @ of last Library block
   Increment and record file sequence
   Sum total feet of tape used
Write updated Library Control block
Correct EDR's entry number
decrement by one
Write EDR block
RETURN
Note: If and I/O error is encountered during DREAD
Write and error message
EXIT to RETURN
6. Special Notes

6.1 Major Constant

<table>
<thead>
<tr>
<th>Constant</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPRODS</td>
<td>I*2</td>
<td>Constant = X'8000', flags production to be done</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See table in Appendix for the format of the Library block.

6.3 Error Handling

Should an I/O error occur during a DREAD the following message is written:

ECL ****** I/O ERROR ENCOUNTERED.

Followed by data passed in IMES(25) of COMMON PERMMSG. The data is described in the DATO documentation.

6.4 Other

Subroutine ECE should be called before ECL, else no update is performed to the library block for the EDR processing. Also, the most recent version of the EDR block must be passed in as the calling sequence.
6.3 Error Handling

If the Library block in the log is not exactly the same as the library block passed in in the calling sequence, the following message is written and the user abend 300 taken.

*****************************************************************
THE LIBRARY BLOCK USED IN EDR PROCESSING (BLOCK/XXXX) HAS BEEN TEMPERED.
ABENDING FROM SUBROUTINE ECL.
*****************************************************************

If an I/O error should occur during a DREAD the following message is written:

ECL ***** I/O ERROR ENCOUNTERED.
Followed by data passed in IMES(25) of COMMON FERMSG. The data is described in the DAIO documentation.

If ECE was not successful, i.e. EDR block not marked 'done', the following message is printed and a return taken.

ECL ***** LIBRARY BLOCKS NOT UPDATED FOR EDR PROCESSING OF EDR BLOCK/XXXX.
6.4 Other

Subroutine EGE should be called before ECI, else no update is performed to the library block for the EDR processing.
Also, the most recent version of the EDR block must be passed as an argument in the calling sequence.
1. Subroutine LOL

2. Function
   Subroutine LOL (library processing - open library) locates
   the first library block marked for processing, in time order
   by the start FDSC entry.

3. Programmer and Date
   Eunice Eng, October 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for LOL is as follows:

   CALL LOL(ATTRIB,STIME,ATTRBK,LIBBLK,RTCODE).

   The arguments in the calling sequence are described in the
   list below:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIB</td>
<td>I*4</td>
<td>I</td>
<td>@ of Attribute block</td>
</tr>
<tr>
<td>STIME</td>
<td>R*8</td>
<td>I</td>
<td>FDSC sought</td>
</tr>
<tr>
<td>ATTRBK(32)</td>
<td>I*2</td>
<td>Z,C</td>
<td>Ency-Attribute block</td>
</tr>
<tr>
<td>LIBBLK(32)</td>
<td>I*2</td>
<td>0</td>
<td>Library block</td>
</tr>
<tr>
<td>RTCODE</td>
<td>I*4</td>
<td>0</td>
<td>Return Code</td>
</tr>
</tbody>
</table>

   =0, Library block found for processing
   =4, No library block found for processing
   =8, serious error

4.2 Subroutines Called
   LOL calls subroutines BLDSER, SWRITE AND RDIRET.
   LOL also calls SACC function subroutine LOR and DAIO
   subroutine DREAD.

4.3 COMMON Variables
   LOL updates variables ISLOT, INSLOT, HMXFT, ILIBBK, IATTBK,
   HLIBSQ, HLBSER, DSTPDS, DNFDS of COMMON LOGDAT. IMES(26)
   of COMMON PERRMSG is used to write an I/O error message.
5. Procedure

Set return code, RTOCODE = 0, successful
Read Directory block
If (satellite block not found, i.e. RTOCODE = 0)
    EXIT to RETURN
Read Satellite block
Read Library Control block
    Update COMMON LOGDAT
    I1SLOT
    INSLOT
    HXFT
Pass DSN and Vol-Ser to Library block processing

Library Blocks
Initialize to find earliest Library block to process
LATTRB = @ of attribute block passed
NLIBBK = 0
DFDSC = 9999999999999999D70
Do Until (last Library block processed)
    Read a Library block
    DO until (last entry
        If(@ of present lib. block = @ of last lib. block processed)
            EXIT to NEXTLIB
        If (4th process)
            EXIT to NEXTLIB
        If (present start FDSCL/STime)
            EXIT to NEXTLIB
        If (LATTRB = 0)
            LATTRB = default to fixxxxibb attrib. of the first Library block found
        If (present attrb $\neq$ LATTRB)
            EXIT to NEXTLIB
        If (present start FDSCL/DFDSC)
            DFDSC = present FDSCL
            NLIBBK = @ of present Library block
    Fi
NEXTBLK
Get address of next Library block
Od
If (no library block marked for processing, i.e. NLIBBK = 0)
   Write error message
   RTCODE = 4
   EXIT to RETURN

NILBBK = NLIBBK (record @ of Library block in COMMON)

Read Library block to be processed

Mark entry, 'process begun'

If (# of entries = 4)
   Mark next entry 'to do'
   Record @ of Attribute block
   Increment # of entries

Fi

Write Library block

Update COMMON LOGDAT
   HLBSQ
   HLBSEQ
   DSFDS
   DENFDS
   LIBSER
   LIBDSN

Read Attributes block

RETURN
6. Special Notes

6.1 Major Variables and/or Constants

<table>
<thead>
<tr>
<th>Variable or Constant</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPRBEK</td>
<td>I*4</td>
<td>$P$ of present Library block</td>
</tr>
<tr>
<td>QDSNS(18)</td>
<td>L*1</td>
<td>Mask of DSN for Library tape</td>
</tr>
<tr>
<td>QSERMS(6)</td>
<td>L*1</td>
<td>Mask of Library tape vol-ser</td>
</tr>
<tr>
<td>LATTRE</td>
<td>I*4</td>
<td>$P$ of attribute block being processed</td>
</tr>
<tr>
<td>NLIBBK</td>
<td>I*4</td>
<td>$P$ of present Library block that meets time criterion</td>
</tr>
</tbody>
</table>

6.2 Data Structure and/or Tables

See Table Of Appendix for format of a Library block.
6.3 Error Handling

If LOL is unsuccessful in finding a library block marked for processing, a return code of 4 is sent to the calling routine, an error message is written and a return taken.

LOL ***** NO LIBRARY FILE MARKED FOR PROCESSING.

If an I/O error is encountered in a call to subroutine DREAD, a return code of 4 is passed to LOL's calling subroutine and the following error message is written:

LOL ***** I/O ERROR ENCOUNTERED.

followed by I/O data passed to LOL by COMMON FERMSC, described in DAIO documentation.
1. Subroutine LOW

2. Function
   Subroutine LOW (Library processing – open work) allocates a new work block.

3. Programmer and Date
   Bunice Eng, October 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for subroutine LOW is as follows:

   CALL LOW(ATTRBK,WRKBLK,RTCODE)

   The following list describes the arguments in the calling sequence:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRBK(32)</td>
<td>I*2</td>
<td>I</td>
<td>Ency-Attribute block</td>
</tr>
<tr>
<td>WRKBLK(32)</td>
<td>I*2</td>
<td>O</td>
<td>Work block</td>
</tr>
<tr>
<td>RTCODE</td>
<td>I*4</td>
<td>O</td>
<td>Return Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>=0, work block allocated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>=4, work block not allocated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>=8, serious error</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called
   LOW calls subroutines RDIRET, BLDSER, EOPROL AND EMATTE. LOW also calls SACC DATA subroutine BREAD.

4.3 COMMON Variables
   The following variables of COMMON LOGDAT are updated: IATTBK, I1WSLT, INWSLT, IWRKBK, WRKDSN, HWKSER, and WRKSER.
   IMES(36) of COMMON FERMSG is used to write an I/O error message.
5. Procedure

Set Return code, RTCODE = 0, successful

Read a Directory block

If (Satellite block not found, i.e. RTCODE=8)

EXIT to RETURN

Get next block to be allocated

If (next block address > no. of blocks in data set)

Write error message

RTCODE = 4

EXIT to RETURN

Read Satellite block

Read Work Control block

Get address of last work block

Get address of next tape volume

Read tape volume block for work tape vol-ser

If (next vol-ser not in range)

Write error message

RTCODE = 8

EXIT to RETURN

Update COMMON LOGDAT

WKSER
HWKSER
WRKDSN
ILWSLT

Read last work block

Clear and Write new work block

Write prolog

Clear rest of entries
Get and record Ency-Attribute address

If (default): set LADDR = 0

Set default address, LADDR = 0

FI

Record address of previous work block of same Ency-Attributes
Record Library Topic Library elected
Write New work block

Save New work block address

IWRKBK

Update and Write Directory block.

RETURN
6. Special Notes

6.1 Major Variables and/or Constants

<table>
<thead>
<tr>
<th>Variable or Constant</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NXTBLK</td>
<td>I*4</td>
<td>Address of next block to be allocated</td>
</tr>
<tr>
<td>ICONTWK</td>
<td>I*4</td>
<td>Address of Work Control block</td>
</tr>
<tr>
<td>ILASTB</td>
<td>I*4</td>
<td>Address of last work block</td>
</tr>
<tr>
<td>NXTVOL</td>
<td>I*4</td>
<td>Address of block with next tape volume serial for work tapes</td>
</tr>
</tbody>
</table>

6.2 Data Structure

See Table in Appendix for format of work block.
6.3 Error Handling

If LOW is unsuccessful in allocating a new work block, a return code of 4 is sent to the calling subroutine, the message below is written and a return taken.

LOW ****** NEXT AVAILABLE BLOCK NUMBER EXCEEDS XXXXXXXXX.

If the next work volume serial is out of the given range of work tapes' volume serial, the following message is written, the return code is set to 8 and a return taken.

LOW ****** NEXT WORK VOL-SER = XXXXX GREATER THAN XXXXX

If an I/O error occurs in DREAD, a called subroutine, the return code is set to 8 and the following message is written before a return taken.

LOW ****** I/O ERROR ENCOUNTERED.

followed by data from COMMON FERMSG. The data is described in DAIO documentation.
1. Subroutine LCW

2. Function
   Subroutine LCW (library processing – temporarily close work block) temporarily closes a work block

3. Programmer and Date
   E. Eng, October 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for LCW is as follows:
   
   CALL LCW(WRKBK,ATTREK,MAXFLE,RTCODE)
   
   The following list defines the arguments in the calling sequence.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRKBLK(32)</td>
<td>I*2</td>
<td>I</td>
<td>Work block</td>
</tr>
<tr>
<td>ATTREK(32)</td>
<td>I*2</td>
<td>I</td>
<td>Ency-Attribute block</td>
</tr>
<tr>
<td>MAXFLE</td>
<td>I*4</td>
<td>I</td>
<td>Maximum # of files to process</td>
</tr>
<tr>
<td>RTCODE</td>
<td>I*4</td>
<td>O</td>
<td>Return Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0, normal return</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>=4, call LOW before calling another LOL</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LREAD</td>
<td>Subroutine to deblock the EDR Log on a read or a write</td>
</tr>
<tr>
<td>LWRITE</td>
<td></td>
</tr>
</tbody>
</table>

4.3 COMMON Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LATREK</td>
<td>I*4</td>
<td>I</td>
<td>Ency-Attribute</td>
</tr>
<tr>
<td>ILIBBK</td>
<td>I*4</td>
<td>I</td>
<td>Of Library Block</td>
</tr>
<tr>
<td>I1RVOL</td>
<td>I*4</td>
<td>I</td>
<td>Start record volume</td>
</tr>
<tr>
<td>INRVOL</td>
<td>I*4</td>
<td>I</td>
<td>End record Volume</td>
</tr>
<tr>
<td>IRES(26)</td>
<td>I*4</td>
<td>I</td>
<td>I/O error data</td>
</tr>
<tr>
<td>IWKBLK</td>
<td>I*4</td>
<td>I</td>
<td>Of Work block</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMMON LOGDAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMMON LOGDAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMMON LOGDAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMMON LOGDAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMMON FERMUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMMON LOGDAT</td>
</tr>
</tbody>
</table>
5. Procedure

Test for Tempered Work block
Read Work block from Log
If (work block passed work block read)
   Write error message
   Call ABEND(300)
Reset Return code \# RCODE = 0
Update Work Block
   Place library address in work list
   Mark start and end record volume
   If (first library block processed)
      Record start record volume
      Update last work block
      If (last block exists)
         Read last work block
         Estab. forward link
         Write updated last work block
   Fi.
   Update previous work block w/ same Attr.
   If (prev. work block exists)
      Read work block
      Estab. forward link
      Write updated work block
   Fi
Update Ency-Attr. block
   If (first work block)
      record first work block
   Fi
Record last work block
Write Attribute block
Update Work Control block
Read Directory block
   If (successfull not successful)
      EXIT to RETURN
Read Satellite block
Read Work Control block
If (first work block)
   record first work block
Record last work block
Increment vol-ser last used
Write Updated Work Control block

Fi
Update end record volume
record end record volume
Write Updated Work block
Read Updated work block to pass to LCL
Test for full work block
MXFILE = 15
If (input MAXFILE < MXFILE < 15)
    MXFILE = MAXFILE
If (# of Library blocks = MXFILE)
    RTCODE = 4
Fi

RETURN
6. Special Notes

6.1 Major Variables and/or Constants

<table>
<thead>
<tr>
<th>Variable or Constant</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTODO</td>
<td>I*2</td>
<td>= X'8000', flags a request for processing</td>
</tr>
<tr>
<td>IBLKNC</td>
<td>I*4</td>
<td>Library block address</td>
</tr>
<tr>
<td>HPRWRK</td>
<td>I*2</td>
<td>Present work block address</td>
</tr>
<tr>
<td>LSTWRK</td>
<td>I*4</td>
<td>Address of previous work block of same Ency-Attributes</td>
</tr>
<tr>
<td>ILASTW</td>
<td>I*4</td>
<td>Offset last work block pointed by present work block</td>
</tr>
</tbody>
</table>

6.2 Data Structure and/or Tables
See Table in Appendix for format of a work block.
6.3 Error Handling

If an I/O error is detected in subroutine DREAD the following message is written and a return taken.

LCW ****** I/O ERROR ENCOUNTERED.
followed by data from COMMON FERMSG. The data is described in DAI0 documentation.

If the work block has been tempered during the library processing the following message is printed and a user abend (300) taken.

**************************************************************************
THE WORK BLOCK USED IN LIBRARY PROCESSING (BLOCK/XXXX) HAS BEEN TEMPERED.
ABENDING FROM SUBROUTINE LCW.
**************************************************************************

If the work block has $\frac{1}{2}$ library addresses in its list, a return code of 4 is passed through the calling sequence to request that subroutine LLOW be called before another call to subroutine LOL is issued.
1. Subroutine LCL

2. Function
   Subroutine LCL (library processing - close library) closes a library block.

3. Programmer and Date
   Ennise Eng, October 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for LCL is as follows:
   
   CALL LCL(LIBBLK,WRKBLK,CKADL)
   
   The following 2x2 list describes the arguments in the calling sequence.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIBBLK(32)</td>
<td>I*2</td>
<td>I</td>
<td>Library block</td>
</tr>
<tr>
<td>WRKBLK(32)</td>
<td>I*2</td>
<td>I</td>
<td>Work block</td>
</tr>
<tr>
<td>CKADL</td>
<td>I*4</td>
<td>I</td>
<td>last byte contains error message</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called
   LCL calls subroutine WRITE and SACC subroutines KTIME and YDMD and SACC function subroutine LOR.

4.3 COMMON Variables
   COMMON LOGDAT is referenced, but none of the variables are altered.
5. Procedure

Check for updated work block

If (© of Library block not in work list)

Write error message
EXIT to RETURN

Check for tempered Library block

Read Library block form Log

If (library block passed ≠ Library block in Log)

Write error message
Call ABEND(300)

Temporarily close Library block

Mark entry temporarily closed – Z30
Record error in disposition
Record date of process
Record © of Ency-Attribute
Record © of Work block

Write Updated Library block

RETURN
6. Special Notes

6.1 Major Variables and/or Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPROD</td>
<td>L×1</td>
<td>= X'20', flags processing (temp)</td>
</tr>
<tr>
<td>QTECC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.2 Data Structure and/or Tables

See Table in Appendix for format of a library and work block.

6.3 Error Handling

None.

If the current library block has not successfully been added to the work list in the work block (i.e. subroutine LCL called before subroutine LCM), the following message is printed and a return taken.

```
LCL ***** WORK BLOCK DOES NOT HAVE PRESENT LIBRARY ADDRESS IN ITS LIST.
LIBRARY BLOCK XXXXX WILL NOT BE UPDATED.
```

If the library block has been tampered during library processing, the following message is written followed by a user abend (300).

```
******************************************************************************
THE LIBRARY BLOCK USED IN LIBRARY PROCESSING (BLOCK XXXXX) HAS BEEN TAMPERED.
ABENDING FROM SUBROUTINE LCL.
******************************************************************************
```
If an I/O error occurs the following message is written followed by data from COMMON FERMG.

LCL ***** I/O ERROR Encountered.
1. Subroutine LTERM

2. Function

Subroutine LTERM (Library processing - terminate processing) closes work and library blocks.

3. Programmer and Date

Nand Lal, November 1977

4. Interface

4.1 Calling sequence

The FORTRAN calling sequence for LTERM is as follows:

CALL LTERM(WRKBLOK)

The following describes the argument in the calling sequence.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRKBLOK(32)</td>
<td>I*2</td>
<td>I</td>
<td>Work block</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LREAD</td>
<td>Reads an EDR Log block and unblocks a record. LWRITE reads before writing</td>
</tr>
<tr>
<td>(LWRITE)</td>
<td>SACC = compares strings byte by byte</td>
</tr>
<tr>
<td>KCLC</td>
<td>SACC = abend the program</td>
</tr>
<tr>
<td>ABEND</td>
<td></td>
</tr>
</tbody>
</table>

4.3 COMMON Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IWRKEX</td>
<td>I*4</td>
<td>I</td>
<td>@ of work block - COMMON LOGDAT</td>
</tr>
<tr>
<td>INRVOL</td>
<td>I*4</td>
<td>I</td>
<td>End record volume - COMMON LOGDAT</td>
</tr>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td></td>
<td>I/O error data - COMMON FERMISG</td>
</tr>
</tbody>
</table>
5. Procedure

Temper Test

Read work block in Log

If (work block passed / work block in Log)

Write error message

Call ABEND (300)

Update work block

Close Library block

Do Until (List in Work block Exhausted)

Read Library block

Decrement # entries

Mark Library entry done

Write Library block

Od

Record end volume number

Record processing disposition - 'to do'

Merge processing

CITENCY processing

RETURN
6. Special Notes

6.1 Major Variables and/or Constants

<table>
<thead>
<tr>
<th>Variable or Constant</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTODO</td>
<td>I#2</td>
<td>Z8000, to process flag</td>
</tr>
</tbody>
</table>

6.2 Data Structure and/or Tables

See Table in Appendix for format of a work and library block of an EDR block.
6.3 Error Handling

If the work block has been tempered with before calling LTERM the following message is written.

******************************************************************************
THE WORK BLOCK USED IN LIBRARY PROCESSING (BLOCK XXXX) HAS BEEN TEMPERED.
ABENDING FROM SUBROUTINE LTERM.
******************************************************************************

If an I/O error occurs the following message is written followed by data from COMMON FERMSC.

LTERM ***** I/O ERROR ENCOUNTERED
1. Subroutine WCOW

2. Function

Subroutine WCOW (work processing CITENCY - open work) finds
and opens a work block marked for CITENCY processing.

3. Programmer and Date

Emice Eng, November 1977

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for WCOW is as follows:

CALL WCOW (WRKBLK, ILOGUN, RTCODE)

The arguments in the calling sequence are described in the
list below.

<table>
<thead>
<tr>
<th>ARGUMENT</th>
<th>TYPE</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRKBLK(32)</td>
<td>I*2</td>
<td>0</td>
<td>WORK BLOCK</td>
</tr>
<tr>
<td>ILOGUN</td>
<td>I*4</td>
<td>1</td>
<td>DEVICE NUMBER OF LOGICAL UNIT OF LOG</td>
</tr>
<tr>
<td>RTCODE</td>
<td>I*4</td>
<td>0</td>
<td>RETURN CODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 = NORMAL RETURN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 = WC WORK BLOCK FOUND MARKED FOR CITENCY PROCESSING</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8 = SEVERE ERROR</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

WCOW calls subroutines RDIRST, BLDSER, and WRFR. WCOW
also calls SACC subroutines DRDAD, KNC and KOB.

