MJS Automatic Log System

A

I Voyager Processing Interface

EDRLOG

Main Routine EDRLOG does all its own interface

EDRSAV B

EOE	Open EDR block
EOL	Open Library block
ECLEOT	Close EDR end of Library tape
ECE	Close EDR block
ECL	Close Library block

ENCGEN C

LOL	Open Library
LOW	Open Work
LCW	Close Work
LCL	Close Library
LTERM	Temp Close

WRKCIT D

WCOW	Open Work
WCOC	Open CIT
WCCC	Close CIT
WCCW	Close Work

MERGE E

II UTILITIES F

LREAD (LWRITE)	Read Log in Blocks = 1 block
EOPROL	Open Prolog
RDIRET	Read Directory block
BLDSER	Build [illegible]
DIRSAT	Read Directory
FSTSER	Find first [illegible] Assign a slot

MJS Automatic Log System

III INITIALIZER G

INTLOG — Initialize log
BLOCKS — Block data

IV MAINTAINANCE H

~~ALTER~~ ALTBLK — Alter a byte in the perm log
REDR — Remove EDR tapes
RWORK — Remove work tapes
RCIT — Remove CIT tapes
ASSIGN — Assign work and/or CIT tapes
CITSMT —
CITAKN —

V LIST FEATURES I

CALLIST — driver
BAKLOG — Back log
CITRMV — List of CIT tapes to remove
DMPLOG — Dump log in Hex
EDRRMV — List EDR tapes to remove
INCOMP — List incomplete processes
LISTAL — List all (all or one of following)
LCIT — CIT blocks
LATTR — Attribute blocks
LEDR — EDR "
LENC — Encyclopedia "
LLIB — Library "
LWRK — Work "
REMSLT — Show remaining slots
SHWCIT — driver - CITRMV
SHWEDR — driver - EDRRMV

MJS Automatic Log System

VI User's Guides J

INTLOG

ALTBLK

THS update

LIST

VII Appendix K

COMMONS
BLOCKS
LOGDAT

MJS ~~BL~~LOG 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 EDRLOG. The user inputs the tape vol-ser into the Log. EDRLOG requests that the next process, EDRSAV, be done for this tape and returns the TLS slot number assigned the new EDR tape.

EDRSAV copies the 800 BPI EDR tape onto a file on a Library tape (6250 BPI). EDRSAV requests that the next process, ENCGEN, be run for this file, if EDRSAV successfully processes the EDR tape.

ENCGEN reformats the EDR data into encyclopedia tapes with volumes, chapters and verses. ENCGEN, if successful, requests that

~~the~~

the MERGE and ~~CIT~~ WRKCIT 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 WRKCIT process writes two tapes to be sent to CIT, 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 maybe either finding a block to process or allocating and partially filling a new block.