4.3 COMMON Variables

The list below describes the variables used from COMMON LOGDAT.

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>TYPE</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRKSER</td>
<td>I*4</td>
<td>0</td>
<td>BINARY WORK TAPE SERIAL</td>
</tr>
<tr>
<td>INRVL</td>
<td>I*4</td>
<td>0</td>
<td>FIRST RECORD VOLUME ON WORK TAPE</td>
</tr>
<tr>
<td>INRSLT</td>
<td>I*4</td>
<td>0</td>
<td>LAST RECORD VOLUME ON WORK TAPE</td>
</tr>
<tr>
<td>INRSLT</td>
<td>I*4</td>
<td>0</td>
<td>FIRST WORK SLOT</td>
</tr>
<tr>
<td>INRSLT</td>
<td>I*4</td>
<td>0</td>
<td>LAST WORK SLOT</td>
</tr>
<tr>
<td>INRSLT</td>
<td>I*4</td>
<td>0</td>
<td>SERIAL OF WORK BLOCK</td>
</tr>
<tr>
<td>WRKSER</td>
<td>I*4</td>
<td>0</td>
<td>WORK TAPE VOL-SER. BUCDIC</td>
</tr>
<tr>
<td>WRKSER</td>
<td>L*4</td>
<td>0</td>
<td>SATELLITE 1O. BUCDIC</td>
</tr>
<tr>
<td>USATID(12)</td>
<td>L*4</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

IMES(25) of COMMON FERMSSG is used to write an I/O error message.
Reset Return Code

RTOCODE = 0

Read Directory block

If (satellite block not found)

Write-error-message

EXIT to RETURN

Read Satellite Block

Read Work Control Block

Update Common LOGDAT

IWRKEK

WRKDSN

IWSLT

IWSLT

Get mask of work vol-ser

WORK Find Work Block marked for CIFYENCY processing

Read Work block

If (work block marked for CIFYENCY processing)

Mark processing begun

Update COMMON LOGDAT

ILRVOL

ILRVOL

WRKSER

Write Work block

EXIT to RETURN

Get next work block

If (address of next work block ≠ 0)

EXIT to WORK
WCON

Write error message

RTOCODE = 4

RETURN Return

Note: If I/O error during DALO)

Write error message

RTOCODE = 3

EXIT to RETURN.
6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LG000</td>
<td>L#1</td>
<td>=230, FLATE 'TO ON'</td>
</tr>
<tr>
<td>CBEGUN</td>
<td>L#1</td>
<td>=740, FLAGS 'PROCCESS BEGUN'</td>
</tr>
<tr>
<td>IDATAU</td>
<td>I#4</td>
<td>8 OF SATELLITE BLOCK</td>
</tr>
<tr>
<td>IVCLSN</td>
<td>I#4</td>
<td>SERIAL OF WORK TAPE</td>
</tr>
<tr>
<td>IMARK</td>
<td>I#4</td>
<td>8 OF WORK CONTROL BLOCK</td>
</tr>
<tr>
<td>CURASKR(6)</td>
<td>L#1</td>
<td>WORK SERIAL MASK</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Table of for the format of a Work block.
6.3 Error Handling

If no work blocks are found marked for CITEENCY processing the following message is printed, a return code of 4 is set and a return taken.

_WCON ***** NO WORK BLOCK FOUND MARKED FOR CITEENCY PROCESSING._

If an I/O error is detected, the following message is printed followed by I/O data from COMMON FERMISG, a return code of 8 is set and a return taken.

_WCON ***** I/O ERROR ENCOUNTERED._
1. Subroutine WCOC

2. Function

Subroutine WCOC (work processing CITENCY-open) opens the CITENCY block allocated and opens a CITENCY block.

3. Programmer and Date

Eunice Eng, October 1977

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for WCOC is as follows:

CALL WCOC (CITBLK, ILOGUN, RETCODE)

The arguments in the calling sequence are described below.

<table>
<thead>
<tr>
<th>ARGUMENTS</th>
<th>TYPE</th>
<th>IAG</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITBLK</td>
<td>I*2</td>
<td>C</td>
<td>CITENCY BLOCK</td>
</tr>
<tr>
<td>ILOGUN</td>
<td>I*4</td>
<td>I</td>
<td>LOGICAL UNIT FOR LOG</td>
</tr>
<tr>
<td>RETCODE</td>
<td>I*4</td>
<td>D</td>
<td>RETURN CODE</td>
</tr>
</tbody>
</table>

=0, CITENCY BLOCK ALLOCATED
=4, CITENCY BLOCK NOT ALLOC.
=8, SERIOUS ERROR

4.2 Subroutines Called

WCOC calls subroutines BLDSER, EOPROL, RDIRECT and SUBRITE. WCOC is in turn called by DREAD.
### 4.3 Common Variables

The list below describes the variables used from COMMON I/O DAT.

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>TYPE</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINRASK</td>
<td>I*2</td>
<td>I</td>
<td>BINARY NOT RAW RATES' SERIAL</td>
</tr>
<tr>
<td>FINRASK1</td>
<td>I*2</td>
<td>D</td>
<td>BINARY RAW RATES' SERIAL</td>
</tr>
<tr>
<td>ICITYB</td>
<td>I*4</td>
<td>I</td>
<td>E OF CITENCY BLOCK ALLOCATED</td>
</tr>
<tr>
<td>LARKEK</td>
<td>I*4</td>
<td>I</td>
<td>E OF Work Block</td>
</tr>
<tr>
<td>CCKUSH(1)</td>
<td>L#1</td>
<td>C</td>
<td>MASK OF CITENCY DSN, EBCDIC</td>
</tr>
<tr>
<td>CKNRSR(6)</td>
<td>L#1</td>
<td>C</td>
<td>NOT RAW RATES' SERIAL, EBCDIC</td>
</tr>
<tr>
<td>CKNRSR(6)</td>
<td>L#1</td>
<td>C</td>
<td>RAW RATES' SERIAL, EBCDIC</td>
</tr>
<tr>
<td>OSATIC(1)</td>
<td>L#1</td>
<td>I</td>
<td>SATELLITE ID, EBCDIC</td>
</tr>
</tbody>
</table>

**INES(26)** of COMMON FERMSG is used to write an I/O error message.
Reset Return Code

RTCODE = 0

Read Directory Block

If (satellite block not found)
   EXIT to RETURN

Get next block to be allocated

If (next block exceeds blocks in data set)
   Write error message
   RTCODE = 4
   EXIT to RETURN

Read Satellite Block

Read CITENCY Control block

If (all CITENCY slots filled)
   Write error message
   RTCODE = 8
   EXIT to RETURN

Get next serial

If (next serial out of range)
   Write error message
   RTCODE = 4
   EXIT to RETURN

Update LOGDAT

HRAWSR
HNRSR
QRAWSR
QNRWSR
QCMDSN

Get address of last CITENCY block
Fill new CITENCY block

Prolog

Raw rates' serial

Not raw rates' serial

Mark slot allocation

Address of work block

Write New CITENCY block

Update DIRECTORY block

Last block allocated

Write Updated Directory

RETURN Return

Note: If (I/O error during DAIO)

Write error message

RTCODE = 3

EXIT to RETURN
6. Special Notes

6.1 Major Constants and Variables

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSCNTY</td>
<td>1#4</td>
<td>&amp; OF LAST CITENCY BLOCK</td>
</tr>
<tr>
<td>GCALDO</td>
<td>1#1</td>
<td>&amp; OF CITENCY CONTROL BLOCK</td>
</tr>
<tr>
<td>IGICHON</td>
<td>1#4</td>
<td>&amp; OF SATCHELITE BLOCK</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Table of the CITENCY block for the format.
6.3 Error Handling

If the next block to be allocated exceeds the data set the following message is written, a return code of 4 is set and a return taken.

**WCOG **NEXT BLOCK TO BE ALLOCATED EXCEEDS XXXXX.

If all the CITENCY slots are already used, a return code of 8 is set, the following message is printed and a return taken.

**WCOG **CITENCY SLOTS ALL USED.

PLEASE CLEAR SLOTS FOR NEXT SET OF CITENCY SERIALS.

If WCOG discovers that there are no more CITENCY slots to be used, the same message above is printed, a return code of 4 is set and a return taken.

If an I/O error is encountered, the following message is written, a return code of 8 is set and a return taken.

**WCOG **I/O ERROR ENCOUNTERED.

The message is followed by I/O data passed by COMMON PI0MSG.
1. Subroutine WCCG

2. Function

Subroutine WCCG (work processing CITERNCY - close CITERNCY) closes a CITERNCY block.

3. Programmer and Date

Runice Eng, November 1977

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for WCCG is as follows:

```
CALL WCCG(CITELK,ILOGUN)
```

The arguments in the calling sequence are described in the list below.

<table>
<thead>
<tr>
<th>ARGUMENTS</th>
<th>LVAR</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITELK(32)</td>
<td>I#2</td>
<td>I</td>
<td>CITERNCY BLOCK</td>
</tr>
<tr>
<td>ILOGUN</td>
<td>I#4</td>
<td>I</td>
<td>LOGICAL UNIT FOR LOG</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

WCCG calls subroutines WRITE and RDIRET. WCCG also calls SACC subroutine KCUC and DATA subroutine DRDAD.

4.3 COMMON Variables

The list below describes the variables used from COMMON LOGDAT.

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>LVAR</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FWRMSK</td>
<td>I#2</td>
<td>I</td>
<td>CITERNCY, NET, RAW, SERIAL</td>
</tr>
<tr>
<td>ICLOCX</td>
<td>I#4</td>
<td>I</td>
<td>8 OF CITERNCY BLOCK</td>
</tr>
<tr>
<td>ITMAXK</td>
<td>I#4</td>
<td>I</td>
<td>SLUT FOR NET, RAW RATES</td>
</tr>
<tr>
<td>ITMAXL</td>
<td>I#4</td>
<td>I</td>
<td>SLUT FOR RAW RATES</td>
</tr>
<tr>
<td>CSAT10(12)</td>
<td>L#1</td>
<td>I</td>
<td>SATELLITE 10, ESDUIC</td>
</tr>
</tbody>
</table>

IMES(26) of COMMON PERSNO is used to write an I/O error message.
READ CITENCY block from file and
IF (CITENCY block tempered during processing)

Write error message

ABEND(300)

Update CITENCY block

Record slot of raw rates' tape

Record slot of not raw rates' tape

Mark slot allocation to be removed

Write Updated CITENCY Block

Update Previous CITENCY block'

IF (@ of previous CITENCY block ≠ ∅)

Link forward to newest CITENCY block

Write Updated previous CITENCY block

FI

Update CITENCY Control block

Read Directory

IF (Satellite block not found)

EXIT to NEKKKH ABEND (404)

Read Satellite block

Read CITENCY Control block

Last tape serial used

IF (@ of last CITENCY = ∅)

@ of first CITENCY block

FI

@ of last CITENCY block

Write Updated CITENCY Control block
WCCC

Read latest CSECT.CY block
EXIT to RETURN

ABEND  Abend (430)

RETURN  Return

Note: If I/O error during DA10

Write error message
EXIT to ABEND
6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAST</td>
<td>*4</td>
<td># OF LAST CITENCY BLOCK</td>
</tr>
<tr>
<td>ICITCN</td>
<td>*4</td>
<td># OF CITENCY CONTROL BLOCK</td>
</tr>
<tr>
<td>IGATAD</td>
<td>*4</td>
<td># OF SATELLITE BLOCK</td>
</tr>
<tr>
<td>GRENQV</td>
<td>*1</td>
<td>=220, REQUESTS REMOVAL OF TAPE</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Table of a mark.CITENCY block for the format of
6.3 Error Handling

If the CITENCY block was tempered with during the work processing the following message is written followed by a user ABEND (300).

*****************************************************************************

THE CITENCY BLOCK USED IN WORK PROCESSING (BLOCK XXXXX) HAS BEEN TEMPERED.

ABENDING FROM SUBROUTINE WCCC.

*****************************************************************************

If the satellite block is not found for updating the CITENCY Control block, a user ABEND (400) is taken.

If and I/O error is detected the following message is written followed by I/O data from COMMON FERMIG, and a user ABEND (400) is taken.

WCCC ***** I/O ERROR ENCOUNTERED.
1. Subroutine WCCW

2. Function
Subroutine WCCW (work processing CISENCY - close work) closes a work block.

3. Programmer and Date
Enice Eng, November 1977

4. Interface
4.1 Calling Sequence
The FORTRAN calling sequence for WCCW is as follows:

    CALL WCCW(WRKBLK,CITBLK,ILOGUN)

The arguments in the calling sequence are described in the list below.

<table>
<thead>
<tr>
<th>ARGUMENT</th>
<th>TYPE</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRKBLK</td>
<td>1*2</td>
<td>I</td>
<td>WORK BLOCK</td>
</tr>
<tr>
<td>CITBLK</td>
<td>1*2</td>
<td>I</td>
<td>CISENCY BLOCK</td>
</tr>
<tr>
<td>ILLOGUN</td>
<td>1*4</td>
<td>I</td>
<td>LOGICAL ONLY NUMBER OF LOG</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called
WCCW calls subroutine READ. WCCW also calls SACC subroutines KCLC and ABEND and DATA-subroutine-READ.

4.3 COMMON Variables
The list below describes the variables used from COMMON LOGDAT.

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>TYPE</th>
<th>I/O</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICITBLK</td>
<td>1*4</td>
<td>I</td>
<td>1 OF CISENCY BLOCK</td>
</tr>
<tr>
<td>ILWORK</td>
<td>1*4</td>
<td>I</td>
<td>1 OF WORK BLOCK</td>
</tr>
</tbody>
</table>

IMES(26) of COMMON FERMSG is used to write an I/O error message.
Temper Test

Work block from EMU

If (Work block tempered during processing)

Write error message

ABEND (300)

Check CITENCY closure

If (CITENCY slot allocation not marked for $D$ removal)

Write error message

EXIT to RETURN

Update Work block

Mark processing done

Record CITENCY block address

Write Updated Work block

RETURN

Return

Note: If an error encountered during a SCAN
Write error message,
EXIT to RETURN.
6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNF</td>
<td>L#1</td>
<td>=22C, FLAG 'PROCESSING DONE'</td>
</tr>
<tr>
<td>DRENCF</td>
<td>L#1</td>
<td>=22C, FLAG 'REMOVE TAPE'</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Table of for the format of a work block and a CITENCY block.
6.3 Error Handling
If the work block has been tempered during the work processing, the following message is written and a user ABEND (300) is taken.

******************************************************************************
THE WORK BLOCK USED IN WORK PROCESSING (BLOCK XXXXX) HAS BEEN TEMPERED.
******************************************************************************

ABEND FROM SUBROUTINE WCCW.
******************************************************************************

If the CITENCY block has not been closed, WCCW writes the following message and returns.

WCCW ****** WORK BLOCK NOT UPDATED FORM WORK PROCESSING OF CITENCY BLOCK.

If an I/O error is detected the following message is written and a return taken.

WCCW ****** I/O ERROR ENCOUNTERED.
The message is followed by I/O data passed in COMMON FERMSG.
1. Subroutine LREAD (with ENTRY LWRITE)

2. Function
   Subroutine LREAD unblocks voyager log data blocks.

3. Programmer and Date
   Nand Lal, November 1977

4. Interface

4.1 Calling Sequence

   The FORTRAN calling sequences for LREAD and LWRITE are as follows:

   CALL LREAD(UNIT, RECORD, AREA, *)
   CALL LWRITE(UNIT, RECORD, AREA, *)

   The list below describes the arguments in the calling sequences.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT</td>
<td>I*4</td>
<td>I</td>
<td>Logical unit number of EDR log</td>
</tr>
<tr>
<td>RECORD</td>
<td>I*4</td>
<td>I</td>
<td>Record number</td>
</tr>
<tr>
<td>AREA(16)</td>
<td>I*4</td>
<td>O</td>
<td>One Log record</td>
</tr>
<tr>
<td>*</td>
<td>I*4</td>
<td></td>
<td>Statement number to branch to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in case of I/O error</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DREAD</td>
<td>FTIO - reads a logical record from dBk LOG</td>
</tr>
<tr>
<td>DWRITE</td>
<td>FTIO - writes a logical record to LOG</td>
</tr>
<tr>
<td>KMVC</td>
<td>SACC - moves bytes from one area into another</td>
</tr>
</tbody>
</table>

4.3 COMMON Variables

   NONE
LREAD

Calculate block # of the record
Calculate displacement into block of record
read the block
If (I/O error)
    EXIT to RETURN1
Move 64 bytes from read buffer into user area
RETURN

WRITE

Entry
Calculate block # of the record
Calculate displacement into block of record
Read the block
If (I/O error)
    EXIT to RETURN1
Move 64 bytes from user area into read buffer
Write Read buffer block back to LOG
RETURN
6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFER(1808)</td>
<td>I*4</td>
<td>buffer of one log block</td>
</tr>
<tr>
<td>BLOCK</td>
<td>I*4</td>
<td>block number</td>
</tr>
<tr>
<td>DISP</td>
<td>I*4</td>
<td>displacement into block of record</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Appendix for structure of the EDR Log.

- # record/block  = 113
- bytes/block     = 1808
- bytes/record    = 64

6.3 Error Handling

If an I/O error is encountered during DREAD, an alternate return is taken to the calling routine. The calling routine should handle the error.
1. Routine: Subroutine LEDAD


3. English Name: Log Read

4. Language: FORTRAN or FORTRAN II, level 21.6, 367/91/75 06/MVT

5. Purpose: Read unlakes MJS Log data blocks.

6. Calling sequence:
   CALL LEDAD(UNIT, RECORD, AREA)
   CALL LWRIT (UNIT, RECORD, AREA)

   Argument | Type | I/O   | Description
   ----------|------|------|----------------
   UNIT      | I#   | I    | logical unit number of MJS Log
   RECORD    | I#   | I    | record number
   AREA(16)  | I#   | O    | one Log record
   *         | I#   | I    | statement number to branch to in case of I/O error

7. Notes:
   7a. Restrictions: None
   7b. Special Features: None

8. Variables:
   8a. Local
       VARIABLE BUFFER [1804]
       BLOCk
       MISE

   8b. COMMON
       COMMON

9. I/O Information:
   Unit No.  Description
   UNIT     MJS Log

10. Error Handling:
    I/O error
        an alternate return taken

11. Subroutines Called:
    Subroutine Description
    DREAD Read a logical record from Log
    DWRITE Write a logical record to Log
    RMOV Move bytes from one area into another

12. Called By:
    Routine Description
    MJS Log Utility

13. Method:
    Calc. block # of record
    Calc. displacement into block of record
    Read the block
    IF (I/O error)
    EXIT to return
    Move 64 bytes from read buffer into user area

    LWRITE Calc. block # of record
    Calc. displacement into block of record
    Read the block
    IF (I/O error)
    EXIT to return
    Move 64 bytes from user area to real buffer
    Write read buffer block to Log
    RETURN

14. Reference:
    Format for MJS Log
MJS Log Structure:

- # records/block = 113
- bytes/block = 1338
- bytes/record = 54

C5. Programmer and Date:


C16. Modifications: None

---

*** END OF MEMBER ***

- 93 RECORDS PROCESSED

---

---
1. Subroutine EOPROL

2. Function
   Subroutine EOPROL writes part of the EDR log prolog for a block.

3. Programmer and Date
   Bunice Eng, September 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for EOPROL is as follows:
   
   CALL EOPROL(IBLKCD,IPREAD,HBLOCK,HBLOCK)
   
   The calling arguments are described in the list below:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBLKCD</td>
<td>I*4</td>
<td>I</td>
<td>Block type code/See Table of Appendix for codes.</td>
</tr>
<tr>
<td>IPREAD</td>
<td>I*4</td>
<td>I</td>
<td>Block number of the previous block of similar code.</td>
</tr>
<tr>
<td>HBLOCK(32)</td>
<td>I*2</td>
<td>I/O</td>
<td>Block to have prolog written</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called
   EOPROL calls SACC Subroutines KTIME and YMDM.

4.3 COMMON Variables
   None
5. Procedure

Get Block identifier code
Get Creation Date
Get Address of previous block of this type
Zero Address of next block of this type

RETURN
6. Special Notes

6.1 Major Variables and/or Constants
None

6.2 Data Structure and Tables
See Tables in Appendix for the format of the
EDR log prolog

6.3 Error Handling
None
1. Routine: Subroutine EOPROL


3. English Name: Open Prolog

4. Language: FORTRAN 77, Level 21.6, OS/MVS

5. Purpose: EOPROL writes part of the MJS log prolog for a block.

6. Calling Sequence:
   CALL EOPROL(IBLKCD,IBREAD,HBLOCK,HBLOCK)

   Argument | Type | Description
   IBLKCD   | I+4  | block type code. See MJS Log for codes.
   IBREAD   | I+4  | block number of the previous block of similar code.
   HBLOCK(32) | I+2 | block to have prolog written.