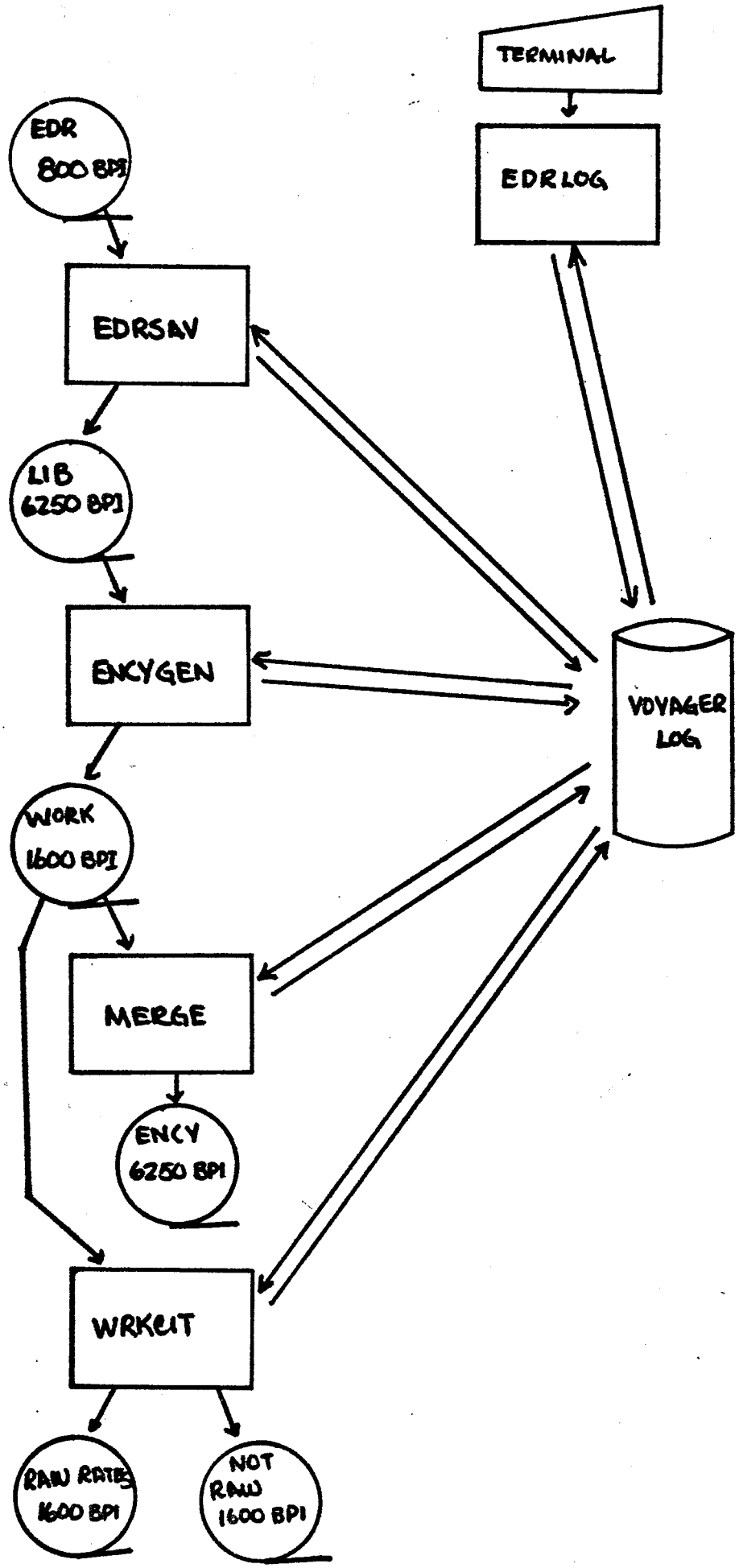
A close updates an old block, mark process done, or writes the new block, mark to process and link with ~~the~~ an ~~previous~~ old block.

The sources of the interface routines are located in the source library of their respective procedure routines, i.e. EOE, EOL, ECE, ECL

and ECCEOT are with EORSAV.

~~See the user's guides for the~~

Since these are interface subroutines they do not have individual user's guides other than their own documentation.


TERMINAL
EDR
800 BPI
EDRLOG
EDRSAV
LIB
6250 BPI
ENCYGEN
VOYAGER
LOG
WORK
1600 BPI
MERGE
ENCY
6250 BPI
WRKCIT
RAW RATES
1600 BPI
NOT
RAW
1600 BPI

1. Subroutine EOE

2. Function

Subroutine EOE (EDR processing - open EDR) locates the next EDR block to be processed.

3. Programmer and Date

Eunice 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.

Argument	Type	I/O	Description
EDRBLK(32)	I*2	O	EDR block
RTCODE	I*4	O	Return code = 0, EDR block found = 4, No EDR block found for processing = 8, error encountered

4.2 Subroutines Called

EOE calls Subroutine RDIRET and LREAD (ENTRY LWRITE). ~~EOE calls SACC DAIO subroutines~~ EOE calls ~~DREAD and~~ SACC ~~function~~ subroutines ~~LOR~~. KOC and KCLC.

4.3 COMMON Variables

EOE ~~assigns~~ updates IEDRBK, EDRSER, EDRDSN, and IEDRSL of COMMON LOGDAT. IMES(26) of COMMON FERMSG is used to ~~xxx~~ write and I/O error message.

EDRSER = JPL number
EDRDSN = CRS

EOE

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

~~Rea~~ Read EDR Control block

READ Do Until (all EDR blocks processed, ie. next EDR @ = 0)

Read EDR block

Get address of next EDR block
IF (#entries .EQ. 4) get next EDR BLOCK and GO TO READ
Get location of last entry

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

6.1 Special Notes

6.2 Major Variables and Constants

Variable or Constant	Type	Description
QMSKTP	L*1	Constant = X'80', flags 'to process'
QMSKPR	L*1	Constant = X'40', flags 'process in progress
NSTBLK	I*4	Address of next EDR block
LASTEN	I*4	Address of last entry in EDR block

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 RDIRET.

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 FORTRANH 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

Eunice Eng, September 1977

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for EOL is as follows:

```
CALL EOL(LIBBLK,RTCODE)
```

The arguments int the calling sequence are described below:

Arguments	Type	I/O	Description
LIBBLK(32)	I*2	O	Library block
RTCODE	I*4	O	Return Code = 0, block allocated = 4, block not allocated = 8, error encountered

4.2 Subroutines Called

EOL calls Subroutines RDIRET, ~~EWRITE~~ LREAD (with ENTRY LWRITE), BLOSEC and EOPROL. ~~EOL also calls SACC DATO subroutine DREAD.~~

4.3 COMMON Variables

EOL updates ILIBBK, LIBSER, I1SLOT, INSLOT, LIBDSN, HLIBSQ, HFEET and HMXFT of COMMON LOGDAT. IMES(26) of COMMON FERMSG is used to write an I/O error message.

$$\text{LIBSER} = \text{M}\binom{1}{2}\text{LXXX}$$

$$\text{LIBDSN} = \text{V}\binom{A}{B}.\text{AAXXXXXX}$$

EOL

```
Set Return Code, RTCODE = 0
Read Directory
If (satellite block address not found, i.e. RTCODE = 8)
     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
          HLBSER
          HLIBSQ
          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 = 8
          EXIT TO RETURN
```

6. Special Notes

6.1 Major Variables

Variable	Type	Description
HNXTAV	I*2	Next available block number
ILIBLS	I*4	Number of the current last Library block.

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 RDIRET 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 XXXXXXXXX.

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 FERMSG. The I/O data is described in DAIO documentation.

1. Subroutine ECLEOT

2. Function

Subroutine ECLEOT (EDR processing -end of tape) declares the present library tape full.

3. Programmer and Date

Eunice 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.

Argument	Type	I/O	Description
RTCODE	I*4	O	Return Code = 0, next vol-ser available = 8, next vol-ser out of range

4.2 Subroutines Called

Subroutine ECLEOT calls subroutine ~~EWRITE~~ LREAD (with ENTRY LWRITE), RDIRET and BLDSER. ~~and SAGG DAIO subroutine DREAD.~~

4.3 COMMON Variables

ECLEOT updates LIBSER, HLIBSQ and HFEET of COMMON LOGDAT. IMES(26) of COMMON FERMSG 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
     HFEET, =0, feet used on tape
     LIBSER, EBCDIC tape vol-ser
     HLBSER, 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 XXXXXX.

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 FERMSG 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.

Argument	Type	I/O	Description
EDRBLK(32)	I*2	I	EDR block
LIBBLK(32)	I*2	I	Library block

4.2 Subroutines Called

ECE calls subroutines ~~EWRITE~~ LREAD (with ENTRY LWRITE) ~~and RDHEAT~~. ECE calls SACC subroutines KTIME, YDMD, ~~and function subroutine LOR and DAIO subroutine DREAD~~ KNC, KOC, and KCLC.

4.3 COMMON Variables

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

ECE

```
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 = Z0F
               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

Variable or Constant	Type	Description
QPRDON	L*1	Constant = X'20', flags production done
HOLDBK(32)	I*2	'Old' EDR block

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:

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

Followed by data passed in IMES926) of COMMON FERMSG. 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 ~~in~~ 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 FERMSG. 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 mesage 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 ~~ee~~ 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 ARCXXXXXX FROM SLOT XXXXXX . ~~UPDATE TLS AND~~ LOG AFTER REMOVAL.

6.4 Other

ECE must be compiled under FORTRANH with parm=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

Eunice 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.

Argument	Type	I/O	Description
EDRBLK(32)	I*2	I	EDR block to be updated
LIBBLK(32)	I*2	I	Library block

4.2 Subroutines Called

ECL calls Subroutine ~~EWRITE~~ LREAD (with ENTRY LWRITE) and R DIRST and SACC ~~DAIO~~ subroutine ~~DRBAD~~ KCLC.

4.3 COMMON Variables

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

ECL

```
Test last EDR entry for success, i.e. E1 or EØ
     If(not successful)
          Write error message
          EXIT to RETURN
Test for altered Library block
     If (Library block tempered 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(@ of prev. lib ≠ Ø)
     estab. forward link
  FI
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

Constant	Type	Description
HPRODS	I*2	Constant = X'8000', flags production to be done

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(26) of COMMON FERMSG. The data is described in the DAIO 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 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 the calling sequence, the following message is written and a 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(26) 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 ECE should be called before EC1, 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 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 -

Argument	Type	I/O	Description
ATTRIB	I*4	I	@ of Attribute block
STIME	R*8	I	FDSC sought
ATTRBK(32)	I*2	O	Ency-Attribute block
LIBBLK(32)	I*2	O	Library block
RTCODE	I*4	O	Return Code =0, Library block found for processing =4, No library block found for processing =8, serious error

4.2 Subroutines Called

LOL calls subroutines BLDSER, ~~EWRITE~~ LREAD (LWRITE) AND RDIRET.
LOL also calls SACC function subroutine LOR and DAIO ~~subroutine DREAD~~.