7. Notes:
   7a. Restrictions: None

   7b. Special Features: None

8. Variables:
   8a. Local
   Variable | Type | Description
   None

   8b. COMMON
   COMMON | Variables
   None

9. I/O Information:
   Unit No. | Use Description
   None

10. Error Handling: None

11. Subroutines Called:
   Subroutine | Description
   KTMN | SACC - gets current system time.
   YDMD | SACC - converts current system time to year, month and day.

12. Called By:
   Routine | Description
   MJS Log Utility

13. Method:
   Get block identifier code
   Get creation date
   Get address of previous block of this type
   Get zero address of next block of this type
   Return

14. Reference:
   Format for MJS Log

15. Programmer and Date:
    Ronnie Eng., Computer Sciences Corp., Sept. 1977

16. Modifications: None

*** END OF MEMBER ***  69 RECORDS PROCESSED  ***********************
1. Subroutine RDIRET

2. Function
   Subroutine RDIRET reads the directory block of the EDR Log and returns the satellite block address and the directory block.

3. Programmer and Date
   Eunice Eng, September 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for RDIRET is as follows:
   
   CALL RDIRET(NUNIT,HDIREK,HDIRREK,ISATAD,RTCODE)
   
   The list below describes the arguments in the calling sequence.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUNIT</td>
<td>I*4</td>
<td>I</td>
<td>Unit number of EDR Log</td>
</tr>
<tr>
<td>HDIREK(32)</td>
<td>I*2</td>
<td>O</td>
<td>Directory block</td>
</tr>
<tr>
<td>ISATAD</td>
<td>I*4</td>
<td>O</td>
<td>Address of Satellite block</td>
</tr>
<tr>
<td>RTCODE</td>
<td>I*4</td>
<td>O</td>
<td>Return Code = 0, normal return = 8, problems encountered</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called
   RDIRET calls SACE-DAIO subroutine DREAD.

4.3 COMMON Variables
   RDIRET uses IMES(26) of COMMON FERMSG to write an I/O error message from called subroutine DREAD. RDIRET updates ISATCD Of COMMON LOGDAT.
5. Procedure

Set Return Code, RTCODE = 8, failed
Read a Directory block
  If (Directory block not read)
    Write an error message
    EXIT to RETURN
  Check Satellite id and get Satellite address
    If (Satellite id not found)
      Write an error message
      EXIT to RETURN
    Get Satellite id code to Update COMMON LOGDAT

Get Satellite block address
Reset Return code, RTCODE = 0

RETURN
6. Special Notes

6.1 Major Variables and/or Constants
None

6.2 Data Structure and Tables
See Table of Appendix for format of directory block.

6.3 Error Handling
If RDIRET detects an error a return to the calling routine with a return code of 8 is taken after one of the following messages is printed.

RDIRET ****** DIRECTORY BLOCK NOT FOUND.
RDIRET ****** SATELLITE ID = AAAAAAAAAAAAA NOT FOUND.
RDIRET ****** I/O ERROR ENCOUNTERED.

The I/O error message is followed by data from COMMON FERMSC. For description of the data see SACC DAIO documentation.
1. Routine: Subroutine ROTMRT
2. System, Satellite, Version:
   MJS Automatic Log Interface, Voyager-1 and -2, Ver. 1
3. English Name: Real Directory
4. Language:
   FORTRAN or FORTRAN level 21.6 360/91/75 OS/MVT
5. Purpose:
   ROTMRT reads the directory block and returns the satellite block address and the directory block.
6. Calling Sequence:
   CALL ROTMRT(MUNIT, HDIRB, IDIRB, ISATAD, RTCODE)
   Argument   Type   I/O   Description
   MUNIT     integer   I   unit number of MJS Log
   HDIRB     integer   I   directory block
   IDIRB     integer   I   address of satellite block
   ISATAD     integer   I   return code
   RTCODE     integer   O   8 = problems encountered

7. Notes:
7a. Restrictions: None
7b. Special Features: None

8. Variables:
8a. Local Variables   Type   Description
     None
8b. COMMON
     COMMON      Variables
     TERRS (26)   - I/O
     LOGLAT   - I/O

9. I/O Information:
   Unit No.   Use   Description
   MUNIT     MJS Log

10. Error Handling:
    Directory block write error message, set return
    not found code to 8 and return
    Satellite id write error message, set return
    not found code to 8 and return
    I/O error write error message, set return

11. Subroutines Called:
    Subroutine Description
    ERRMA read MJS Log doing its own deblocking

12. Called By:
    Routine Description
    MJS Log Utility

13. Method:
    SET RETURN CODE = 8
    READ directory block
    IF (I/O error)
      WRITE error message
      EXIT to RETURN
    IF (directory block not read)
      WRITE error message
      EXIT to RETRIEVE
    IF (satellite id not found)
      WRITE error message
      EXIT to RETURN
    BLKAD
    Update satellite id in COMMON LOGLAT
    Set satellite block address
    SET RETURN CODE = 8
    Return
C14. Reference:
Format for MJS Log
COMMON LOGDAT
PTIO COMMON PHANS

C15. Programmer and Date:
Furino Eng, Computer Sciences Corp., Sept. 1977

C16. Modifications: None

*** END OF MEMBER ***  89 RECORDS PROCESSED  ******************************************
1. Subroutine BLDSER

2. Function
   Subroutine BLDSER builds a 6 digit, EBCDIC, tape, volume-
   serial number given the 4-byte integer.

3. Programmer and Date
   Bunice Eng, September 1977

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for BLDSER is as follows:
   
   CALL BLDSER(IVOSER,QVOSER,QMSKAD)
   
   The following list defines the arguments in the calling
   sequence.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVOSER</td>
<td>I*4</td>
<td>I</td>
<td>Binary Volume-Serial of tape</td>
</tr>
<tr>
<td>QVOSER(6)</td>
<td>L*1</td>
<td>0</td>
<td>EBCDIC vol-ser</td>
</tr>
<tr>
<td>QMSKAD(6)</td>
<td>L*1</td>
<td>I</td>
<td>Mask for vol-ser</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called
   BLDSER XXX calls SACC subroutine INCORE and function
   subroutine LOR.

4.3 COMMON Variables
   None
5. Procedure

Convert binary vol-ser to EBCDIC vol-ser
Get vol-ser using the vol-ser mask
RETURN

6. Special Notes

6.1 Major Variables and/or Constants
None

6.2 Data Structure
The volume-serial is a six character field, three characters
to define the satellite and the process and three decimal
digits.

6.3 Error Handling
None

6.4 Other
Compile under FORTRANH. Do not use parm=XL, as subroutine
LOR is used from the Fortran Library and not expanded during
compilation time.
Routine: Subroutine BLOSER


English Name: Build Serial

Language: FORTRAN level 21.6 360/71/75 05/M7T

Purpose: BLOSER builds a 6 digit, EBCDIC, tape volume-serial number given a 4-byte integer.

Calling Sequence: CALL BLOSER(IVMSTG, QMSTG, QSMKAD)

Arguments: Type I/O Description
IVMSTG I*4 I binary volume-serial of tape
QSMSTG(6) L*4 0 EBCDIC vol-ser
QSMKAD(6) L*1 I mask for vol-ser

7a. Restrictions: Compile under FORTRAN w/o para=XL for for FORTRAN Subroutine I03.

7b. Special Features: None

Variables:

Pa. Local
Variable Type Description
None

8d. COMMON - COMMON Variables
None

I/O Information:
Unit No. Use Description
None

Error Handling: None

Subroutines Called:
Subroutine Description
INCO2? SACC - converts binary to EBCDIC
L0P SACC - logical ox

Called By:
Routine Description
MMS Log Utility

Method:
Convert binary vol-ser to EBCDIC vol-ser
Get vol-ser using the vol-ser mask

Reference:
A Tape vol-ser has 6 characters. The first 3 define the satellite and process. The last 3 are used as a decimal serial.

Programmer and Date:
Brian Eng, Computer Sciences Corp., Sept. 1977

Modifications: None

*** END OF MEMBER *** 67 RECORDS PROCESSED
1. Subroutine DIRSAT

2. Function.
   Read the MJS Log directory block and locate the desired satellite block.

3. Programmer and Date.
   Eunice Eng, January 1978

4. Interface.

4.1 Calling Sequence.
   The FORTRAN calling sequence for Subroutine DIRSAT is as follows:
   
   CALL DIRSAT(IREC, QSATID, ISATAD, RTOCODE)
   
   The following list describes the arguments in the calling sequence.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IREC(16)</td>
<td>I*4</td>
<td>0</td>
<td>directory block</td>
</tr>
<tr>
<td>QSATID(12)</td>
<td>L*1</td>
<td>I</td>
<td>satellite ID</td>
</tr>
<tr>
<td>ISATAD</td>
<td>I*4</td>
<td>0</td>
<td>address of the satellite block</td>
</tr>
<tr>
<td>RTOCODE</td>
<td>I*4</td>
<td>0</td>
<td>return code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0= normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4= satellite block not found</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8= I/O error</td>
</tr>
</tbody>
</table>

4.2 Subroutine Called.

   Subroutine Description
   KCLC SACC - Logical compare byte by byte
   LREAD Unblocks and reads an MJS Log record

4.3 COMMON Variables.

   Variable Type COMMON Description
   IMES(26) I*4 FERMSG DAIO I/O error data

4.4 I/O Units.

   Unit No. Description
   06 terminal, output messages
   10 sysout A, hardcopy, data output
   11 terminal or data set, data output
   25 MJS Log
5. Procedure.
Reset codes and variables
Read directory and search satellite ID.
   If satellite ID, not found
       write error message
       set RTCODE = 4
       EXIT to RETURN
Copy directory record block to sent to calling routine
6. Special Notes.

6.1 Major Variables or Constants.
None

6.2 Data Structure and Tables.
See Appendix for the structure of the MJS Log directory.

6.3 Error Handling.
If the satellite ID does not match those listed in the directory block, the following message is written, the return code is set to 4, and a return taken.

SATELLITE ID, QSATID = AAAAAA IS INVALID.

If an I/O error is encountered while reading the MJS Log, the following message is written, the return code is set to 8 and a return taken.

I/O ERROR ON LOG.

DAIO error data from COMMON FERMSG is listed after the above message.
1. Subroutine FSTSER

2. Function

Subroutine FSTSER returns the first sequential serial assigned a TLS (Tape Library System) slot within a range of serials.

3. Programmer and Date

Eunice Eng, January 1978

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for FSTSER is as follows:

CALL FSTSER(LAST, ISPAN, DMSK, IFIRST)

The following list describes the arguments in the calling list.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAST</td>
<td>I*4</td>
<td>I</td>
<td>last serial in range</td>
</tr>
<tr>
<td>ISPAN</td>
<td>I*4</td>
<td>I</td>
<td>number of serials in range, used to calculate first serial in range</td>
</tr>
<tr>
<td>DMSK</td>
<td>R*8</td>
<td>I</td>
<td>vol-ser mask</td>
</tr>
<tr>
<td>IFIRST</td>
<td>I*4</td>
<td>0</td>
<td>first serial allocated a TLS slot in range. = 0, if no serial in range is assigned a slot</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

Subroutine | Description
---------------------------------------
ENCORE       | SACC - converts binary to EBCDIC
SHOW         | TLS - returns the slot a serial is assigned - returns a 0 if a slot is not assigned |

4.3 COMMON Variables

None

4.4 I/O Devices

None
5. Procedure

Calculate start serial
   First serial = last serial - (ISPAN* - 1)
DO until (serial with slot found)
   build mask for vol-ser
   quiz TLS if vol-ser is assigned a slot
OD
If (no slots have been assigned in the range)
   IFIRST = 0
FI
RETURN
   return

* ISPAN - number of tape slots allocated for the production process. The programmer is assuming the last ISPAN serials are those to be logically assigned a slot.

6. Special Notes

6.1 Major Variables and Constants
   None

6.2 Data Structure and Tables
   See Appendix for a description of the structure of the MJS Log.

6.3 Error Handling
   None
1. Routine: INILOG

2. System, Satellite, Version:
   Initialize MJS Log, Voyager-1 and -2, Ver. 1

3. English Name: Initialize Log

4. Language:
   FORTRAN or FORTRAN level 21.6
   360/91/75 OS/MVT

5. Purpose:
   Initialize MJS Log up through the control blocks for 2 satellites

6. Calling Sequence:
   Argument Type I/O Description
   None

7. Notes:
   7a. Restrictions: Only initializes for 2 satellites. MJS Log
       Directory block has room for 3 satellites.
   7b. Special Features: None

8. Variables:
   8a. Local
   Variable Type Description
   None
   8b. COMMON
   COMMON BLOCKS: QALK(84,15) - I

9. I/O Information:
   Unit No. Use Description
   25 MJS Log

10. Error Handling: None

11. Subroutines Called:
    Subroutine Description
    WRITE DAQ - write a record to disk buffer
    DCLOSE DAQ - sends last disk buffer to disk
    RTIME SACC - gets system date
    YMDH SACC - translates system date to yr/mo/da

12. Called By:
    Routine Description
    None

13. Method:
    Set date of creation from the system
    DO UNTIL (all control blocks written)
    Put in creation date
    Write a block of block data
    Close disk data set
    Stop

14. Reference:
    Format for MJS Log
    COMMON BLOCKS

15. Programmer and Date:

16. Modifications: None

*** END OF MEMBER *** 70 RECORDS PROCESSED *************
1. Routine: ALTER


3. English Name: Alter Block

4. Language: FORTRAN or FORTRAN Level 21.0 360/91/75 05/MAY

5. Purpose: Alter a byte in a given MJS Log block. Usually used to reset flags for processing dispositions. Designed for TSO (time sharing option) real-time execution.

6. Calling Sequence:
   Argument Type I/O Description
   None

7. Notes:
   7a. Restrictions: None
   7b. Special Features: Hardcopy documents why a change was made and what the change was.

8. Variables:
   8a. Local
      Variable Type Description
      IBLOCK I#4 Input card variable BLOCK. Block
      TBYTE I#4 Input card variable BYTE. Byte to be altered in BLOCK.
      QVAL I#1 Input card hex variable QVAL. Value put into BYTE of BLOCK.

   8b. COMMON
      COMMON Variables
      FERMNG I#2

9. I/O Information:
   Unit No. Use Description
   05 Input data, terminal
   11 Output data, terminal
   17 Output hardcopy, sysout=A
   23 MJS Log, disk data set

10. Error Handling:
    I/O error on MJS Log

11. Subroutines Called:
    Subroutine Description
    LS310 Reads MJS Log and returns correct record.
    LM310 Entry in LS310. Writes a record to MJS Log.

12. Called By:
    Routine Description
    None

13. Method:
    DO UNTIL (All input cards exhausted)
      Request, read and record reason for altering block
      Request, read and record block to alter
      Byte in block to alter
      Value that will go into the Byte
      Read block
      Replace Byte
      Write block back to MJS Log
      Display block is altered
    STOP

14. Reference:
    MJS Log Format
C15. Programmer and Date:
[Name and date]

C16. Modifications:
Ver. 2: Documentation feature of asking for a reason for altering was implemented.

*** END OF MEMBER ***  35 RECORDS PROCESSED  ****************************
1. **RWORK**

2. **Function**

**RWORK** searches the MJS Log work blocks for serials within the user specified range to be removed. If serials within the range have been completely processed, the serials are removed from TLS (tape Library System) slots and the MJS Log.

The serials are removed in sequence starting with the first, presently allocated MJS Work serial (within 25 serials of the last serial).

3. **Programmer and Date**

Eunice Eng, February 1978

4. **Interface**

4.1 **Calling Sequence**

None

4.2 **Subroutines Called**

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNC</td>
<td>SACC - logical AND</td>
</tr>
<tr>
<td>KOC</td>
<td>SACC - logical OR</td>
</tr>
<tr>
<td>KCLC</td>
<td>SACC - logical compare</td>
</tr>
<tr>
<td>KMVC</td>
<td>SACC - logical byte move</td>
</tr>
<tr>
<td>LREAD</td>
<td>deblocks and reads a MJS Log record</td>
</tr>
<tr>
<td>DIRSAT</td>
<td>reads the MJS directory for the satellite block number</td>
</tr>
<tr>
<td>FSTSER</td>
<td>quizzes TLS to locate the first serial, within 25 from the last serial, that has been assigned a TLS slot</td>
</tr>
<tr>
<td>INCORE</td>
<td>SACC - converts binary data to EBCDIC data</td>
</tr>
<tr>
<td>REMOVE</td>
<td>removes a vol-ser from TLS</td>
</tr>
</tbody>
</table>

4.3 **COMMON Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DAIO I/O error data</td>
</tr>
</tbody>
</table>
4.4 I/O Devices

<table>
<thead>
<tr>
<th>Unit NO.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>terminal, input data</td>
</tr>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy of data output</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>

4.5 Input

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID(12)</td>
<td>L*1</td>
<td>satellite ID. User only inputs satellite number, QSATID(9)</td>
</tr>
<tr>
<td>ISER1</td>
<td>I*4</td>
<td>first serial in the range of serials to be removed</td>
</tr>
<tr>
<td>ISERN</td>
<td>I*4</td>
<td>last serial in the range of serials to be removed</td>
</tr>
</tbody>
</table>

Format

1. satellite ID card
   column 12345
   or
   1 or 2

2. serial range card
   column 12345678901234567890
   prompt first last
   XXX XXX
   ISER1 ISERN
5. Procedure

Identify program on hard copy
Get satellite ID from input
Read directory for satellite block no
   If (satellite block not found)
       EXIT to STOP
Request and Read satellite range
Prepare vol-ser mask
Check ISER1 and ISERN
   If (ISER1 > ISERN)
       Write error message
       Exit To STOP
Read satellite block for Work Control block
Read Work Control block
   If (ISER1 ≠ XXXX first serial assigned a slot,
       within 25 from last serial)
       Write error message
       EXIT to STOP
   If (ISERN > last serial used)
       Write error message
       EXIT to STOP
Read Work blocks in backward link until ISER1 found
   If (ISER1 not found)
       Write error message
       EXIT To STOP
Find first serial that has NOT been fully processed
   If (processing disposition not XXXX both dead (200)
       or both done (220))
       If (present serial ≤ ISERN) i.e. a serial
           within the remove range cannot be removed
           Write error message
           EXIT to STOP
Remove serials from TLS slots
DO until (serial = ISERN)
   build vol-ser form serial
   Remove serials from TLS slots
OD
STOP
Stop
6. Special Notes
6.1 Variables and Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSK</td>
<td>R*8</td>
<td>vol-ser of tape to be removed from TLS</td>
</tr>
<tr>
<td>QTMSK(8)</td>
<td>L*1</td>
<td>equivalent to DMSK</td>
</tr>
<tr>
<td>ISERN</td>
<td>I*4</td>
<td>last serial to be removed, in user given range</td>
</tr>
<tr>
<td>ISER1</td>
<td>I*4</td>
<td>first serial in range to be removed</td>
</tr>
<tr>
<td>IWRKC</td>
<td>I*4</td>
<td>MJS block number for the Work Control</td>
</tr>
<tr>
<td>QDONE</td>
<td>L*1</td>
<td>z20, constant a that flags process completed, in disposition field</td>
</tr>
<tr>
<td>ISATAD</td>
<td>L*4</td>
<td>number of satellite block in MJS Log</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables
See Appendix for a description of the structure of a Work block xxxxxj in the MJS Log.

6.3 Error Handling
The user input serial range is checked for order and validity. Should one of the following messages appear, the job is terminated.

FIRST (= XXX) SHOULD BE LESS THAN OR EQUAL TO LAST (= XXX).

***** NONE OF THE SERIALS ARE ASSIGNED TO A TLS SLOT.

The above message implies that the last 25 serials in the work block are not currently assigned in a TLS slot.

ISER1 (= XXX) IS NOT EQUAL TO THE FIRST CURRENTLY ASSIGNED SERIAL IN THE WORK CONTROL BLOCK (= YYY)

YYY is the first currently assigned serial in the Control block.

THE LAST SERIAL TO BE REMOVED (ISERN = XXX) IS OUT OF RANGE.

THE LAST USED SERIAL IN THE WORK CONTROL BLOCK IS XXX.