4.3 COMMON Variables

LOL updates variables I1SLOT, INSLOT, HMXFT, ILIBBK, IATTBK, HLIBSQ, HLBSER, DSTFDS, DENFDS of COMMON LOGDAT. IMES(26) of COMMON FERMSG is used to write an I/O error message.

5. Procedure

```
Set return code, RTCODE = 0, sucessful
Read Directory block
If (satellite block not found, i.e. RTCODE = 0)
     EXIT to RETURN
Read Satellite block
Read Library Control block
     Update COMMON LOGDAT
          I1SLOT
          INSLOT
          HMXFT
     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 FDSC∠ STime)
                    EXIT to NEXTLIB
               If (LATTRB = 0)
                    LATTRB = default to ~~xxxxxxxxx~~ attrb.
                         of the first Library block found
               If (present attrb ≠ LATTRB)
                    EXIT to NEXTLIB
               If (present start FDSC∠ DFDSC)
                    DFDSC = present FDSC
                    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
        ILIBBK = 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
             HLIBSQ
             HLBSER
             DSTFDS
             DENFDS
             LIBSER
             LIBDSN
        Read Attribute block
RETURN
```

6. Special Notes

6.1 Major Variables and/or Constants

Variable or Constant	Type	Description
IPRSBK	I*4	@ of present Library block
QDSMNS(18)	L*1	Mask of DSN for Library tape
QSERMS(6)	L*1	Mask of Library tape vol-ser
LATTRB	I*4	@ of attribute block being processed
NLIBBK	I*4	@ of present Library block that meets time criterian

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 ~~XXXX~~ 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 ~~XXXXXX~~ COMMON FERMSG, described in DAIO documentation.

1. Subroutine LOW

2. Function

Subroutine LOW (Library processing - open work) allocates a new work block.

3. Programmer and Date

Eunice 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.

Argument	Type	I/O	Description
ATTRBK(32)	I*2	I	Ency-Attribute block
WRKBLK(32)	I*2	O	Work block
RTCODE	I*4	O	Return Code =0, work block allocated =4, work block not allocated =8, serious error

4.2 Subroutines Called

LOW calls subroutines RDIRET, BLDSER, EOPROL AND ~~EWRITE~~ LREAD (LWRITE). LOW also calls SACC ~~DAIO subroutine DREAD.~~

4.3 COMMON Variables

The following variables of COMMON LOGDAT are updated: IATTBK, I1WSLT, INWSLT, IWRKBK, WRKDSN, HWKSER, and WRKSER. IMES(26) 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 next tape serial for Library Tape ~~address of block of tape volumes~~

~~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

WRKSER

HWKSER

WRKDSN

IIWSLT

INWSLT

~~Read last work block~~

Clear and Write new work block

Write prolog

Clear rest of entries

Get and record Ency-Attribute address

~~If (default, i.e. ~~[illegible]~~ IATTBK = 0)~~

~~Set default address, IATTBK = 0~~

~~Fi~~

Record address of previous work block of same Ency-Attributes
Record Library tape library address
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

Variable or Constant	Type	Description
NXTBLK	I*4	Address of next block to be allocated
ICNTWK	I*4	Address of Work Control block
ILASTB	I*4	Address of last work block
NXTVOL	I*4	Address of block with next tape volume serial for ~~Library~~ work tapes

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 XXXXXXXXXX.

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 = XXXXXX GREATER THAN XXXXXX