ISER1 = XXX NOT FOUND.

The last message flags a maximum serious error. Although TLS has the serial assigned the MJS Log does not have any record of the slot having been used. Check the latest Log listing.
SERIAL XXX WITHIN RANGE HAS NOT BEEN PROCESSED.

If the above message is printed, none of the serials have been removed. Reenter your range, stopping before the above mentioned serial.

I/O ERROR ON LOG.

The above message is followed by data from DAIO I/O COMMON FERMSG.
1. **ASSIGN**

2. **Function**

   ASSIGN assigns tape vol-sers to TLS (Tape Library System) slots that are available. ASSIGN assigns all available slots and updates the Work or CIT Control block in the MJS Log. ASSIGN is meant to run on TSO (Time Sharing Option).

3. **Programmer and Date**

   Eunice Eng, February 1978

4. **Interface**

4.1 **Calling Sequence**

   None

4.2 **Subroutines called**

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCIC</td>
<td>SACC - logical compare</td>
</tr>
<tr>
<td>ASSIGN</td>
<td>assigns slots to vol-ser in TLS</td>
</tr>
<tr>
<td>SHOW</td>
<td>displays the slot a vol-ser is assigned or vice-versa</td>
</tr>
<tr>
<td>LREAD</td>
<td>deblocks and reads and/or writes a MJS Log record</td>
</tr>
<tr>
<td>(LWRITE)</td>
<td></td>
</tr>
<tr>
<td>DIRSAT</td>
<td>reads the directory xxx for the satellite block number</td>
</tr>
<tr>
<td>KMVC</td>
<td>SACC - logical byte move</td>
</tr>
</tbody>
</table>

4.3 **COMMON Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DAIO I/O error data</td>
</tr>
</tbody>
</table>

4.4 **I/O Devices**

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>terminal, input data</td>
</tr>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy of session</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>
### 4.5 Input

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID(12)</td>
<td>I$^*$1</td>
<td>satellite ID. Program requests only the satellite number, QSATID(9).</td>
</tr>
<tr>
<td>QTYPE(4)</td>
<td>I$^*$4</td>
<td>type of tapes to assign. 'WORK' = assign work tapes 'CIT' = assign CIT tapes</td>
</tr>
<tr>
<td>MAX</td>
<td>I$^*$4</td>
<td>maximum amount of tapes to assign.</td>
</tr>
</tbody>
</table>

**Format**

1. **Satellite ID card**
   - column 12345
     - 1
     - or
     - 2

2. **Tape type card**
   - column 1234567890123456789
     - work
     - cit

3. **Maximum amount card**
   - column 12345
     - XX
     - \(0\) to 24
5. Procedure

Request and Read all input
Get satellite ID
Get tape type to assign
Get maximum amount of tapes to assign (defaults to 24 or 25)
If (maximum > 24)
   write error message
   EXIT to END

Set-up to Assign
Reset number of assigns (IDID = 0)
Set-up vol-ser mask
Read directory block for satellite block
   If (satellite id invalid)
      (message from SUBROUTINE DIRSAT)
      EXIT to END
Read satellite block for control block numbers
IWRKA = address of Work Control
ICITA = address of CIT Control
If (tape type = 'WORK')
   Fill second byte of vol-ser mask (QMSK(2) = 'w')
   If (defaulted maximum, i.e. MAX = 0)
      MAX = 25
   FI
   MJS block to be read assigned, IBLOCK = IWRKA
   EXIT to ASSIGN
If(tape type = 'CIT ')
   Fill second byte of vol-ser mask, QMSK(2) = 'C'
   If (defaulted maximum, ie MAX = 0)
      MAX = 24
   FI
   MJS block to be read assigned, IBLOCK = ICITA
   EXIT to ASSIGN
Else write error message
   EXIT TO END

ASSIGN
Ready to assign slots
Read Control block
   LSTSER = last xrrxx serial assigned in Control
   IISLOT = first slot in range of slots used
   INSLOT = last slot in range of slots used
DO until (all slots in range checked)
   If (slot = ' EMPTY')
      Increment last serial, LSTSER = LSTSER + 1
   If (LSTSER > 999)
      Write error message
      Decrement LSTSER, LSTSER = LSTSER - 1
      EXIT to END
   ELSE assign LSTSER to present slot
   Write message of assignment
   Count # of assigns, IDID = IDID + 1
   If (IDID = MAX)
      EXIT to WRITE

OD
WRITE
   Update Control block
   Record last serial assigned

END
Stop
6. Special Notes

6.1 Major Variables and/or Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX</td>
<td>I*4</td>
<td>maximum amount of assigns requested by user. Defaults 25 for WORK, 24 for CIT</td>
</tr>
<tr>
<td>DMSK</td>
<td>R*8</td>
<td>EBCDIC vol-ser</td>
</tr>
<tr>
<td>QMSK(8)</td>
<td>L*1</td>
<td>equivalent to DMSK</td>
</tr>
<tr>
<td>IDID</td>
<td>I*4</td>
<td>number of assigns completed</td>
</tr>
<tr>
<td>ICITA</td>
<td>I*4</td>
<td>block number of CIT Control</td>
</tr>
<tr>
<td>ISLOT</td>
<td>I*4</td>
<td>current TLS slot</td>
</tr>
<tr>
<td>TWRKA</td>
<td>I*4</td>
<td>block number of WORK Control</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Appendix for the structure of the Control blocks in the MJS Log.

See Table for the structure of the TLS commands.
6.3 Error Handling

If an invalid satellite number is passed, Subroutine DIRSAT writes the following error message and the program is terminated. User should check input.

SATELLITE ID, QSATID = AAAAAAAAAAA IS INVALID.

If the user requests a maximum number of assigns greater than 24, the following message is printed and the program is terminated. Request less than 24 or check input to be right justified in a field of 2 digits (i.e. 05 instead of 5_ (=50)).

MAX GREATER THAN 24.

If a tape type other than 'WORK' or 'CIT' is requested the following message is printed and the program terminated. Check input.

INVALID QTYPE - AAAA - CHECK INPUT.

If serials should exceed 999, the following message is printed describing the condition and action taken.

LAST SERIAL = XXXX NOW EXCEEDS 999. UPDATING CONTROL AND TERMINATING.

Note the last serial recorded will be 999.

If an I/O error occurs while reading or writing to the MJS Log, the following message is written along with DAIO error data from COMMON FERMSG before the program is terminated.

I/O ERROR ON MJS LOG.
1. **RCIT**

2. **Function**

   RCIT searches the MJS Log CIT blocks for serials that fall within the user specified range to be removed. If the serials within the range are marked for removal, the serials are removed from TLS (Tape Library System) and the MJS Log.

   The serials are removed in sequence, starting with the first presently allocated CIT serial within 24 serials of the last serial.

3. **Programmer and Date**

   Eunice Eng, January 1978

4. **Interface**

4.1 **Calling Sequence**

   None

4.2 **Subroutines Called**

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNC</td>
<td>SACC - logical AND</td>
</tr>
<tr>
<td>KOC</td>
<td>SACC - logical OR</td>
</tr>
<tr>
<td>KCLC</td>
<td>SACC - logical compare</td>
</tr>
<tr>
<td>KMVC</td>
<td>SACC - logical byte move</td>
</tr>
<tr>
<td>LREAD</td>
<td>deblocks and reads and/or writes a MJS Log record (LWRITE)</td>
</tr>
<tr>
<td>DIRSAT</td>
<td>reads the MJS Log directory for satellite block</td>
</tr>
<tr>
<td>FSTSER</td>
<td>locates the first serial assigned a TLS slot</td>
</tr>
<tr>
<td>INCORE</td>
<td>SACC - converts binary data to EBCDIC</td>
</tr>
<tr>
<td>REMOVE</td>
<td>removes a vol-ser from TLS</td>
</tr>
</tbody>
</table>

4.3 **COMMON Variable**

<table>
<thead>
<tr>
<th>Variable</th>
<th>type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>DAIO I/O error data</td>
</tr>
</tbody>
</table>
4.4 I/O Devices

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>terminal, input data</td>
</tr>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy of session</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>

4.5 Input

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID(12)</td>
<td>I*1</td>
<td>satellite ID. Only the satellite number is requested QSATID(9).</td>
</tr>
<tr>
<td>ISER1</td>
<td>I*4</td>
<td>first serial in range to be removed</td>
</tr>
<tr>
<td>ISERN</td>
<td>I*4</td>
<td>last serial in range to be removed</td>
</tr>
</tbody>
</table>

Format

1. satellite id card
   column 12345
   1 or 2

2. serial range card
   column 12345678901234567890
   prompt first last
   | XXX | XXX |
   | ISER1 | ISERN |
5. Procedure

Identify program on hardcopy

Get satellite ID.

Read Directory for satellite block number
   If satellite block not found
       EXIT to STOP

Request and read satellite range

Prepare vol-ser mask

Check ISER1 and ISERN
   If (ISER1 > ISERN)
       Write error message
       EXIT to STOP
   If (ISER1 not odd)
       Write error message
       EXIT to STOP
   If (ISERN not even)
       Write error message
       EXIT to STOP

Read satellite block for CIT Control block

Read CIT Control block
   Check if remove request out of range.
   Find the first serial assigned a TLS slot within 24
       serials from the last serial
       If (last 24 serials not assigned a TLS slot)
           Write error message
           EXIT to STOP
   If (ISER1 first assigned serial .AND. ISERN > last used
       serial)
       Write error message
       EXIT to STOP

Read CIT blocks, linking backward until block with ISER1 found.
   If (block not found)
       Write error message
       EXIT to STOP

REMOVE

Read CIT block
   If (serial not marked for removal)
       EXIT to STOP
   If (serial to be removed > ISERN)
       EXIT to STOP
Remove serial from TLS
   Write successful remove message
Increment serial by one
Remove serial from TLS
   Write successful remove message
Update MJS Log block - slots removed
Get address of next block
EXIT to REMOVE

STOP

stop
6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSK</td>
<td>R*8</td>
<td>vol-ser of tape to be removed</td>
</tr>
<tr>
<td>QMSK(8)</td>
<td>I*1</td>
<td>equivalent to DMSK</td>
</tr>
<tr>
<td>IGONE</td>
<td>I*4</td>
<td>binary serial of CIT tape to be removed</td>
</tr>
<tr>
<td>QGONE</td>
<td>I*1</td>
<td>z10, flags serial removed from TLS slot in MJS Log</td>
</tr>
<tr>
<td>IFIRST</td>
<td>I*4</td>
<td>first current serial within 24 serials of the last serial to be assigned a TLS slot. If = 0, no serials were assigned a slot.</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables.

See Appendix for a description of the structure of the MJS Log.

6.3 Error Handling.

The first and last serial in the user requested range of removal is checked. If the serials fail to meet the following tests, a message will be printed and the program is terminated. The following messages are self explanatory.

ISER1 = XXX IS GREATER THAN ISERN = XXX.

FIRST MUST BE ODD.  \( \neq \) \{ \text{FIRST = ISER1} \}
LAST MUST BE EVEN. \( \neq \{ \text{LAST = ISERN} \}

For the above messages the user should consult a list of the MJS Log or check typed input.

THE SERIALS IN THE REMOVE REQUEST IS OUT OF RANGE (XXX TO YYY).

In this message the existing valid range is given in parenthesis. Check user input.

The following message

***** NONE OF THE SERIALS ARE ASSIGNED TO TLS SLOTS.

means there are at present no CIT tapes to be removed. TLS has them all removed. If the MJS Log does not reflect this, TLS or the MJS log must be updated. (MJS Log through ALTBLK. TLS through tlsupdte)
The following message occurs if all the CIT blocks have been searched and the user requested first serial in the range has not been found. The program is terminated.

\texttt{ISERI = XXX NOT FOUND.}

This message indicates there is an error in the MJS Log. Check the latest MJS listing. TLS has the serial assigned. May have to update the Log through program ALTBLK.

6.4 Other

The first serial not marked for removal, in the range of serials, terminates the program. Therefore, it is possible that the entire range may not have been removed. The hardcopy will let you know which tapes have been removed from TLS. If the program did not terminate properly, check a log list to see if the vol-sers have been removed from the Log.
1. REDR

2. Function

REDR locates the MJS Log EDR block which contains the user specified vol-ser. REDR removes the vol-ser from its TLS (Tape Library System) slot and the MJS Log.

3. Programmer and Date

Eunice Eng, February 1978

4. Interface

4.1 Calling Sequence

NONE

4.2 Subroutines Called

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNC</td>
<td>SACC - Logical AND</td>
</tr>
<tr>
<td>KOC</td>
<td>SACC - Logical OR</td>
</tr>
<tr>
<td>KCLC</td>
<td>SACC - Logical compare</td>
</tr>
<tr>
<td>KMVC</td>
<td>SACC - logical byte move</td>
</tr>
<tr>
<td>LREAD</td>
<td>deblocks and reads and/or writes a MJS Log record</td>
</tr>
<tr>
<td>WRITE</td>
<td></td>
</tr>
<tr>
<td>DIRSATS</td>
<td>reads the MJS Log directory block for the satellite block number</td>
</tr>
<tr>
<td>REMOVE</td>
<td>removes a vol-ser from a TLS slot</td>
</tr>
</tbody>
</table>

4.3 COMMON Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DAIO I/O error data</td>
</tr>
</tbody>
</table>

4.4 I/O Devices

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>terminal, input data</td>
</tr>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy of data output</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>
### 4.5 Input

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID(12)</td>
<td>L*1</td>
<td>satellite ID. The program only requests the satellite number, QSATID(9).</td>
</tr>
<tr>
<td>QVOL(8)</td>
<td>L*1</td>
<td>QVOL(3)-QVOL(8), the 6 digit vol-serp of the EDR tapes to be removed, one at a time.</td>
</tr>
</tbody>
</table>

**Format**

1. **Satellite id card**
   
   column 12345 

2. **vol-ser card**
   
   column 123456789 
   
   AAAAAA 
   
   vol-ser
5. Procedure

Get satellite ID.
Read Directory for satellite block number
If (satellite block not found)
   EXIT to STOP
Read satellite block for EDR Control
Read EDR Control for last EDR block number

GETEDR
Read input, vol-ser to be removed
if(no more vol-ser to be removed)
   EXIT to STOP
DO until (block w/ vol-ser found, reading EDR blocks in backward link)
   If(vol-ser found)
      If (slot marked for removal)
         Remove vol-ser from TLS
         Remove vol-ser from MJS Log
         Write success message
         EXIT to GETEDR
      Else (slot not marked for removal)
         Write error message
         Exit to GETEDR
   OD
If (vol-ser not found)
   Write error message
   EXIT to GETEDR

STOP
Stop
6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVOL</td>
<td>R*8</td>
<td>vol-ser of tape to be removed. Right justified, padded to left blanks.</td>
</tr>
<tr>
<td>QVOL(8)</td>
<td>I*4</td>
<td>equivalent to DVOL.</td>
</tr>
<tr>
<td>ILAST</td>
<td>I*4</td>
<td>block number of last EDR block in EDR Control.</td>
</tr>
<tr>
<td>QGONE</td>
<td>L*1</td>
<td>z10, constant to flag that the vol-ser has been removed</td>
</tr>
<tr>
<td>QREMOV</td>
<td>L*1</td>
<td>z20, constant to flag a vol-ser is removable</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Appendix for a description of the structure of the MJS Log.

6.3 Error Handling

If one of the following two messages is printed, no removal is performed and the program will request another vol-ser.

AAAAAAA\(^1\) VOL-SER NOT FOUND FOR AAAAAAAA\(^2\).

AAAAAAA\(^3\) NOT MARKED FOR REMOVAL.

If an I/O error is detected on the MJS Log the following message is printed followed by I/O data from DAIO COMMON FERMSG.

I/O ERROR ENCOUNTERED ON LOG.

\(^1\) vol-ser

\(^2\) satellite ID

\(^3\) vol-ser
1. **RWORK**

2. **Function**

   **RWORK** searches the MJS Log work blocks for serials the user requests to be removed. If found and marked to be removed, i.e. CIT and Merge process done, the serial is removed from TLS.

3. **Programmer and Date**

   Eunice Eng, February 1978

4. **Interface**

4.1 **Calling Sequence**

   None

4.2 **Subroutines Called**

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNC</td>
<td>SACC - logical AND</td>
</tr>
<tr>
<td>KOC</td>
<td>SACC - logical OR</td>
</tr>
<tr>
<td>KCLC</td>
<td>SACC - logical compare</td>
</tr>
<tr>
<td>KMVC</td>
<td>SACC - logical byte move</td>
</tr>
<tr>
<td>DREAD</td>
<td>SACC - DAIO reads a disk data set</td>
</tr>
<tr>
<td>DIRSAT</td>
<td>reads the MJS directory for the satellite block number.</td>
</tr>
<tr>
<td>INCORE</td>
<td>SACC - converts binary data to EBCDIC data</td>
</tr>
<tr>
<td>REMOVE</td>
<td>removes a vol-ser from TLS</td>
</tr>
</tbody>
</table>

4.3 **COMMON Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DAIO I/O error data, input</td>
</tr>
</tbody>
</table>

4.4 **I/O Devices**

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>terminal, input data</td>
</tr>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hard copy of data output</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>
### 4.5 Input

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID(12)</td>
<td>L*1</td>
<td>satellite ID. User only inputs satellite number, QSATID(9)</td>
</tr>
<tr>
<td>ISER</td>
<td>I*4</td>
<td>serial to be removed</td>
</tr>
</tbody>
</table>

#### Format

1. satellite ID card
   - column 12345
   - 1
   - or
   - 2

2. serial to be removed card
   - column 12345
   - XXX use: all three digits right justified.
5. Procedure

Identify program on hard copy
Set-up, Request and Read Satellite ID
Read Log directory
IF (error)
    EXIT to STOP
Read satellite block
Read Work control block
LASTB = last block allocated
LASTS = last serial allocated

SER REMOVE LOOP
Request, Read and Echo serial
    Build Mask for vol-ser
    Ask, Read & Echo user serial
    If (no more requests)
        EXIT to SER
Get address of first block to search, ILASTB
IF (user serial .GT. Last log serial)
    Write error message
    EXIT to SER
DO UNTIL (all blocks read in a backward link)
Read a work block
IF (user serial used by current work block)
    IF (user serial can be removed)
        build vol-ser to be removed
        remove vol-ser from TLS
        write remove message
        EXIT to SER
    ELSE
        write error message
        EXIT to SER
    FI
FI
Get address of next work block to read
OD
Write error message (serial not found)
EXIT to SER

STOP Stop
End
6. Special Notes

6.1 Variables and Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSK</td>
<td>R*8</td>
<td>Vol-ser of tape to be removed from TLS</td>
</tr>
<tr>
<td>QMSK(8)</td>
<td>L*1</td>
<td>equivalent to DMSK</td>
</tr>
<tr>
<td>ILASTB</td>
<td>I*4</td>
<td>last work block assigned</td>
</tr>
<tr>
<td>ILASTS</td>
<td>I*4</td>
<td>last work serial assigned TLS slot</td>
</tr>
<tr>
<td>ISER</td>
<td>I*4</td>
<td>serial to be removed</td>
</tr>
<tr>
<td>IWRKC</td>
<td>I*4</td>
<td>MJS block number for work control block</td>
</tr>
<tr>
<td>QDONE</td>
<td>L*1</td>
<td>z20, constant that flags process completed, in disposition field</td>
</tr>
<tr>
<td>IBATAD</td>
<td>I*4</td>
<td>number of satellite block in MJS Log</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Appendix for a description of the structure of a work block in the MJS Log.

6.3 Error Handling

The following messages inform the user that the requested serial was not removed for the stated reason. These messages do not terminate the program. The next serial to be removed is requested.

SERIAL REQUESTED FOR REMOVAL XXX EXCEEDS LAST SERIAL ASSIGNED YYY.

XXX SERIAL NOT MARKED FOR REMOVAL.

XXX SERIAL NOT FOUND.

An I/O error encountered on the log produces the following message before the program is terminated.

I/O ERROR on LOG.

The above message is followed by data from DAIO I/O COMMON FERMSG.
1. CALLIST

2. Function.

CALLIST is a main calling routine for the MJS Log list programs.

3. Programmer and Date.

Eunice Eng, January 1978

4. Interface.

4.1 Calling Sequence.

None

4.2 Subroutines Called.

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCLC</td>
<td>SACC - logical compare byte by byte</td>
</tr>
<tr>
<td>BAKLOG</td>
<td>displays the MJS outstanding requests</td>
</tr>
<tr>
<td>DMPLOG</td>
<td>gives a hexadecimal dump of records within a user given range</td>
</tr>
<tr>
<td>INCOMP</td>
<td>displays the incomplete processes of the MJS Log.</td>
</tr>
<tr>
<td>LISTAL</td>
<td>a calling subroutine to display the requested types of blocks</td>
</tr>
<tr>
<td>REMSLT</td>
<td>displays the remaining slots of the library work and CIT controls.</td>
</tr>
<tr>
<td>SHWCIT</td>
<td>a calling subroutine to display the CIT slots marked for removal.</td>
</tr>
<tr>
<td>SHWEDR</td>
<td>a calling subroutine to display the EDR slots marked for removal.</td>
</tr>
</tbody>
</table>

4.3 COMMON Variables.

The following describes the variable in COMMON FERMSG.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DAIO COMMON that displays data of an I/O error.</td>
</tr>
</tbody>
</table>

4.4 I/O Devices

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>terminal, data max input</td>
</tr>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy, data output</td>
</tr>
<tr>
<td>11</td>
<td>terminal or data set, data output</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>
4.5 Input.

CALLIST requests the input for the list subroutines. CALLIST requests the following for all the subroutines.

**Menu number**

**MENU** I*4 number associated with a list subroutine

**Format**

column 1234567890

**Satellite ID.**

**QSATID(12) L*1 satellite ID., (EBCDIC)**

**Format**

11111111112

column 12345678901234567890

voyager-X

The following describes input requested for the BAKLOG and INCOMP list programs.

**Response card**

**QSA Y(4) L*1 Yes or No response for a request to list backlog types. Default = no**

**Format**

column 12345

yes

no

**Type card**

**ITYP E I*4 numeric code for type of backlog or incomplete processes to list**

**Format**

column 123456789

The following describes input requested for the LISTAL list subroutine.

**Response Card**

see above

**Type Card**

**IBLOCK(10) I*4 list of list all block types**

**Format**

column 123456789012345678901234567890

XX XX XX XX XX XX XX XX XX
The following describes input requested for the DMPLOG subroutine.

**RANGE card**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IST</td>
<td>I*4</td>
<td>first block to be dumped. Must be given.</td>
</tr>
<tr>
<td>LEND</td>
<td>I*4</td>
<td>last block to be dumped. Defaults to the last block allocated, recorded in Log directory.</td>
</tr>
</tbody>
</table>
5. Procedure.

MENU Request subroutine by menu number
If (no more requests) EXIT to END
If (menu = 7) go to DUMFLOG
Request satellite ID.
If (menu no. = 1)
  Call SHWEDR, subprogram to list max EDR slots marked for removal
  EXIT to MENU
If (menu no = 2)
  Call SHWCT, subprogram to list CIT slots marked for removal
  EXIT to MENU
If (menu no = 3)
  Call REMSLOT, subprogram to list remaining tape slots in Library, Work and CIT controls.
  EXIT to MENU
IF (menu no = 4)
  If (list of backlog types requested)
    list codes associated with types.
  Fi
Request backlog types
Call BAKLOG, subroutine to list outstanding requests of the named type
  EXIT to MENU
If (menu no = 5)
  If (list of incomplete types requested)
    list codes associated with incomplete types
  Fi
request incomplete process type
Call INCOMP, subroutine to display incomplete processes of the named type.
  EXIT to MENU
If (menu no = 6)
  If (list of block types requested)
    list codes associated with list all types
  Fi
Request list of 'list all' block types
Call LISTAL, calling subroutine to display the named block types
  EXIT to MENU
DMPLOG
If (menu no = 7)
    Set last record to be dumped = 0, causes a default to list all records to the last record.
    Request and read first and last records to be read
    Call DMPLOG, subroutine that gives a hexadecimal dump of the records in the given range.
    EXIT to MENU
Write error message ('Wrong menu number')
EXIT TO XXXX MENU
END
Stop
End
6. Special Notes

6.1 Major Variables and Constants
None

6.2 Data Structure and Tables.
See Appendix for a description of the structure of the MJS Log.

6.3 Error Handling.
The satellite ID is checked in the list subroutines.

If the wrong menu number is input, the following message is written and another menu number is requested.
WRONG MENU NUMBER.

If a data set is given for unit 11, is filled, a TSO message is written, and the program execution is terminated.
(The CLIST - 'ZBEKE.LIB.CLIST(LISTER)' - allocates the data set with 60 tracks.)

6.4 Other
There is a TSO CLIST that will run CALLIST in foreground from a load module. The CLIST will list the CALLIST's menu if requested.
1. Subroutine BAKLOG

2. Function

BAKLOG displays the outstanding production processes monitored by the MJS Log.

3. Programmer and Date

Eunice Eng, January 1978

4. Interface

4.1. Calling Sequence.

The FORTRAN calling sequence for BAKLOG is as follows:

CALL BAKLOG(QSATID, ITYPE)

The following list describes the arguments in the calling list.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID(12)</td>
<td>I*1</td>
<td>I</td>
<td>satellite ID</td>
</tr>
</tbody>
</table>
| ITYPE | I*4 | I | number associated with the type of backlog
| =01, all of the below |
| =03, EDR processing |
| =04, Encyclopedia generation |
| =05, Merge and CIT processing |

4.2 Subroutines Called

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCLC</td>
<td>SACC - logical compare</td>
</tr>
<tr>
<td>LREAD</td>
<td>deblocks and reads an MJS Log record</td>
</tr>
<tr>
<td>DIRSAT</td>
<td>reads and returns the MJS Log directory block, also locating the satellite block address</td>
</tr>
<tr>
<td>DLCUPK</td>
<td>Unpacks the R*8 FDSC for the MJS satellite</td>
</tr>
</tbody>
</table>

4.3 COMMON Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>I/O I/O error data</td>
</tr>
</tbody>
</table>
### 4.4 I/O Devices

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy, data output</td>
</tr>
<tr>
<td>11</td>
<td>terminal or data set, data output</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>
5. Procedure

Read directory for satellite block number
   If (satellite block not found)
       EXIT to RETURN

Read Satellite block for CONTROL block numbers
   IEDR, EDR control block
   ILIB, Library control block
   IWRK, Work control block

   If EDR requested
       Read EDR control block
       If no EDR blocks present
           Write error message
           EXIT to LIB
       Write headers
       DO until (all EDR blocks searched)
           If marked for processing, list
        OD
    FI

LIB   If Library requested
       Read Library control block
       If (no Library blocks present)
           Write error message
           EXIT to WORK
       Write headers
       DO until (all library blocks read)
           If marked for processing, list
        OD
    FI

WORK   If work requested
       Read Work control block
       If no work blocks present
           Write error message
           EXIT to RETURN
       Write headers
       DO until (all work blocks read)
           If marked for processing, list
        OD
    FI

RETURN
   return
6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILIB</td>
<td>I*4</td>
<td>Library Control block number</td>
</tr>
<tr>
<td>IEDR</td>
<td>I*4</td>
<td>EDR Control block number</td>
</tr>
<tr>
<td>IWRK</td>
<td>I*4</td>
<td>Work Control block number</td>
</tr>
<tr>
<td>QTODO</td>
<td>L*1</td>
<td>z80, constant that flags a process requested</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Table

See Appendix for the data structure of the MJS Log.

6.3 Error Handling

If there are no EDR, Library or Work blocks for a given satellite, the following messages are printed respectfully and execution continues.

NO EDR BLOCKS FORAAAAAAA IN LOG AT PRESENT.

NO LIBRARY BLOCKS FORAAAAAAA IN LOG AT PRESENT.

NO WORK BLOCKS FORAAAAAAA IN LOG AT PRESENT.

AAAAAAA = satellite ID

If an I/O error occurs while reading the MJS Log the following message is written followed by I/O data from DAIO COMMON FERMSG. The subroutine returns to the calling routine.

I/O ERROR ON LOG.
1. Subroutine CITRMV

2. Function
   CITRMV displays the CIT tapes marked for removal.

3. Programmer and Date
   Eunice Eng, January 1978

4. Interface

4.1 Calling Sequence
   The FORTRAN calling sequence for CITRMV is as follows:
   
   CALL CITRMV(ICIT, ISATCD)

   The following list describes the arguments in the calling sequence.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICIT</td>
<td>I*4</td>
<td>I</td>
<td>CIT control block number</td>
</tr>
<tr>
<td>ISATCD</td>
<td>I*4</td>
<td>I</td>
<td>satellite code</td>
</tr>
</tbody>
</table>

4.2 Subroutines called

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNC</td>
<td>SACC - logical AND</td>
</tr>
<tr>
<td>KOC</td>
<td>SACC - logical OR</td>
</tr>
<tr>
<td>KCIC</td>
<td>SACC - logical compare</td>
</tr>
<tr>
<td>LREAD</td>
<td>deblocks and reads an MJS Log record</td>
</tr>
<tr>
<td>INCORE</td>
<td>SACC - changes binary data to EBCDIC</td>
</tr>
</tbody>
</table>

4.3 COMMON Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DAIO I/O error data</td>
</tr>
</tbody>
</table>
4.4 I/O Devices

<table>
<thead>
<tr>
<th>Unit NO.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy of data output</td>
</tr>
<tr>
<td>11</td>
<td>terminal or data set, data output</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>

5. Procedure

Get Satellite code

Read CIT Control block to get first CIT block

DO until (all CIT blocks read)
   If tape marked for removal, list
   FI
OD

Return

6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QREMOV</td>
<td>L*1</td>
<td>z20, constant to flag a slot marked for removal</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Appendix for the data structure of the MJS Log.

6.3 Error Handling

If an I/O error occurs while reading the MJS Log, the following message is written followed by I/O data from DAIO COMMON FERMSG. The subroutine then returns to the calling routine.

   I/O ERROR ON LOG.
1. Subroutine DMPLOG

2. Function

DMPLOG dumps the MJS Log in hexadecimal within a given range of blocks.

3. Programmer and Date

Eunice Eng, January 1978

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for DMPLOG is as follows:

```
CALL DMPLOG(IST,IEND)
```

The following list describes the arguments in the calling list.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IST</td>
<td>I*4</td>
<td>I</td>
<td>first block to be dumped.</td>
</tr>
<tr>
<td>IEND</td>
<td>I*4</td>
<td>I</td>
<td>last block to be dumped, default to last record allocated</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

Subroutine Description

LREAD deblocks and reads a record from the MJS Log

4.3 COMMON Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FERMSG</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DAIO I/O error data</td>
</tr>
</tbody>
</table>

4.4 I/O Devices

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy of data output</td>
</tr>
<tr>
<td>11</td>
<td>terminal or data set, data output</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>
5. Procedure

If (IEND = 0)
    set default to last record allocated, which is recorded in the
directory
FI

DO until (range of blocks exhausted)
    read a block
    dumb the block in hexadecimal format
OD

Return

6. Special Notes

6.1 Major Variables and Constants

None

6.2 Data Structure and Tables

To interpret the dump, see Appendix for a description
of the blocks in the MJS Log.

6.3 Error Handling

If an I/O error occurs during while reading the MJS Log, the following
message is written followed by data from COMMON FERMSG from the
DAIO routines. The subroutine then returns to the calling
routine.

I/O ERROR ON LOG.
1. Subroutine EDRRMV

2. Function

   EDRRMV displays all the EDR tapes marked for removal. EDRRMV is patterned after CITRMV.

3 - 6 See Subroutine CITRMV.
1. Subroutine INCOMP

2. Function

Subroutine INCOMP displays the incomplete processes noted in the MJS Log. INCOMP is patterned after DMLOG.

3 - 6 See Subroutine DMLOG.
1. Subroutine LISTAL

2. Function

LISTAL is a driver to the specialized 'list all' subroutines, which lists given blocks of the MJS Log.

3. Programmer and Date

Eunice Eng, January 1978

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for LISTAL is as follows:

CALL LISTAL(QSATID,IBLOCK)

The following list describes the arguments in the calling list.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID(12)</td>
<td>I*1</td>
<td>I</td>
<td>satellite ID</td>
</tr>
<tr>
<td>IBLOCK(19)</td>
<td>I*4</td>
<td>I</td>
<td>list of block types to be listed</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCIT</td>
<td>displays CIT records</td>
</tr>
<tr>
<td>LEDR</td>
<td>displays EDR records</td>
</tr>
<tr>
<td>LENC</td>
<td>displays Encyclopedia records</td>
</tr>
<tr>
<td>LLIB</td>
<td>displays Library records</td>
</tr>
<tr>
<td>LWRK</td>
<td>displays Work records</td>
</tr>
<tr>
<td>LATTR</td>
<td>displays Attribute blocks</td>
</tr>
<tr>
<td>LREAD</td>
<td>deblocks and reads an MJS Log record</td>
</tr>
</tbody>
</table>

4.3 COMMON Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DAIO I/O error data</td>
</tr>
</tbody>
</table>

4.4 I/O Devices

<table>
<thead>
<tr>
<th>Unit NO.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy of data output</td>
</tr>
</tbody>
</table>
5. Procedure

Set-up truth 

Read directory block for satellite block number

Read satellite block for Control block addresses

If (Attribute blocks requested)
   Call LATTR to list all attribute blocks
FI

IF (EDR blocks requested)
   Call LEDR to list all EDR blocks
FI

IF (Library blocks requested)
   Call LLIB to list all Library blocks
FI

IF (Work blocks requested)
   Call LWRK to list all Work blocks
FI

If (Encyclopedia blocks requested)
   Call LENC to list all Encyclopedia blocks
FI

IF (CIT blocks requested)
   Call LCIT to list all CIT blocks
FI

Return
6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCALL(10)</td>
<td>L+1</td>
<td>logical array to call appropriate list all subroutines</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T = call</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F = do not call</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

None

6.3 Error Handling

If an I/O error occurs in one of the list all subroutines, LISTAL returns to its calling routine.

If an I/O error occurs while reading the MJS Log in Subroutine LISTAL, the following message is written along with DAIO error data passed in COMMON FERMSG. Then a XXXXXX return is taken to the xxxxxx calling subroutine.

LISTAL ------ I/O ERROR READING LOG.
1. Subroutine LCIT

2. Function

LCIT reads and displays the CITency Control block and its CIT ency
blocks for a given satellite.

3. Programmer and Date

Eunice Eng, January 1978

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for LCIT is as follows:

CALL LCIT(ICCITB,RTCODE)

The arguments in the calling sequence are described in the list below.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICCITB</td>
<td>I*4</td>
<td>I</td>
<td>Cit control block number</td>
</tr>
<tr>
<td>RTCODE</td>
<td>I*4</td>
<td>O</td>
<td>Return code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0, normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 8, I/O error encountered</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

Subroutine Description

LREAD deblocks and reads an MJS Log record

4.3 COMMON Variables

Variable Description

IMES(26) DA10 I/O error data

4.4 I/O Devices

Unit No. Description

06 terminal, output messages
10 sysout A, hardcopy output data
11 terminal or data set, output data
25 MJS Log
5. Procedure

Reset return code, RCODE = 0

Read and display CIT control block

Get address of the first CIT block

Write header for CIT blocks

DO until (all CIT blocks read)
    display contents of CIT blocks
OD

Return

6. Special Notes

6.1 Major Variables and Constants

None

6.2 Data structure and Tables

See Appendix for the data structure of the CIT control and CIT ency blocks.

6.3 Error Handling

If an I/O error occurs while reading the MJS Log, the following message is written along with the DAIO error data passed in COMMON FERMSG. The return code is set to 8 before exiting to the calling subroutine.

LCIT ***** I/O ERROR ENCOUNTERED.
1. Subroutine LATTR

2. Function

LATTR is a 'list all' subroutine that lists all the Attribute blocks for a given satellite.

3. Programmer and Date

Eunice Eng, January 1978

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for LATTR is as follows:

CALL LATTR(ICATTB,RTCODE)

The following list describes the arguments in the calling list.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICATTB</td>
<td>I*4</td>
<td>I</td>
<td>first Attribute block number</td>
</tr>
<tr>
<td>RTCODE</td>
<td>I*4</td>
<td>O</td>
<td>return code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0, normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 8, I/O error encountered</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

Subroutine  Description
LREAD       reads an MJS Log record and does its own deblocking

4.3 COMMON Variables

Variable  Type  COMMON Description
IMES(26)  I*4    FERMGSG DAIO I/O error data

4.4 I/O Device

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy of data output</td>
</tr>
<tr>
<td>11</td>
<td>terminal or data set, data output</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>
5. Procedure

Reset return code, RTCODE = 0

Write Headers

DO until (all Attribute blocks read)
    display contents of attribute block
OD

Return

6. Special Notes

6.1 Major Variables and Constants

None

6.2 Data Structure and Tables

See Appendix for description of Attribute blocks in an MJS Log.

6.3 Error Handling

If an I/O error is encountered while reading the MJS Log the following message is written along with DAIO data. The return code is set to 8 and a return taken.

I/O ERROR READING ATTRIBUTE BLOCK.
1. Subroutine LEDR

2. Function

LEDR reads and displays the EDR control block and its EDR blocks for a given satellite. Similar to LCIT.

3. - 6.2 See Subroutine LCIT

6.3 Error Handling

If an I/O error occurs while reading the MJS Log the following message is written along with DAIO I/O data from COMMON PERMSG. The return code is set to 8 and a return taken.

I/O ERROR DURING EDR LISTING.

1. Subroutine LENC

2. Function

LENC displays the Encyclopedia Control and associated Encyclopedia blocks for a given satellite. Similar to Subroutine LCIT.

3. - 6.3 See Subroutine LCIT

1. Subroutine LLIB

2. Function

LLIB displays the Library Control and associated Library blocks for a given satellite. Similar to Subroutine LCIT.

3. - 6.3 See Subroutine LCIT.

1. Subroutine LWRK

2. Function

LWRK displays the work Control and associated Work blocks for a given satellite. Similar to Subroutine LCIT.

3. - 6.3 see Subroutine LCIT.
1. Subroutine REMSLT

2. Function

REMSLT displays the remaining serials in the Library Control, the Work Control and the CIT ency Control blocks for a given satellite. REMSLT assumes the serials are used sequentially.

3. Programmer and Date

Eunice Eng, January 1978

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for REMSLT is as follows:

```
CALL REMSLT(QSATID)
```

The following describes the argument in the calling list.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID(12)</td>
<td>I*1</td>
<td>I</td>
<td>satellite ID</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LREAD</td>
<td>deblocks and reads a MJS Log record</td>
</tr>
<tr>
<td>DIRSAT</td>
<td>reads and returns the directory block.</td>
</tr>
<tr>
<td></td>
<td>Also checks the satellite id.</td>
</tr>
</tbody>
</table>

4.3 COMMON Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DAIO I/O error message</td>
</tr>
</tbody>
</table>

4.4 I/O Devices

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy, data output</td>
</tr>
<tr>
<td>11</td>
<td>terminal or data set, data output</td>
</tr>
<tr>
<td>25</td>
<td>MJS Log</td>
</tr>
</tbody>
</table>
5. Procedure

Read Directory for satellite block number
Identify satellite and write header
Read Satellite block
Read Library Control block
Display remaining slots
Read Work Control block
Display remaining slots
Read CIT Control block
Display remaining slots

Return

6. Special Notes

6.1 Major Variables and Constants

None

6.2 Data Structure and Tables

See Appendix for structure of Control blocks.

6.3 Error Handling

If satellite block not found, a return to the calling routine is taken.

If an I/O error is detected while reading the MJS Log the following message is written along with the DAIO I/O data passed through COMMON PERMSG. Then a return is taken.

I/O ERROR ON LOG.
1. Subroutine SHWCIT

2. Function

SHWCIT sets up to display CIT encyclopedia tapes marked for removal. Subroutine RWCIT does the actual listing.

3. Programmer and Date

Eunice Eng, January 1978

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for SHWCIT is as follows:

CALL SHWCIT(QSATID)

The following describes the argument in the calling sequence.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID(12)</td>
<td>I*1</td>
<td>I</td>
<td>satellite ID</td>
</tr>
</tbody>
</table>

4.2 Subroutines Called

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCLC</td>
<td>SACC - logical compare</td>
</tr>
<tr>
<td>LRREAD</td>
<td>deblocks and reads MJS Log record</td>
</tr>
<tr>
<td>CITRMV</td>
<td>displays the CIT tapes marked for removal</td>
</tr>
</tbody>
</table>

4.3 COMMON Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>COMMON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMES(26)</td>
<td>I*4</td>
<td>FERMSG</td>
<td>DATO I/O error data</td>
</tr>
</tbody>
</table>

4.4 I/O Device

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>terminal, output messages</td>
</tr>
<tr>
<td>10</td>
<td>sysout A, hardcopy of data output</td>
</tr>
</tbody>
</table>
5. Procedure

Identify the list for hardcopy

Read directory block for satellite block number
  If (satellite block not found)
  Write error message
  EXIT to RETURN

Read Satellite block for CIT Control block number

Call display program

RETURN

Return

6. Special Notes

6.1 Major Variables and Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICIT</td>
<td>I*4</td>
<td>Cit Control block number</td>
</tr>
<tr>
<td>ISATCD</td>
<td>I*4</td>
<td>satellite code passed to subroutine CITRMV to help build vol-ser</td>
</tr>
<tr>
<td>QSATCD(4)</td>
<td>L*1</td>
<td>equivalent to ISATCD</td>
</tr>
</tbody>
</table>

6.2 Data Structure and Tables

See Appendix for structure of an MJS Log directory and CIT Control block.