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(WRKBLK,ATTRBK,MAXFLE,RTCODE)
```

The following list defines the arguments in the calling sequence.

Argument	Type	I/O	Description
WRKBLK(32)	I*2	I	Work block
ATTRBK(32)	I*2	I	Ency-Attribute block
MAXFLE	I*4	I	Maximum # of files to process
RTCODE	I*4	O	Return Code = 0, normal return =4 , call LOW before calling another LOL

4.2 Subroutines CAlled

Subroutine	Description
LREAD (LWRITE)	Subroutine to deblock the EDR Log on a read or a write

4.3 COMMON Variables

Variable	Type	I/O	Description
IATTBK	I*4	I	@ Ency-Attribute - COMMON LOGDAT
ILIBBK	I*4	I	@ of Library Block - COMMON LOGDAT
I1RVOL	I*4	I	Start record volume - COMMON LOGDAT
INRVOL	I*4	I	End record Volume - COMMON LOGDAT
IMES(26)	I*4		I/O error data - COMMON FERMS
IWRKBK	I*4	I	@ of Work block - COMMON LOGDA

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 , RTCODE = 0
Update Work Block
     Place Library address in work list
     Mark start and end record volume
          If (first Library block processed for this work block)
               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 (successful not successful)
                              EXIT to RETURN
                    Read Satellite block
                    Read Work Control block
                    If (first work block)
                         record first work block
                    fi
```

```
                                    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 MAXFLE ∋ 0 < MAXFLE < 15)
                              MXFILE = MAXFLE
                         If (# of Library blocks = MXFILE)
                              RTCODE = 4
                         Fi
RETURN
```

6. Special Notes

6.1 Major Variables and/or Constants

Variable or Constant	Type	Description
HTODO	I*2	= X'8000', flags a request for processing
IBLKNO	I*4	Library block address
HPRWRK	I*2	Present work block address
LSTWRK	I*4	Address of previous work block of same Ency-Attributes
ILASTW	I*4	Adr of last work block pointed to by present work block

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 DAIO 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 15 library addresses in its list, a return code of 4 is passed through the calling sequence to request that subroutine LOW 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

Eunice Eng, October 1977

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for LCL is as follows:

CALL LCL(LIBBLK,WRKBLK, ERROR)

The following ~~XXXX~~ list describes the arguments in the calling sequence.

Argument	Type	I/O	Description
LIBBLK(32)	I*2	I	Library block
WRKBLK(32)	I*2	I	Work block
ERROR	I*4	I	last byte contains error code

4.2 Subroutines Called

LCL calls subroutine ~~E~~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 dispositon

Record date of process

Record @ of Ency-Attribube

Record @ of Work block

Write Updated Library block

RETURN

6. Special Notes

6.1 Major Variables and/or Constants

Constant	Type	Description
~~QPRODN~~ QTCLOS	L*1	= X'~~20~~' 30, flags processing ~~done~~ temporarily closed

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 successufully been added to the work list in the work block (i.e. subroutine LCL called before subroutine LCW), the following message is printed and a return taken.

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

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

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

*************************************************
```

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

LCL ***** I/O ERROR ENCOUNTERED.

1. Subroutine LTERM

2. Function

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

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(WRKBLK)
```

The following describes the argument in the calling sequehce.

Argument	Type	I/O	Description
WRKBLK(32)	I*2	I	Work block

4.2 Subroutines Called

Subroutine	Description
LREAD (LWRITE)	Reads an EDR Log block and unblocks a record. LWRITE reads before writing
KCLC	SACC - compares strings byte by byte
ABEND	SACC - abend the program

4.3 COMMON Variables

Variable	Type	I/O	Description
IWRKBK	I*4	I	@ of work block - COMMON LOGDAT
INRVOL	I*4	I	End record volume - COMMON LOGDAT
IMES(26)	I*4		I/O error data - COMMON FERMSG

LTERN

5. Procedure

→ check if work [illegible]

```
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

Variable or Constant	Type	Description
HTODO	I*2	Z8000, to process flag

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 XXXXX) HAS
    BEEN TEMPERED.
ABENDING FROM SUBROUTINE LTERM.
*****************************************************
```

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

```
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

Eunice 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.

ARGUMENT	TYPE	I/O	DESCRIPTION
WRKBLK(32)	I*2	O	WORK BLOCK
ILOGUN	I*4	I	DEVICE NUMBER OF LOGICAL UNIT OF LOG
RTCODE	I*4	O	RETURN CODE 0 = NORMAL RETURN 4 = NO WORK BLOCK FOUND MARKED FOR CITENCY PROCESSING 8 = SERIOUS ERROR

4.2 Subroutines Called

WCOW calls subroutines RDIRET, BLDSER, and ~~BWRITE~~ LREAD (with SETEV LWRITE). WCOW also calls SACC subroutines ~~DREAD~~, KNC and KOC.

4.3 COMMON Variables

The list below describes the variables used from COMMON LOGDAT.

VARIABLE	TYPE	I/O	DESCRIPTION
BWKSER	I*2	O	BINARY WORK TAPE SERIAL
IIRVOL	I*4	O	FIRST RECORD VOLUME ON WORK TAPE
INRVOL	I*4	O	LAST RECORD VOLUME ON WORK TAPE
IIWSLT	I*4	O	FIRST WORK SLOT
INWSLT	I*4	O	LAST WORK SLOT
IWRKBK	I*4	O	# OF WORK BLOCK
WRKDSN(18)	L*1	O	MODEL OF WORK DSN, EBCDIC
WRKSER(6)	L*1	O	WORK TAPE VOL-SER, EBCDIC
GSATID(12)	L*1	I	SATELLITE ID, EBCDIC

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