6.3 Error Handling

If the user input satellite id passed to SHWCIT is invalid, the following message is printed before a return is taken.

SATELLITE ID, QSATID = AAAAAAAAAAAAAA INVALID.

If an I/O error is encountered while reading the MJS Log, the following message is printed along with the DAIO error data passed through COMMON FERMSG. The subroutine then returns to the calling routine.

I/O ERROR ENCOUNTERED.
1. Subroutine SHWEDR

2. Function

SHWEDR sets-up to display EDR tapes marked for removal. Subroutine EDRRMV does the actual listing. The subroutine is similar to SHWCIT.

3. - 6.3 see Subroutine SHWCIT
User's Guide

1) Allocate 50 sequential tracks on a permanent user's disk.

2) Alter data common blocks if necessary.

3) Compile, link, load, and run INTLOG

a) Time set: 600 600
I. Function
ALTBLK is used to change byte data in the Voyager Automatic Log. This is useful in resetting process dispositions. ALTBLK backs up the Log at least once in the day it is used onto a day scratch disk. A hard copy is returned to the user for his records.

II. Restrictions
The user must be very familiar with the format of the Log. Only one or two authorized persons should be allowed to alter the Log. It is strongly suggested that the user dump the blocks of the Log he will be altering for sake of history keeping. Dumping the Log may be done via 'ZBEKE.LIB.CLIST(LISTER)' see userside for LISTER.

ALTBLK is expected to be run in large region on TSO foreground.

III. Syntax
A. To start
   `ex `zbeke.lib.clist(altblk)`'

B. To terminate session
   `/*` when prompted for a reason for altering the block.

C. User input
   `prompt WRITE REASON FOR ALTERING BLOCK (IN 80 CHAR.)`
   `user user text`
   `prompt INPUT BLOCK BYTE QVAL(HEX)`
   `user xxxxx xx zz`

IV. Error Conditions
None. Therefore, the user must be careful when typing in block numbers and byte numbers. The user may recover from typing in the wrong byte number if the block was dumped before altering.

CLIST Messages or Error Conditions
1. DATA SET NOT FOUND ON VOLUME AAAAAA. DATA SET IS BEING UNCATALOGED
   user response: Ignore. Means a back up of the Log has not been made yet for the day.

2. DO ENDED DUE TO ERROR+
   user response: type in ?.
   SYSTEM ABEND CODE 80A
   user response: Logon to larger region.

V. Load Module
   `ZBEKE.LIST.LOAD(ALTBLK)`

VI. TSO CLIST
   `ZBEKE.LIB.CLIST(ALTBLK)`

VII. Background JCL PROC
   None.
'ZBEKE. LIB.CLIST (ALTBLK)'

00010 PROC O
00020 SRCHDS 'ZBEKE.LOG.DATA'
00030 SYSRC (EQ 0) GO TO LABEL QCOPY
00040 ALLOC DA('ZBEKE.LOG.DATA') NEW TRACKS SPACE(50)
00050 GO TO LABEL COPY
00060 CC
00070 LABEL QCOPY
00080 ASK 'DO YOU WANT TO BACKUP SBMJS.LOG.DATA? (YES/NO)'
00090 SYSRC (EQ 4) GO TO LABEL FREE
00100 CC
00110 LABEL COPY
00120 CFREE F(SYSUT1, SYSUT2, SYSIN, SYSPRINT) A(LOG)
00130 ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F) BUFNO(1)
00140 ALLOC F(SYSUT1) DA('SBMJS.LOG.DATA') SHR USING(LOG)
00150 ALLOC F(SYSUT2) DA('ZBEKE.LOG.DATA') SHR USING(LOG)
00160 ALLOC F(SYSPRINT) DUMMY
00170 ALLOC F(SYSIN) DUMMY
00180 DO IEBGENER
00190 CFREE F(SYSUT1, SYSUT2, SYSIN, SYSPRINT)
00200 TPRINT 'SBMJS.LOG.DATA BACKED ONTO SCRATCH ZBEKE.LOG.DATA'
00210 CC
00220 LABEL FREE
00230 CFREE F(FT05F001, FT06F001, FT10F001, FT25F001)

0 CFREE ATTR(LOG)
00250 ALLOC F(FT05F001) DA(*)
00260 ALLOC F(FT06F001) DA(*)
00270 ALLOC F(FT10F001) SYSOUT
00280 ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F)
00290 ALLOC F(FT25F001) DA('SBMJS.LOG.DATA') OLD USING(LOG)
00300 CALL 'ZBEKE.LIST.LOAD(ALTBLK)'
00310 CFREE F(FT25F001) A(LOG)
00320 FREE F(FT10F001) SYSOUT(A)
00330 END
END OF DATA
LOGTLS - MJS LOG and TLS UPDATE PROGRAMS

There are four functions in the MJS Log and TLS Update system. At present they are expected to run on TSO real-time.

I. Function.

A. RMVEDR - remove EDR tapes marked for removal, one at a time from MJS Log and TLS.

B. RMVWRK - remove Work tapes with CIT and Merge processes marked done from TLS, one user specified serial at a time.

C. RMVCIT - remove CIT tapes, within a user given range from Log and TLS.

D. ASSIGN - assign Work or CIT tapes, up to a user specified maximum (LE 5) to fill unassigned slots allotted in TLS to the given MJS process.

E. Backup - all above functions give the user the option of backing up the MJS Log "SBMJS.LOG.DATA" onto a one day scratch data set "ZBEKE.LOG.DATA". If a backup does not exist in "ZBEKE.LOG.DATA", one will be done automatically once per day of use.

F. Hardcopy - all above functions also provide the user with a hard copy of his/her session.

II. Restrictions.

A. These programs must be run using user id = SBMJS, since these tapes are TLS userid protected.

B. The user must use TSO with the following region requirements:
   91 any region (only one region size at present)
   75 large region size(192), meaning 192k

C. One satellite is processed at a time, except for ASSIGN.

D. The user should know how to use command tlsupdte on TSO incase of fixups.

III. Syntax to invoke routines.

A. To run RMVEDR:
   ex 'ZBEKE.LIB.CLIST(RMVEDR)'

B. To run RMVWRK:
   ex 'ZBEKE.LIB.CLIST(RMVWRK)'

C. To run RMVCIT:
   ex 'ZBEKE.LIB.CLIST(RMVCIT)'

D. To run ASSIGN:
   ex 'ZBEKE.LIB.CLIST(ASSIGN)'

E. To close data sets incase of an 80A abend (see error)
   ex 'ZBEKE.LIB.CLIST(CLOGTLS)'

F. To list Log:
   ex 'ZBEKE.LIB.CLIST(LISTALL)'
   or background batch
stab YYY,t(h00h00)
job card
'zbeke.lib.cntl(listall)'
// exec notifyys
endinput

G. To invoke TLS update programs only:
tlsupdte

IV. Error Conditions.
A. General - If the execution terminates abnormally, i.e.
   system 80A, execute 'ZBEKE.LIB.CLIST(CLOGTLS)'
   to close data sets properly. Then list the
   Log using one of the above mentioned methods,
   to check the state of the Log. The user must
   then decide what kind of fixups are needed if
   any.

B. RMVEDR:
   1. XXXXXX NOT MARKED FOR REMOVAL.
      Response:
      a. check typing
      b. if vol-ser was entered twice, only the last
         entered will get marked. Must contact official
         person to alter the MJS Log manually to get to
         the first entered vol-ser.
      c. continue

   2. I/O ERROR ENCOUNTERED ON LOG.
      Response
      a. try listing Log
      b. save message and following lines of data to help
      c. continue

C. RMVWRK:
   1. SERIAL REQUESTED FOR REMOVAL XXX EXCEEDS LAST SERIAL
      ASSIGNED YYY.
      Response:
      a. check typing
      b. get latest Log listing
      c. continue

   2. XXX SERIAL NOT MARKED FOR REMOVAL.
      Response:
      a. check typing
      b. get latest Log listing
      c. continue

   3. XXX SERIAL NOT FOUND.
      Response:
      a. check typing
      b. continue

   4. I/O ERROR ENCOUNTERED ON LOG.
      Response: See RMVEDR above

D. RMVCLT:
   1. ISER1 = XXX IS GREATER THAN ISERN = YYY.
      Response:
a. Check typing. ISER1 = first serial, ISERN = last serial.
b. restart CLIST to continue

2. FIRST MUST BE ODD.
Response:
   a. restart CLIST to continue. This time
      first serial must be odd.

3. LAST MUST BE EVEN.
Response:
   a. restart CLIST to continue. This time
      last serial must be even.

4. ***** NONE OF THE SERIALS ARE ASSIGNED TO A TLS SLOT.
Response:
   a. check latest Log listing. No more removals
      can be done.

5. THE SERIALS IN THE REMOVE REQUEST IS OUT OF RANGE
   (XXX to YYY).
   XXX first log assigned serial
   YYY last log assigned serial
Response:
   a. check latest Log listing
   b. restart CLIST to continue

6. I/O ERROR ENCOUNTERED ON LOG.
Response:
   see RMVEDR above

E. ASSIGN:
1. MAX GREATER THAN 5.
Response:
   a. check typing
   b. restart CLIST to continue

2. INVALID QTYPE - AAAA - CHECK INPUT.
Response:
   a. check typing
   b. restart CLIST to continue

3. LAST SERIAL = XXX NOW EXCEEDS 999. UPDATING CONTROL
   AND TERMINATING.
Response:
   a. the last serial assigned was 999. Must
      allocate and initialize the next sequential
      Log.

4. I/O ERROR ENCOUNTERED ON LOG.
Response: See RMVEDR above.

V. Load Modules:
A. RMVEDR - 'ZBEKE.VOYAGER.LOAD(REDR)'
B. RMVWRK - 'ZBEKE.VOYAGER.LOAD(RWORK)'
C. RMVCIT - 'ZBEKE.VOYAGER.LOAD(RCIT)'
D. ASSIGN - 'ZBEKE.VOYAGER.LOAD(ASSIGN)'
E. LISTALL - 'ZBEKE.LIST.LOAD(LISTER)'

VI. TSO CLISTS:
A. RMVEDR - 'ZBEKE.LIB.CLIST(RMVEDR)'

Response: See RMVEDR above.
B. RMVWRK - 'ZBEKE.LIB.CLIST(RMVWRK)'
C. RMVCIT - 'ZBEKE.LIB.CLIST(RMVCIT)'
D. ASGN - 'ZBEKE.LIB.CLIST(ASGN)'
E. CLOGTLS - 'ZBEKE.LIB.CLIST(CLOGTLS)'
F. LISTALL - 'ZBEKE.LIB.CLIST(LISTALL)'

VII. Background PROC:
A. LISTALL - 'ZBEKE.LIB.CNTL(LISTALL)'

01510  01520  01530  01540  01550  01560  01570
RPlVEDR

00010 PROC 0
00020 TPRINT 'YOU SHOULD BE LOGGED ON UNDER ID SBMJS'
00030 TPRINT 'YOU SHOULD ALSO BE IN LARGE REGION.'
00040 SRC HD "ZBEKE.LOG.DATA"
00050 SYSRC (EQ 0) GO TO LABEL QCOPY
00060 ALLOC DA('ZBEKE.LOG.DATA') NEW TRACKS SPACE(50)
00070 CO TO LABEL COPY
00080 CC
00090 LABEL QCOPY
00100 ASK 'DO YOU WANT TO BACKUP SBMJS.LOG.DATA?'
00110 SYSRC (EQ 4) GO TO LABEL FREE
00120 CC
00130 LABEL COPY
00140 CFREE F(SYSUT1, SYSUT2, SYSIN, SYSPRINT) A(LOG)
00150 ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F) BUFNO(1)
00160 ALLOC F(SYSUT1) DA('SBMJS.LOG.DATA') SHR USING(LOG)
00170 ALLOC F(SYSUT2) DA('ZBEKE.LOG.DATA') SHR USING(LOG)
00180 ALLOC F(SYSPRINT) DUMMY
00190 ALLOC F(SYSIN) DUMMY
00200 DO IEGENER
00210 CFREE F(SYSUT1, SYSUT2, SYSIN, SYSPRINT)
00220 TPRINT 'SBMJS.LOG.DATA BACKED ONTO SCRATCH ZBEKE.LOG.DATA'
00230 CC
00240 LABEL FREE
00250 CFREE F(FT05F001, FT06F001, FT25F001) A(LOG)
00260 ALLOC F(FT05F001) DA(*)
00270 ALLOC F(FT06F001) DA(*)
00280 ALLOC F(FT25F001) SYSOUT
00290 ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F) BUFNO(1)
00300 ALLOC F(FT25F001) DA('SBMJS.LOG.DATA') USING(LOG) OLD
00310 CC
00320 CFREE F(SYSIN, SYSPRINT, VSN, PVTLIBDD, SLOT)
00330 CFREE A(PRINT, IN)
00340 ALLOC DA('ZBEKE.SYSIN.DATA') NEW TRACKS SPACE(2)
00350 ALLOC DA('ZBEKE.PRINT.DATA') NEW CYL SPACE(1,1)
00360 ATTR PRINT RECFM(V B) LRECL(137) BLKSIZE(7265) BUFNO(1)
00370 ATTR IN RECFM(V B) LRECL(80) BLKSIZE(80)
00380 ALLOC F(SYSIN) DA('ZBEKE.SYSIN.DATA') SHR USING(IN)
00390 ALLOC F(SYSPRINT) DA('ZBEKE.PRINT.DATA') SHR USING(PRINT)
00400 ALLOC F(VSN) DA('SYS2.TLS.VSN') SHR
00410 ALLOC F(PVTLIBDD) DA('SYS2.TLS.LOAD') SHR
00420 ALLOC F(SLOT) DA('SYS2.TLS.SLOT') SHR
00430 CC
00440 CALL 'ZBEKE.Voyager.LOAD(REDR)'
00450 CC
00460 CFREE F(FT10F001) SYSOUT(A)
00470 CFREE F(FT25F001)
00480 CFREE F(SYSIN, SYSPRINT, VSN, PVTLIBDD, SLOT)
00490 ALLOC F(SYSIN) DA(*)
00500 ALLOC F(SYSPRINT) DA(*)
00510 DELETE 'ZBEKE.PRINT.DATA'
00520 DELETE 'ZBEKE.SYSIN.DATA'
00530 END
00010 PROC 0
00020 TPRINT 'YOU SHOULD BE LOGGED ON UNDER ID SBMJS'
00030 TPRINT 'YOU SHOULD ALSO BE IN LARGE REGION.'
00040 SRCHDS 'ZBEKE.LOG.DATA'
00050 SYSRC (EQ 0) GO TO LABEL QCOPY
00060 ALLOC DA('ZBEKE-LOG-DATA') NEW TRACKS SPACE(50)
00070 GO TO LABEL COPY
00080 CC
00090 LABEL QCOPY
00100 ASK 'DO YOU WANT TO BACKUP SBMJS.LOG.DATA?'
00110 SYSRC (EQ 4) GO TO LABEL FREE
00120 CC
00130 LABEL COPY
00140 CFREE F(SYSUT1, SYSUT2, SYSIN, SYSPRINT) A(LOG)
00150 ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F) BUFNO(1)
00160 ALLOC F(SYSUT1) DA('SBMJS-LOG-DATA') SHR USING(LOG)
00170 ALLOC F(SYSUT2) DA('ZBEKE-LOG-DATA') SHR USING(LOG)
00180 ALLOC F(SYSPRINT) DUMMY
00190 ALLOC F(SYSIN) DUMMY
00200 DO IEBGENER
00210 CFREE F(SYSUT1, SYSUT2, SYSIN, SYSPRINT)
00220 TPRINT 'SBMJS.LOG-DATA BACKED ONTO SCRATCH ZBEKE-LOG-DATA'
00230 CC
00240 LABEL FREE
00250 CFREE F(FT05F001, FT06F001, FT10F001, FT25F001) A(LOG)
00260 ALLOC F(FT05F001) DA(*)
00270 ALLOC F(FT06F001) DA(*)
00280 ALLOC F(FT10F001) SYSPUT
00290 ATTR LOG BLKSIZE(7232) LRECL(64) RECFM(F B) BUFNO(1)
00300 ALLOC F(FT25F001) DA('SBMJS-LOG-DATA') USING(LOG) OLD
00310 CC
00320 CFREE F(SYSIN, SYSPRINT, VSN, PVTLIBDD, SLOT)
00330 CFREE A(PRINT, IN)
00340 ALLOC DA('ZBEKE-SYSIN.DAT') NEW TRACKS SPACE(2)
00350 ALLOC DA('ZBEKE-PRINT.DAT') NEW CYL SPACE(1,1)
00360 ALLOC F(PRINT) DA('ZBEKE-PRINT.DAT') SHR USING(PRINT)
00370 ATTR IN RECFM(F B) LRECL(137) BLKSIZE(7265) BUFNO(1)
00380 ATTR LOG BLKSIZE(7232) LRECL(64) RECFM(F B) BUFNO(1)
00390 ALLOC F(SYSIN) DA('ZBEKE-SYSIN.DAT') SHR USING(IN)
00400 ALLOC F(SYSPRINT) DA('ZBEKE-PRINT.DAT') SHR USING(PRINT)
00410 ALLOC F(VSN) DA('SYS2.TLS.VSN') SHR
00420 ALLOC F(PVTLIBDD) DA('SYS2.TLS.LOAD') SHR
00430 CC
00440 CALL 'ZBEKE.VOYAGER-LOAD(RWORK)'
00450 CC
00460 CFREE F(FT10F001) SYSPUT(A)
00470 CFREE F(FT25F001)
00480 CFREE F(SYSIN, SYSPRINT, VSN, PVTLIBDD, SLOT)
00490 ALLOC F(SYSIN) DA(*)
00500 ALLOC F(SYSPRINT) DA(*)
00510 DELETE 'ZBEKE-PRINT.DAT'
00520 DELETE 'ZBEKE-SYSIN.DAT'
00530 END
RMVCIT

0010 PROC O
0020 TPRINT 'YOU SHOULD BE LOGGED ON UNDER ID SBMJS'
0030 TPRINT 'YOU SHOULD ALSO BE IN LARGE REGION.'
0040 SRCHDS 'ZBEKE-LOG-DATA'
0050 SYSRC (EQ 0) GO TO LABEL QCOPY
0060 ALLOC DA('ZBEKE-LOG-DATA') NEW TRACKS SPACE(50)
0070 GO TO LABEL COPY
0080 CC
0090 LABEL QCOPY
0100 ASK 'DO YOU WANT TO BACKUP SBMJS-LOG-DATA?'
0110 SYSRC (EQ 4) GO TO LABEL FREE
0120 CC
0130 LABEL COPY
0140 CFREE F(SYSUT1,SYSUT2,SYSIN,SYSPRINT) A(LOG)
0150 ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F) BUFNO(1)
0160 ALLOC F(SYSUT1) DA('SBMJS-LOG-DATA') SHR USING(LOG)
0170 ALLOC F(SYSUT2) DA('ZBEKE-LOG-DATA') SHR USING(LOG)
0180 ALLOC F(SYSPRINT) DUMMY
0190 ALLOC F(SYSIN) DUMMY
0200 DO IEBGENER
0210 CFREE F(SYSUT1,SYSUT2,SYSIN,SYSPRINT)
0220 TPRINT 'SBMJS-LOG-DATA BACKED ONTO SCRATCH ZBEKE-LOG-DATA'
0230 CC
0240 LABEL FREE
0250 CFREE F(FT05F001,FT06F001,FT10F001,FT25F001) A(LOG)
0260 ALLOC F(FT05F001) DA(*)
0270 ALLOC F(FT06F001) DA(*)
0280 ALLOC F(FT10F001) SYSOUT
0290 ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F) BUFNO(1)
0300 ALLOC F(FT25F001) DA('SBMJS-LOG-DATA') USING(LOG) OLD
0310 CC
0320 CFREE F(SYSIN,SYSPRINT,VSN,PVTLIBDD,SLOT)
0330 CFREE A(PRINT,IN)
0340 ALLOC DA('ZBEKE-SYSIN-DATA') NEW TRACKS SPACE(2)
0350 ALLOC DA('ZBEKE-PRINT-DATA') NEW CYL SPACE(1,1)
0360 ATTR PRINT RECFM(V B) LRECL(137) BLKSIZE(7265) BUFNO(1)
0370 ATTR IN RECFM(F B) LRECL(80) BLKSIZE(80)
0380 ALLOC F(SYSIN) DA('ZBEKE-SYSIN-DATA') SHR USING(IN)
0390 ALLOC F(SYSPRINT) DA('ZBEKE-PRINT-DATA') SHR USING(PRINT)
0400 ALLOC F(VSN) DA('SYS2-TLS-VSN') SHR
0410 ALLOC F(PVTLIBDD) DA('SYS2-TLS-LOAD') SHR
0420 ALLOC F(SLOT) DA('SYS2-TLS-SLOT') SHR
0430 CC
0440 CALL 'ZBEKE-VOYAGER-LOAD(RCIT)'
0450 CC
0460 CFREE F(FT10F001) SYSOUT(A)
0470 CFREE F(FT25F001)
0480 CFREE F(SYSIN,SYSPRINT,VSN,PVTLIBDD,SLOT)
0490 ALLOC F(SYSIN) DA(*)
0500 ALLOC F(SYSPRINT) DA(*)
0510 DELETE 'ZBEKE-PRINT-DATA'
0520 DELETE 'ZBEKE-SYSIN-DATA'
0530 END
LOGASN

00010 PROC 0
00020 TPRINT "YOU SHOULD BE LOGGED ON UNDER ID SBMJS"  
00030 TPRINT "YOU SHOULD ALSO BE IN LARGE REGION."
00040 SRCHDS 'ZBEKE.LOG.DAT'  
00050 SYSRC (EQ 0) GO TO LABEL QCOPY
00060 ALLOC DA('ZBEKE.LOG.DAT') NEW TRACKS SPACE(50)  
00070 GO TO LABEL COPY
00080 CC  
00090 LABEL QCOPY
00100 ASK 'DO YOU WANT TO BACKUP SBMJS-LOG-DATA?'
00110 SYSRC (EQ 4) GO TO LABEL FREE
00120 CC
00130 LABEL COPY
00140 CFREE F(SYSUT1,SYSUT2,SYSLN,SYSPRINT) A(LOG)  
00150 ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F) BUFNO(1)
00160 ALLOC F(SYSUT1) DA('SBMJS.LOG.DAT') SHR USING(LOG)  
00170 ALLOC F(SYSUT2) DA('ZBEKE.LOG.DAT') SHR USING(LOG)  
00180 ALLOC F(SYSPRINT) DUMMY
00190 ALLOC F(SYSLN) DUMMY
00200 DO IEBUGENER  
00210 CFREE F(SYSUT1,SYSLN,SYSPRINT)
00220 TPRINT 'SBMJS-LOG-DATA BACKED ONTO SCRATCH ZBEKE-LOG-DATA'
00230 CC
00240 LABEL FREE
00250 CFREE F(FT05F001,FT06F001,FT10F001,FT25F001) A(LOG)  
00260 ALLOC F(FT05F001) DA(*)  
00270 ALLOC F(FT06F001) DA(*)  
00280 ALLOC F(FT10F001) SYSOUT
00290 ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F) BUFNO(1)  
00300 ALLOC F(FT25F001) DA('SBMJS.LOG.DAT') USING(LOG) OLD  
00310 CC
00320 CFREE F(SYSLN,SYSPRINT,VSN,PVTLIBDD,SLOT)  
00330 CFREE A(PRINT,IN)
00340 ALLOC DA('ZBEKE-SYSIN-DATA') NEW TRACKS SPACE(2)  
00350 ALLOC DA('ZBEKE-PRINT-DATA') NEW CYL SPACE(1,1)  
00360 ATTR PRINT RECFM(VB) LRECL(137) BLKSIZE(7265) BUFNO(1)  
00370 ATTR IN REC FM(FB) LRECL(80) BLKSIZE(80)  
00380 ALLOC F(SYSLN) DA('ZBEKE-SYSIN-DATA') SHR USING(IN)  
00390 ALLOC F(SYSPRINT) DA('ZBEKE-PRINT-DATA') SHR USING(PRINT)  
00400 ALLOC F(VSN) DA('SYS2.TLS.VSN') SHR
00410 ALLOC F(PVTLIBDD) DA('SYS2.TLS.LOAD') SHR
00420 ALLOC F(SLOT) DA('SYS2.TLS.SLOT') SHR
00430 CC
00440 LABEL ASSIGN  
00450 CALL 'ZBEKE.VOYAGER.LOAD(ASSIGN)'
00460 CC
00470 ASK 'DO YOU WANT TO ASSIGN MORE TAPES? (YES OR NO)'
00480 SYSRC (EQ 0) GO TO LABEL ASSIGN
00490 CC  
00500 CFREE F(FT10F001) SYSOUT(A)  
00510 CFREE F(FT25F001)
00520 CFREE F(SYSLN,SYSPRINT,VSN,PVTLIBDD,SLOT)  
00530 ALLOC F(SYSLN) DA(*)  
00540 ALLOC F(SYSPRINT) DA(*)  
00550 DELETE 'ZBEKE-PRINT-DATA'
00560 DELETE 'ZBEKE-SYSIN-DATA'
00570 END
CLOGTLS

00010 PROC 0
00020 CFREE F(FT10F001) SYSOUT(A)
00030 CFREE F(FT25F001)
00040 CFREE F(SYSIN, SYSPRINT, VSN, PVTLIBDD, SLOT)
00050 ALLOC F(SYSIN) DA(*)
00060 ALLOC F(SYSPRINT) DA(*)
00070 DELETE 'ZBEKE.PRINT.DATA'
00080 DELETE 'ZBEKE.SYSIN.DATA'
00090 END
LISTALL

00010 PROC 0
00020 CFREE F(FT05F001,FT06F001,FT10F001,FT11F001,FT25F001)
00030 CFREE ATTR(LOG)
00040 ATTR LOG BLKSIZE(7232) LRECL(64) RECFM(F B)
00050 ALLOC F(FT05F001) DA('SBMJS-LISTER-DATA')
00060 ALLOC F(FT06F001) DUMMY
00070 ALLOC F(FT10F001) SYSOUT
00080 ALLOC F(FT11F001) DUMMY
00090 ALLOC F(FT25F001) DA('SBMJS-LOG-DATA') OLD USING(LOG)
00100 CALL 'ZBEKE-LIST-LOAD(LISTER)'
00110 FREE F(FT10F001) SYSOUT(A)
00120 FREE F(FT05F001,FT06F001 FT11F001,FT25F001)
00130 ALLOC F(FT05F001) DA(*)
00140 ALLOC F(FT06F001) DA(*)
00150 END
LISTER

Lister describes the list routines of the Voyager Automatic Log.

00010    I. Function:
00040        List all or parts of the Voyager Automatic Log.
00050        A hard copy of the TSO session is returned to the user.
00070    II. Restrictions:
00080        It is highly recommended that the user submit a background job when requesting a list all type list of the entire Voyager Log. (Use 'ZBEKE.LIB.CNTL.(LISTAL)')
00110
00120    III. Syntax:
00130        To start a TSO, selective list of the Log.
00140        ex 'zbeke.lib.clist(lister)' 'log(‘sbmjs.log.data’) -
00160        out11(*)'
00170          A. log = Voyager Automatic Log. Default = ‘sbmjs.log.data’
00180          B. out11 = data set where immediate listing should be
00190          sent. Default = *, user terminal. However, if the
00200          list you expect, is long, use a temporary data set
00210          i.e. ‘zbeke.list.data’ which the CLIST will allocate
00220          60 tracks on a scratch disk. After the session,
00230          the user can QED the data set and selectively search
00240          for the lines he wants.
00250          qed 'zbeke.list.data' nonum
00260          To terminate session type in
00270          /*
00280          when prompted for another menu number.
00290          To submit a background batch job from TSO, the
00300          following job stream is necessary
00310          job card
00320          ='zbeke.lib.cntl(listall)'
00330          // exec notifyts,userid=AAAAA,mode=all
00340          endinput
00350
00360    IV. Error Conditions:
00370        The programs will prompt the user for information. The errors
00380        are listed below by the type of listings is requested.
00390        A. Main Program - CALLIST
00400            1. WRONG MENU NUMBER.
00410            Check typing. Program will prompt for next menu
00420            number.
00430            2. If the OUT11 data set is exceeded, TSO will send a
00440            D-37 message to the user. Alter the CLIST to alloc.
00450            more space when allocating OUT11.
00460        B. BAKLOG
1. NO EDR BLOCKS FOR AAAAAAAAAAAAA IN LOG AT PRESENT.
2. NO LIBRARY BLOCKS FOR AAAAAAAAAAAAA IN LOG AT PRESENT.
   where AAAAAAAAAAAAA = satellite id
3. NO WORK BLOCKS FOR AAAAAAAAAAAAA IN LOG AT PRESENT.

where AAAAAAAAAAAAA = satellite id

Check typing. Program will prompt for next menu number.

2. SATELLITE ID, QSATID = AAAAAAAAAAAAA IS INVALID.
   Check typing. Program will prompt for next menu number

3. I/O ERROR ON LOG.
   Check Log. Program will terminate.

C. DMPLOG

1. I/O ERROR ON LOG.
   see BAKLOG

D. SHWCIT, display CIT tapes marked for removal.

1. I/O ERROR ON LOG.
   see BAKLOG

2. SATELLITE ID, QSATID = AAAAAAAAAAAAA IS INVALID.
   see BAKLOG

E. SHWEDR, display EDR tapes marked for removal.

1. I/O ERROR ON LOG.
   see BAKLOG

2. SATELLITE ID, QSATID = AAAAAAAAAAAAA IS INVALID.
   see BAKLOG

F. INCOMP
   similar to BAKLOG, see BAKLOG.

G. LISTAL
   1. I/O error on LISTAL or any of its Subroutines
      see BAKLOG

H. REMSLT
   1. I/O ERROR ON LOG
      see BAKLOG

2. SATELLITE ID, QSATID = AAAAAAAAAAAAA IS INVALID.
   see BAKLOG

00810 * V. Load Modules

00820 A. 'ZBEKE.LIST.LOAD'

00840 *

00850 VI. TSO CLIST

00860 A. 'ZBEKE.LIB.CLIST(LISTER)'

00870 *

00880 VII. Background JCL PROC

00890 A. 'ZBEKE.LIB.CNTL(LISTALL)'

END OF DATA
00010 PROC O LOG('SBMJS.LOG.DATA') OUT11(*)
00020 CFREE F(FT05F001,FT06F001,FT10F001,FT11F001,FT25F001)
00030 CFREE ATTR(OUT,LOG)
00040 ATTR OUT BLKSIZE(2400) LRECL(120) RECFM(F B)
00050 ATTR LOG BLKSIZE(7232) LRECL(64) RECFM(F B)
00060 ALLOC F(FT05F001) DA(*)
00070 ALLOC F(FT06F001) DA(*)
00080 ALLOC F(FT10F001) SYSOUT
00090 ASK 'DID YOU NAME A DSN FOR OUT11? (YES/NO)'
00100 SYSRC (EQ 4) GO TO LABEL TERM
00110 SRCHDS &OUT11
00120 WHEN SYSRC(LE 4)
00130 ALLOC DA(&OUT11.) NEW SPACE(60) TRACKS
00140 ALLOC F(FT11F001) DA(&OUT11.) USING(OUT) MOD
00150 GO TO LABEL FT25
00160 LABEL TERM
00170 ALLOC F(FT11F001) DA(&OUT11.)
00180 LABEL FT25
00190 ALLOC F(FT25F001) DA(&LOG,) USING(LOG) OLD
00200 ASK 'DO YOU WANT TO SEE THE LIST MENU? (YES/NO)'
00210 SYSRC (EQ 4) GO TO LABEL CALL
00220 TPRINT 'MENU NO.  NAME             DESCRIPTION''
00230 TPRINT '  01  SHWEGR  DISPLAYS EDR TAPES MARKED FOR REMOVAL''
00240 TPRINT '  02  SHWCDT  DISPLAYS CIT TAPES MARKED FOR REMOVAL''
00250 TPRINT '  03  REMSLT  # TAPE SERIALS REMAINING IN SOME CONTROLS''
00260 TPRINT '  04  BAKLOG  BACK LOG ON A GIVEN PROCESS''
00261 TPRINT '  05  INCMPL  INCOMPLETE PROCESSES''
00262 TPRINT '  06  LISTAL  LIST ALL BY BLOCKS''
00263 TPRINT '  07  DMPLOG  HEX DUMP OF THE LOG''
00270 LABEL CALL
00280 CALL 'ZBEKE.LIST.LOAD(LISTER)'
00290 FREE F(FT10F001) SYSOUT(A)
00300 TPRINT 'OUPUT TO FT11F001 ON &OUT11.'
00310 FREE F(FT25F001)
00320 END OF DATA
The following is a CLIST (of TSO) used to invoke the LIST program. The program is interactive with the user at the terminal.

CLIST location: EBEEKF.LIB.CLIST (LISTER)

PROC C LOG('EBMEJS.LOG;DATA*) QLT11(*)
CFREE F(FT15F001, FT05F001, FT10F001, FT11F001, FT25F001)
CFREE ATTR(OUT, LOG)
ATTR OUT ELKSIZE(24CC) LRECL(120) RECFM(F B)
ATTR LOC ELKSIZE(2322) LRECL(2322) RECFM(F)
ALLOC F(FT15F001) CA(*)
ALLOC F(FT05F001) CA(*)
ALLOC F(FT10F001) SYSCUT
ASK 'DO YOU NAME A DSN FOR OUT11?'
SYSFC (EQ 4) GO TO LABEL TERM
SRCHCS EOUT11
WHEN SYSCU(LE 4)
ALLOC CA(EOUT11*) NEW SPACE(60) TRACKS
ALLOC F(FT11F001) CA(EOUT11*) USING(CLT) MC
GC TC LABEL FT25
LABEL TERM
ALLOC F(FT11F001) CA(EOUT11*)
LABEL FT25
ALLOC F(FT25F001) CA(ELOG*) USING(LOG) CLD
ASK 'DO YOU WANT TO SEE THE LIST MENU?'
SYSFC (EQ 4) GO TO LABEL CALL
TFPRINT 'MENU NO., NAME DESCRIPTION'
TFPRINT ' 01 SHWEDR DISPLAYS EDT TAPE MARKED FOR REMOVAL'
TFPRINT ' 02 SHWCIT DISPLAYS CTT TAPE MARKED FOR REMOVAL'
TFPRINT ' 03 REMSLT # TAPE SERIALS REMAINING IN SOME CONTROL'
TFPRINT ' 04 BAKLOG BACK LOG ON A GIVEN PROCESS'
TFPRINT ' 05 INCOMP INCOMPLETE PROCESSES'
TFPRINT ' 06 LISTAL LIST ALL BY BLOCKS'
TFPRINT ' 07 CMPLST HE4 DUMP OF THE LOG'
LABEL CALL
CALL 'EBEELISTLOAD(LIST)
FREE F(FT11F001) SYSCUT(A)
TFPRINT 'CUPUT TO FT11F001 ON EOUT11*
FREE F(FT25F001)

EX 'EBEELIB.CLIST(LISTER)' LOG("____.LOG.____") COUTI("LIST.DAT")

Note: LOG(") gives the name of the LOG to be created. The default is set to 'EDITMS.LOG.DATA'

COUTI("") gives the name of a data output, data set. This is to be used if large output is expected. The default is set to the terminal.
<table>
<thead>
<tr>
<th>Logical Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Terminal, data input</td>
</tr>
<tr>
<td>6</td>
<td>Terminal, printer and error messages</td>
</tr>
<tr>
<td>10</td>
<td>Sysout(a), hard copy of session</td>
</tr>
<tr>
<td>11</td>
<td>Terminal or data set, data output</td>
</tr>
<tr>
<td>25</td>
<td>Log</td>
</tr>
</tbody>
</table>

Load Module Location:
00000 // THISDATE LOG LIST
00020 // LISTALL PROC
00030 // EXEC PGM=LISTER,REGION=150K
00040 // STEPLIB DD DSN=ZBEKE.LIST,LOAD,DISP=SHR
00050 // FT05F001 DD DDNAME=DATA5
00060 // FT06F001 DD DUMMY
00070 // FT10F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265)
00080 // FT11F001 DD DUMMY
00090 // FT25F001 DD DSN=SBJMJS.LOG,DATA,UNIT=2314,DISP=SHR,
00100 // DCB=(LRECL=64,RECFM=FB,BLKSIZE=7232)
00110 // PEND
00120 // EXEC LISTALL
00130 // DATA5 DD *
00140 06
00150 VOYAGER-1
00160 NO
00170 01
00180 06
00190 VOYAGER-2
00200 NO
00210 01
00220 /*
END OF DATA
<table>
<thead>
<tr>
<th>BLOCKS</th>
<th>CSECT</th>
<th>BLOCK 1</th>
<th>DIRECTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC F'0'</td>
<td>DC H'5650'</td>
<td>DC H'15'</td>
<td>DC H'2'</td>
</tr>
<tr>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC X'01'</td>
</tr>
<tr>
<td>DC X'09'</td>
<td>DC X'09'</td>
<td>DC X'09'</td>
<td>DC X'09'</td>
</tr>
<tr>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC H'2'</td>
</tr>
<tr>
<td>DC X'09'</td>
<td>DC X'09'</td>
<td>DC X'09'</td>
<td>DC X'09'</td>
</tr>
<tr>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC H'2'</td>
</tr>
<tr>
<td>DC X'09'</td>
<td>DC X'09'</td>
<td>DC X'09'</td>
<td>DC X'09'</td>
</tr>
<tr>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC H'2'</td>
</tr>
<tr>
<td>DC X'09'</td>
<td>DC X'09'</td>
<td>DC X'09'</td>
<td>DC X'09'</td>
</tr>
<tr>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC H'2'</td>
<td>DC H'2'</td>
</tr>
</tbody>
</table>

* BLOCKS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
* BLOGS IS A BLOCK DATA USED TO INITIATE THE LOG
*------------------------------------------------------------------
<table>
<thead>
<tr>
<th>Block 2</th>
<th>Satellite -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>X'4C1'</td>
</tr>
<tr>
<td>DC</td>
<td>7X1L'00'</td>
</tr>
<tr>
<td>DC</td>
<td>H'61'</td>
</tr>
<tr>
<td>DC</td>
<td>H'62'</td>
</tr>
<tr>
<td>DC</td>
<td>H'63'</td>
</tr>
<tr>
<td>DC</td>
<td>H'64'</td>
</tr>
<tr>
<td>DC</td>
<td>111'0'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 3</th>
<th>Ency Attn -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>X'02'</td>
</tr>
<tr>
<td>DC</td>
<td>7X1L'00'</td>
</tr>
<tr>
<td>DC</td>
<td>H'900'</td>
</tr>
<tr>
<td>DC</td>
<td>X'00'</td>
</tr>
<tr>
<td>DC</td>
<td>H'FFFFF0'</td>
</tr>
<tr>
<td>DC</td>
<td>H'61'</td>
</tr>
<tr>
<td>DC</td>
<td>H'62'</td>
</tr>
<tr>
<td>DC</td>
<td>H'63'</td>
</tr>
<tr>
<td>DC</td>
<td>111'0'</td>
</tr>
<tr>
<td>DC</td>
<td>4F'0'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 4</th>
<th>EDP Control 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>A'03'</td>
</tr>
<tr>
<td>DC</td>
<td>7X1L'00'</td>
</tr>
<tr>
<td>DC</td>
<td>H'61'</td>
</tr>
<tr>
<td>DC</td>
<td>H'62'</td>
</tr>
<tr>
<td>DC</td>
<td>H'63'</td>
</tr>
<tr>
<td>DC</td>
<td>P'64590'</td>
</tr>
<tr>
<td>DC</td>
<td>P'64614'</td>
</tr>
<tr>
<td>DC</td>
<td>111'0'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 5</th>
<th>Library Control 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>X'04'</td>
</tr>
<tr>
<td>DC</td>
<td>7X1L'00'</td>
</tr>
<tr>
<td>DC</td>
<td>H'61'</td>
</tr>
<tr>
<td>DC</td>
<td>H'62'</td>
</tr>
<tr>
<td>DC</td>
<td>H'63'</td>
</tr>
<tr>
<td>DC</td>
<td>H'64'</td>
</tr>
<tr>
<td>DC</td>
<td>P'60640'</td>
</tr>
<tr>
<td>DC</td>
<td>L'0689'</td>
</tr>
<tr>
<td>DC</td>
<td>H'800'</td>
</tr>
<tr>
<td>DC</td>
<td>T'1811'</td>
</tr>
<tr>
<td>DC</td>
<td>CL'61030'</td>
</tr>
<tr>
<td>DC</td>
<td>F'61'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 6</th>
<th>Work Control 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>X'05'</td>
</tr>
<tr>
<td>DC</td>
<td>7X1L'00'</td>
</tr>
<tr>
<td>DC</td>
<td>H'61'</td>
</tr>
<tr>
<td>DC</td>
<td>H'62'</td>
</tr>
<tr>
<td>DC</td>
<td>P'64410'</td>
</tr>
<tr>
<td>DC</td>
<td>P'64343'</td>
</tr>
<tr>
<td>DC</td>
<td>CL'641960'</td>
</tr>
<tr>
<td>DC</td>
<td>H'63'</td>
</tr>
<tr>
<td>DC</td>
<td>7H'0'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 7</th>
<th>Ency Control 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>X'06'</td>
</tr>
<tr>
<td>DC</td>
<td>7X1L'00'</td>
</tr>
<tr>
<td>DC</td>
<td>H'61'</td>
</tr>
<tr>
<td>DC</td>
<td>H'62'</td>
</tr>
<tr>
<td>DC</td>
<td>H'63'</td>
</tr>
<tr>
<td>DC</td>
<td>111'0'</td>
</tr>
</tbody>
</table>