```
WCOW
          Reset Return Code
               RTCODE = 0
          Read Directory block
               If (satellite block not found)
                    ~~Write error message~~
                    EXIT to RETURN
          Read Satellite Block
          Read Work ~~Block~~ Control Block
               Update Common LOGDAT
                    IWRKBK
                    WRKDSN
                    I1WSLT
                    INWSLT
               Get mask of work vol-ser
WORK      Find Work Block marked for CITENCY processing
               Read Work block
                    If (work block marked for CITENCY processing)
                         Mark processing begun
                         Update COMMON LOGDAT
                              I1RVOL
                              INRVOL
                              HWRSER
                              WRKSER
                         Write Work block
                         EXIT to RETURN
                    Get next work block
                         If (address of next work block ≠ 0)
                              EXIT to WORK
```

WCOW

Write error message

RTCODE = 4

RETURN Return

Note: If(I/O error during DAIO)

Write error message

RTCODE = 8

EXIT to RETURN

6. Special Notes

6.1 Major Variables and Constants

Constant or

VARIABLE	TYPE	DESCRIPTION
GTCOO	L*1	=Z80, FLAGS 'TO DO'
GBEGUN	L*1	=Z40, FLAGS 'PROCESS BEGUN'
ISATAD	I*4	@ OF SATELLITE BLOCK
IVOLSR	I*4	SERIAL OF WORK TAPE
IWRKCB	I*4	@ OF WORK CONTROL BLOCK
CWRKSR(6)	L*1	WORK SERIAL MASK

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 CITENCY processing the following message is printed, a return code of 4 is set and a return taken.

WCOW ***** NO WORK BLOCK FOUND MARKED FOR CITENCY PROCESSING.

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

WCOW ***** I/O ERROR ENCOUNTERED.

1. Subroutine WCOC

2. Function

Subroutine WCOC (work processing CITENCY-open ~~a~~ CITENCY) ~~opens a CITENCY block~~. allocates 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, RTCODE)

The arguments in the calling sequence are described below.

ARGUMENTS	TYPE	I/O	DESCRIPTION
CITBLK(32)	I*2	O	CITENCY BLOCK
ILOGUN	I*4	I	LOGICAL UNIT FOR LOG
RTCODE	I*4	O	RETURN CODE =0, CITENCY BLOCK ALLOCATED =4, CITENCY BLOCK NOT ALLOC. =8, SERIOUS ERROR

4.2 Subroutines Called

~~Where~~ WCOC calls subroutines BLDSER, EOPROL, RDIRET and ~~WRITE~~ LREAD (with entry LWRITE). ~~WCOC also calls DAW subroutine DREAD~~

WC00

4.3 COMMON Variables

The list below describes the variables used from COMMON LOGDAT.

VARIABLE	TYPE	I/O	DESCRIPTION
FNRWSR	I*2	O	BINARY NOT RAW RATES' SERIAL
FRAWSR	I*2	O	BINARY RAW RATES' SERIAL
ICITBK	I*4	O	# OF CITENCY BLOCK ALLOCATED
IWRKBK	I*4	I	# OF WORK BLOCK
CCMDSN(18)	L*1	O	MASK OF CITENCY DSN, EBCDIC
CNRWSR(6)	L*1	O	NOT RAW RATES' SERIAL, EBCDIC
CRAWSR(6)	L*1	O	RAW RATES' SERIAL, EBCDIC
QSATID(12)	L*1	I	SATELLITE ID, EBCDIC

IMES(20) 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 → set slot file flag

EXIT to RETURN

Update LOGDAT

HRAWSR

HNRWSR

QRAWSR

QNRWSR

QCMDSN

Get address of last CITENCY block

WCOC

Fill new CITENCY block

Prolog

Raw rates' serial

Not raw rates' serial

Mark slot allocation - ~~allocated~~ requested

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 = 8

EXIT to RETURN

6. Special Notes

6.1 Major Constants and Variables

CONSTANT OR VARIABLE	TYPE	DESCRIPTION
LSCIT	I*4	@ OF LAST CITENCY BLOCK
GALLO	L*1	=Z00, ALLOC. AND GET CIT TAPE SLOT
ICITCN	I*4	@ OF CITENCY CONTROL BLOCK
ISATAD	I*4	@ OF SATELLITE BLOCK

6.2 Data Structure and Tables

See Table of for the format of the CITENCY block.

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.

WCOC ***** NEXT BLOCK TO BE ALLOCATED EXCEEDS XXXXXX.

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

WCOC ***** CITENCY SLOTS ALL USED.
PLEASE CLEAR SLOTS FOR NEXT SET OF CITENCY SERIALS.

If WCOC 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.

WCOC ***** I/O ERROR ENCOUNTERED.

The message is followed by I/O data passed by COMMON FDRMSG.

1. Subroutine WCCC

2. Function

Subroutine WCCC (work processing CITENCY - close CITENCY) closes a CITENCY block.

3. Programmer and Date

Eunice Eng, November 1977

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for WCCC is as follows:

```
CALL WCCC(CITBLK,ILOGUN)
```

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

ARGUMENTS	TYPE	I/O	DESCRIPTION
CITBLK(32)	I*2	I	CITENCY BLOCK
ILOGUN	I*4	I	LOGICAL UNIT FOR LOG

4.2 Subroutines Called

WCCC calls subroutines EWRITE and RDIRET. WCCC also calls SACC subroutine KCLC ~~and DATO subroutine DREAD.~~

4.3 COMMON Variables

The list below describes the variables used from COMMON LOGDAT.

VARIABLE	TYPE	I/O	DESCRIPTION
INRWSR	I*2	I	BINARY NOT RAW SERIAL
ICITBK	I*4	I	# OF CITENCY BLOCK
INRWSL	I*4	I	SLOT FOR NOT RAW RATES
IRAWSL	I*4	I	SLOT FOR RAW RATES
CSATID(12)	L*1	I	SATELLITE ID, EBCDIC

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

WCCC

```
READ CITENCY BLOCK from SDR LOG
If (CITENCY block tempered during processing)
     Write error message
     ABEND(300)
Update CITENCY block
     Record slot of raw rates' tape
     Reocrd 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 REWRN ABEND
     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 CITENCY block

EXIT to RETURN

ABEND Abend (400)

RETURN Return

Note: If(I/O error during DAIO)

Write error message

EXIT to ABEND

6. Special Notes

6.1 Major Variables and Constants

Constant or

VARIABLE	TYPE	DESCRIPTION
ILAST	I*4	@ OF LAST CITENCY BLOCK
ICITCN	I*4	@ OF CITENCY CONTROL BLOCK
ISATAD	I*4	@ OF SATELLITE BLOCK
CREMCV	L*1	=Z20, REQUESTS REMOVAL OF TAPE

6.2 Data Structure and Tables

See Table of for the format of a ~~work~~ CITENCY block.

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 XXXXXX)
     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 FERMSG, and a user ABEND (400) is taken.

```
WCCC ***** I/O ERROR ENCOUNTERED.
```

1. Subroutine WCCW

2. Function

Subroutine WCCW (work processing CITENCY - close work) closes a work block.

3. Programmer and Date

Eunice 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.

ARGUMENT	TYPE	I/O	DESCRIPTION
WRKBLK(32)	I*2	I	WORK BLOCK
CITBLK(32)	I*2	I	CITENCY BLOCK
ILOGUN	I*4	I	LOGICAL UNIT NUMBER OF LOG

4.2 Subroutines Called

WCCW calls subroutine ~~IWRITE~~ LREAD (with ENTRY LWRITE). WCCW also calls SACC subroutines KCLC and ABEND ~~and DAIO subroutine DREAD.~~

4.3 COMMON Variables

The list below describes the variables used from COMMON LOGDAT.

VARIABLES	TYPE	I/O	DESCRIPTION
ICITBK	I*4	I	# OF CITENCY BLOCK
IWRKBK	I*4	I	# OF WORK BLOCK

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

WCCW

Temper Test

Read Work block from EDR Log

If (Work block tempered during processing)

Write error message

ABEND (300)

Check CITENCY closure

If (CITENCY slot allocation not marked for ~~me~~ removal)

Write error message

~~V~~ EXIT to RETURN

Update Work block

Mark processing done

Record CITENCY block address

Write Updated Work block

RETURN Return

Note: If I/O error encountered during a DREAD
Write error message.
EXIT to RETURN

6. Special Notes

6.1 Major Variables and Constants

Constant or VARIABLE	TYPE	DESCRIPTION
DONE	L*1	=Z20, FLAGS 'PROCESSING DONE'
DREMOV	L*1	=Z20, FLAGS 'REMOVE TAPE'

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 XXXXXX)
     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 sequenences 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.