25 MAY 78 11.36.06 - VOL=DBB34K, DSN=Z3EKVJSLOG.ASM
DC H'65'    SPARE  00023
DC H'50'    1ST LIBRARY SERIAL  00023
DC H'80'    LAST LIBRARY SERIAL  00023
DC E'6040'  1ST LIBRARY SLOT  00023
DC E'6050'  LAST LIBRARY SLOT  00023
DC H'10'    LAST VOL-SEZ WRITTEN ON  00023
DC H'00'    LAST DATA FILE SEQUENCE  00023
DC H'7800'  MAX. AMT. OF FEET TO BE USED/TAPE  00024
DC CL6'M2800'  MODEL NO. FOR LIBRARY TAPES  00024
DC P'0'  MODEL VOL-SEZ  00024

*** BLOCK 13 ***** WORK CONTROL 2 *****

DC A'05'    BLOCK IN CODE  00024
DC H'00'    2 OF FIRST WORK BLOCK  00024
DC H'01'    2 OF LAST WORK BLOCK  00024
DC H'10'    1ST WORK SERIAL  00025
DC H'999'    LAST WORK SERIAL  00025
DC E'60435'  1ST WORK SLOT  00025
DC E'60459'  LAST WORK SLOT  00025
DC CL6'M2800'  WORK ENCY  00025
DC CL6'M2800'  WORK ENCY  00025
DC H'01'    3 OF FREE WORK VOLUME  00025
DC 7H'00'  00026

*** BLOCK 14 ***** ENCY CONTROL 2 *****

DC X'06'    BLOCK IN CODE  00026
DC H'00'    2 OF 1ST ENCY BLOCK  00026
DC H'01'    2 OF LAST ENCY BLOCK  00026
DC H'11'    1ST ENCY SERIAL  00026
DC H'999'    LAST ENCY SERIAL  00026
DC E'60439'  1ST ENCY SLOT  00027
DC E'60459'  LAST ENCY SLOT  00027
DC CL6'M2800'  MODEL VOL-SEZ  00027
DC H'00'    3 OF FREE ENCY VOLUME  00027
DC H'2000'  MAX. AMT. OF FEET TO BE USED/TAPE  00027
DC H'01'    00027
DC 7F'0'  00027

*** BLOCK 15 ***** CITENCY CONTROL 2 *****

DC X'07'    BLOCK IN CODE  00027
DC H'00'    2 OF FIRST CITENCY BLOCK  00027
DC H'01'    2 OF LAST CITENCY BLOCK  00027
DC H'11'    1ST CITENCY SERIAL  00028
DC H'999'    LAST CITENCY SERIAL  00028
DC E'60435'  1ST CITENCY SLOT  00028
DC E'60459'  LAST CITENCY SLOT  00028
DC CL6'M2800'  VOL-SEZ MODEL  00028
DC H'00'    3 OF FREE CITENCY VOLUME  00028
DC H'2000'  MAX. AMT. OF FEET/TAPE  00028
DC H'01'    00028
DC 3F'0'  00028

*** END OF "MEMBER" *** 294 RECORDS PROCESSED  00028
<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>Type</th>
<th>Length</th>
<th>Disp</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSATID (12)</td>
<td>L=1</td>
<td>12</td>
<td>0</td>
<td>Satellite ID, (alphanumeric)</td>
</tr>
<tr>
<td>ISATCD</td>
<td>I=4</td>
<td>4</td>
<td>12</td>
<td>Satellite code</td>
</tr>
<tr>
<td>EDORER (6)</td>
<td>L=1</td>
<td>6</td>
<td>16</td>
<td>EDR vol-ser + XXXXX, JPL number</td>
</tr>
<tr>
<td>EDRDSN (6)</td>
<td>L=1</td>
<td>6</td>
<td>22</td>
<td>EDR DSN = CRS</td>
</tr>
<tr>
<td>IEDORSL</td>
<td>I=4</td>
<td>4</td>
<td>23</td>
<td>EDR slot</td>
</tr>
<tr>
<td>NR2EDR</td>
<td>I=4</td>
<td>4</td>
<td>32</td>
<td>No. EDR records</td>
</tr>
<tr>
<td>NR6EDR</td>
<td>I=4</td>
<td>4</td>
<td>36</td>
<td>No. EDR errors</td>
</tr>
<tr>
<td>IEDR8K</td>
<td>I=4</td>
<td>4</td>
<td>40</td>
<td>EDR block address</td>
</tr>
<tr>
<td>HEDRCH</td>
<td>I=2</td>
<td>2</td>
<td>44</td>
<td>2 char. of EDR, JPL-ser</td>
</tr>
<tr>
<td>HLIBSER</td>
<td>I=2</td>
<td>2</td>
<td>46</td>
<td>Lib. vol-ser (binary)</td>
</tr>
<tr>
<td>DISPDS</td>
<td>R=8</td>
<td>8</td>
<td>48</td>
<td>Start FOSC</td>
</tr>
<tr>
<td>DGENDS</td>
<td>R=8</td>
<td>8</td>
<td>56</td>
<td>End FOSC</td>
</tr>
<tr>
<td>LIBSER (6)</td>
<td>L=1</td>
<td>6</td>
<td>64</td>
<td>Lib. vol-ser, EBCD(64) = M(1) M(2) M(3) M(4) M(5) M(6) M(7)</td>
</tr>
<tr>
<td>LIBDSN (18)</td>
<td>L=1</td>
<td>18</td>
<td>70</td>
<td>Lib. DSN, EBCD(18) = V(8), XXXX XXXX XXXX XXXX XXXX XXXX XXXX</td>
</tr>
<tr>
<td>TISLOT</td>
<td>I=4</td>
<td>4</td>
<td>88</td>
<td>Lib. 1st slot</td>
</tr>
<tr>
<td>INSLOT</td>
<td>I=4</td>
<td>4</td>
<td>92</td>
<td>Lib. Last slot</td>
</tr>
<tr>
<td>HLIB8K</td>
<td>I=4</td>
<td>4</td>
<td>96</td>
<td>Lib. block address</td>
</tr>
<tr>
<td>HLIBSEQ</td>
<td>I=2</td>
<td>2</td>
<td>100</td>
<td>Lib. tape file sequence</td>
</tr>
<tr>
<td>HFEE7</td>
<td>I=2</td>
<td>2</td>
<td>102</td>
<td>Amt. of feet used on present tape</td>
</tr>
<tr>
<td>HMAXFT</td>
<td>I=2</td>
<td>2</td>
<td>104</td>
<td>Max. feet on tape</td>
</tr>
<tr>
<td>HWORKER</td>
<td>I=2</td>
<td>2</td>
<td>106</td>
<td>Work VOL-SER (binary)</td>
</tr>
<tr>
<td>IATTBK</td>
<td>I=4</td>
<td>4</td>
<td>108</td>
<td>ENCY-BTR block</td>
</tr>
<tr>
<td>HWORK8K</td>
<td>I=4</td>
<td>4</td>
<td>112</td>
<td>WORK block address</td>
</tr>
<tr>
<td>WROKE (18)</td>
<td>L=1</td>
<td>18</td>
<td>136</td>
<td>WORK DSN, EBCD(18) = V(8)</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Length</td>
<td>Disp</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>--------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>I1WSLT</td>
<td>I*4</td>
<td>4</td>
<td>140</td>
<td></td>
</tr>
<tr>
<td>INWSLT</td>
<td>I*4</td>
<td>4</td>
<td>144</td>
<td></td>
</tr>
<tr>
<td>I1RVOL</td>
<td>I*4</td>
<td>4</td>
<td>148</td>
<td></td>
</tr>
<tr>
<td>INRVOL</td>
<td>I*4</td>
<td>4</td>
<td>152</td>
<td></td>
</tr>
</tbody>
</table>

1st WORK SLOT

Last WORK SLOT

1st record volume number on work block

Last record volume number on work block
The log is basically of two parts, the control blocks and the blocks themselves. Each block is build a forward and backward link.

**THE CONTROL BLOCKS:**

The directory block states what satellites are included in the log, where the satellite directories are and the current size of the log.

The satellite block is a directory of the first control block for a given satellite.

The attribute block describes the conditions under which the ENCGEN procedure was run.

The EDC, Library, Work, Ency and CTE Ency, Control blocks basically keep track of the first and last assigned blocks and range of tape data and serials.

In the case of EDC Library, Work, Ency and CTE Ency Tape limits are also described.
The Blocks

The EOB, Library, Work, Entry, and
Exit blocks maintain forward and
backward links, processing status, and
define the date covered and their
tape location.

The following tables will
give a more detailed description of the
fields inside a log block.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Directory Block</td>
</tr>
<tr>
<td>01</td>
<td>Satellite block</td>
</tr>
<tr>
<td>02</td>
<td>Ency-Attribute block</td>
</tr>
<tr>
<td>03</td>
<td>EDR Control block</td>
</tr>
<tr>
<td>04</td>
<td>Library Control block</td>
</tr>
<tr>
<td>05</td>
<td>Work Control block</td>
</tr>
<tr>
<td>06</td>
<td>Encyclopedia Control Block</td>
</tr>
<tr>
<td>07</td>
<td>CITENCY Control block</td>
</tr>
<tr>
<td>13</td>
<td>EDR block</td>
</tr>
<tr>
<td>14</td>
<td>Library block</td>
</tr>
<tr>
<td>15</td>
<td>Work block</td>
</tr>
<tr>
<td>16</td>
<td>Encyclopedia block</td>
</tr>
<tr>
<td>17</td>
<td>CITENCY block</td>
</tr>
</tbody>
</table>
Structure of Processing Disposition Byte

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>marked for processing</td>
</tr>
<tr>
<td>1</td>
<td>processing has begun</td>
</tr>
<tr>
<td>2</td>
<td>processing was completed</td>
</tr>
<tr>
<td>3</td>
<td>spare</td>
</tr>
<tr>
<td>4-7</td>
<td>processing completion code</td>
</tr>
<tr>
<td></td>
<td>=0 no errors</td>
</tr>
<tr>
<td></td>
<td>=1 errors detected</td>
</tr>
<tr>
<td></td>
<td>=15 serious error</td>
</tr>
</tbody>
</table>

Structure of Slot Allocation Status Byte

<table>
<thead>
<tr>
<th>Byte</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>slot allocation requested</td>
</tr>
<tr>
<td>1</td>
<td>slot allocated</td>
</tr>
<tr>
<td>2</td>
<td>slot is removable</td>
</tr>
<tr>
<td>3-7</td>
<td>spare</td>
</tr>
</tbody>
</table>

\[ CO \] \[ EO \] \[ FC \]
<table>
<thead>
<tr>
<th>Byte</th>
<th>Byte Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type, = 0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type, = 0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Number of blocks in data set</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Address of last block allocated</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Number of satellite entries</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>Maximum number of satellite entries, = 3</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>S/C id. code</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>Number of characters in satellite id.</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Satellite id. (alphanumeric)</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>Address of Satellite block for this satellite</td>
</tr>
<tr>
<td>32-47</td>
<td>16</td>
<td>Same as 16-31 above for second satellite</td>
</tr>
<tr>
<td>48-63</td>
<td>16</td>
<td>Same as 16-31 above for third satellite</td>
</tr>
<tr>
<td>Byte</td>
<td>Byte Length</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Block identifier code, = 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type, = 0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type, = 0</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Address of first Ency-Attribute block</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Address of EDR Control block</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Address of Library Control block</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>Address of Work Control block</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>Address of Encyclopedia Control block</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>Address of CITENCY Control block</td>
</tr>
<tr>
<td>Byte</td>
<td>Byte Length</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Block identifier code, = 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Record volume length in seconds</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>Data Quality Acceptance ON mask</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Data Quality Acceptance OFF mask</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Presence mask</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>Address of first work block</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>Address of first encyclopedia block</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>4 Character mnemonic for this encyclopedia</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>Address of last work block</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>Unused</td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>First encyclopedia volume</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>Last encyclopedia volume</td>
</tr>
<tr>
<td>36</td>
<td>2</td>
<td>Last encyclopedia block</td>
</tr>
<tr>
<td>38</td>
<td>2</td>
<td>Number of characters in title</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>Character title of encyclopedia</td>
</tr>
</tbody>
</table>

* Not actively used at present
<table>
<thead>
<tr>
<th>Byte</th>
<th>Byte Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type, = 0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type, = 0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Address of first EDR block</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Address of last EDR block</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Address of EDR block last modified</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>First slot assigned to EDR's</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>Last slot assigned to EDR's</td>
</tr>
<tr>
<td>Byte</td>
<td>Byte Length</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type, = 0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>address of next block of this type, = 0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>address of first library block</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>address of last library block</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>address of last library block modified</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>first library volume serial</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>last library volume serial</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>first library slot</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>last library slot</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>last library volume serial written on</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>last data set sequence number on above</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>feet used on volume corresp. to last library serial above</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>Maximum amount of tape to be written (in feet) = 1800 feet</td>
</tr>
<tr>
<td>35</td>
<td>18</td>
<td>Model volume DSH (17 char)</td>
</tr>
</tbody>
</table>
|      |             | $V^\left(\begin{array}{c}
\text{A} \\
\text{B}
\end{array}\right)$ \cdot \text{AAXXXXXX}$ |
|      |             | 2 JPL characters | JPL number |
| 54   | 6           | Model volume serial = $M^\left(\begin{array}{c}
\text{1} \\
\text{2}
\end{array}\right)$ \text{WXXD}$ |
WORK CONTROL BLOCK

<table>
<thead>
<tr>
<th>Byte</th>
<th>Byte Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type, = 0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type, = 0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Address of 1st work block</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Address of last work block</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>first work volume serial</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>last work volume serial</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>first work slot</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>last work slot</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Model volume DSN (17 char)</td>
</tr>
</tbody>
</table>

\[ V^{(A)}(B) \]

| 42   | 6           | Model volume serial |

\[ H^{(1)} \_H^{(2)} \_H^{(3)} \]

<p>| 43   | 2           | LAST SERIAL USED |</p>
<table>
<thead>
<tr>
<th>Byte</th>
<th>Byte Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type, = 0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type, = 0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Address of 1st Encyclopedia block</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Address of last Encyclopedia block</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>first encyclopedia volume serial</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>last encyclopedia volume serial</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>first encyclopedia slot</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>last encyclopedia slot</td>
</tr>
<tr>
<td>24</td>
<td>6</td>
<td>Model Volume name (6 char)</td>
</tr>
</tbody>
</table>

<p>| 30   | 2           | LAST ENCY SERIAL USED |
| 32   | 2           | MAXIMUM NUMBER OF BLOCKS |
| 34   | 1           | MODEL VOLUME DSN (18 CHAR) |</p>
<table>
<thead>
<tr>
<th>Byte</th>
<th>Byte Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type, = 0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type, = 0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Address of first CITENCY block</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Address of last CITENCY block</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>First CITENCY volume serial</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>Last CITENCY volume serial</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>First CITENCY slot</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>Last CITENCY slot</td>
</tr>
<tr>
<td>24</td>
<td>6</td>
<td>Model volume name (6 char)</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>Address of free CITENCY volume block</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>Maximum amount of tape to be written on</td>
</tr>
<tr>
<td>Byte</td>
<td>Byte Length</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 13</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>Volume Serial (JPL number)</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>2 Characters in JPL number</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>Slot allocation status</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>Year received</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>Month received</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>Day received</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>Slot number</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>Address of Library entry</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>Number of records</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>Number of errors</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>maximum # of files</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>Number of entries that follow that were processed</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>Processing disposition</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>Year of processing</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>Month of processing</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>Day of processing</td>
</tr>
<tr>
<td>40–43</td>
<td>4</td>
<td>Same as 36–39 above</td>
</tr>
<tr>
<td>44–47</td>
<td>4</td>
<td>Same as 36–39 above</td>
</tr>
<tr>
<td>48–51</td>
<td>4</td>
<td>Same as 36–39 above</td>
</tr>
<tr>
<td>Byte</td>
<td>Byte Length</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 14</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type</td>
</tr>
<tr>
<td>8</td>
<td>8(R#8)</td>
<td>Start-PDSC Start seconds since Jan 1, 1977</td>
</tr>
<tr>
<td>16</td>
<td>8(R#8)</td>
<td>End-PDSC End seconds since Jan 1, 1977</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>Library Tape volume serial</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>Library tape file sequence</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>Address of EDR block</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>Number of entries to follow that processing was completed</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>Processing disposition</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>Year of processing</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>Month of processing</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>Day of processing</td>
</tr>
<tr>
<td>36</td>
<td>2</td>
<td>Address of Ency-Attribute block</td>
</tr>
<tr>
<td>38</td>
<td>2</td>
<td>Address of work block</td>
</tr>
<tr>
<td>40–47</td>
<td>8</td>
<td>Same as 32–39 above</td>
</tr>
<tr>
<td>48–55</td>
<td>8</td>
<td>Same as 32–39 above</td>
</tr>
<tr>
<td>56–53</td>
<td>8</td>
<td>Same as 32–39 above</td>
</tr>
<tr>
<td>Byte</td>
<td>Byte Length</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 15</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type,</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Address of Ency-Attribute block</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>Address of work block of same Ency-Attribute</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Address of next work block of same attribute</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>Number of library blocks</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Start record volume number</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>End record volume number</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>Merge processing disposition</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>Spare</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>Address of Encyclopedia block</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>CITENCY processing disposition</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>spare</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>Address of CITENCY block</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>Address of library blocks corresponding to byte 14 above, each address takes 2 bytes</td>
</tr>
<tr>
<td>62</td>
<td>2</td>
<td>Work table serial number</td>
</tr>
<tr>
<td>Byte</td>
<td>Byte Length</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 16</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Address of attribute block</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Address of previous encyclopedia block for same Ency-Attribute</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Address of next encyclopedia block for same Ency-Attribute</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>Current Block count for this serial</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Start record volume number</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>End record volume number</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>Volume serial of encyclopedia tape</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>Slot number freed by RMVENC</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>Status of this block (00 = active, 01 = dead)</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>Slot number freed by RMVENC</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>Cartridge slot # freed by RMVENC</td>
</tr>
</tbody>
</table>

* documentation added 11/10/78 EWR
<table>
<thead>
<tr>
<th>Byte</th>
<th>Byte Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>block identifier code, = 17</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>year of creation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>month of creation</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>day of creation</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Address of previous block of this type</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Address of next block of this type</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Volume serial of 'raw verse only' encyclopedia</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Volume serial of 'all but raw rates' encyclopedia</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Slot of 'raw verse only'</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Slot of 'all but raw rates'</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>slot allocation status</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>year sent</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>month sent</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>day sent</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>Date acknowledged, year</td>
</tr>
<tr>
<td></td>
<td></td>
<td>month</td>
</tr>
<tr>
<td></td>
<td></td>
<td>day</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>spare</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>Address of work block</td>
</tr>
</tbody>
</table>