Argument	Type	I/O	Description
UNIT	I*4	I	Logical unit number of EDR log
RECORD	I*4	I	Record number
AREA(16)	I*4	O	One Log record
*			Statement number to branch to in case of I/O error

4.2 Subroutines Called

Subroutine	Description
DREAD	FTIO - reads a logical record from ~~disk~~ LOG
DWRITE	FTIO - writes a logical record to LOG
KMVC	SACC - moves bytes from one area into ~~another~~ another

4.3 COMMON Variables

NONE

LREAD

```
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

LWRITE    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

Variable Or Constant	Type	Description
BUFFER(1808)	I*4	buffer of one log block
BLOCK	I*4	block number
DISP	I*4	displacement intb block of record

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, ~~further.~~

```
Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C 1.  Routine:  Subroutine LREAD
C
C 2.  System, Satellite, Version:
C     MJS Automatic Log Interface, Voyager-1 and -2, Ver. 1
C
C 3.  English Name:  Log Read
C
C 4.  Language:
C        FORTRANG or FORTRAN4   level 21.6    360/91/75   OS/MVT
C
C 5.  Purpose:
C     Lread unblocks MJS Log data blocks.
C
C 6.  Calling Sequence:
C     CALL LREAD(UNIT,RECORD,AREA,*)
C     CALL LWRITE(UNIT,RECORD,AREA,*)
C     Argument          Type    I/O    Description
C     UNIT              I*4     I      logical unit number of MJS Log
C     RECORD            I*4     I      record number
C     AREA(16)          I*4     O      one Log record
C     *                 Addr           statement number to branch to
C                                      incase of I/O error
C
C 7.  Notes:
C     7a.  Restrictions:  None
C
C     7b.  Special Features:  None
C
C 8.  Variables:
C     8a.  Local
C          Variable          Type    Description
C          BUFFER(1808)      I*4     buffer of one Log block
C          BLOCK             I*4     block number
C          DISP              I*4     displacement into block of record
C
C     8b.  COMMON
C          COMMON            Variables
C          None
C
C 9.  I/O Information:
C     Unit No.        Use Description
C     UNIT            MJS Log
C
C 10. Error Handling:
C     I/O error          an alternate return taken
C
C 11. Subroutines Called:
C     Subroutine      Description
C     DREAD           DAIO - reads a logical record from Log
C     DWRITE          DAIO - writes a logical record to Log
C     KMVC            SAGC - moves bytes from one area into another
C
C 12. Called By:
C     Routine         Description
C     MJS Log Utility
C
C 13. Method:
C                Calc. block # of the record
C                Calc. displacement into block of record
C                Read the block
C                IF (I/O error)
C                  EXIT to RETURN!
C                Move 64 bytes from read buffer into user area
C                RETURN
C
C     LWRITE     Calc. block # of record
C                Calc. displacement into block of record
C                Read the block
C                IF (I/O error)
C                  EXIT to RETURN!
C                Move 64 bytes from user area to read buffer
C                Write read buffer block to Log
C                RETURN
C
C 14. Reference:
C     Format for MJS Log
```

```
C       MJS Log Structure:                                             000007
C         # records/block  =    113                                    000007
C         bytes/block      =   1308                                    000008
C         bytes/record     =     64                                    000008
C                                                                      000008
C 15.   Programmer and Date:                                           000008
C       Eunice Eng, Computer Sciences Corp., Sept. 1977                000008
C                                                                      000008
C 16.   Modifications:  None                                           000008
C                                                                      000008
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   000008
```

*** END OF MEMBER ***   83 RECORDS PROCESSED   *******************************

1. Subroutine EOPROL

2. Function

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

3. Programmer and Date

Eunice 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.

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

4.2 Subroutines Called

EOPROL calls SACC Subroutines KTIME and YDMD.

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

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   00000
C 1.    Routine:  Subroutine EOPROL                                         00000
C                                                                           00000
C 2.    System, Satellite, Version:                                         00000
C       MJS Automatic Log Interface, Voyager-1 and -2, Ver. 1               00000
C                                                                           00000
C 3.    English Name:  Open Prolog                                          00000
C                                                                           00000
C 4.    Language:                                                           00000
C              FORTRANG or FORTRANH   level 21.6   360/91/75   OS/MVT       00000
C                                                                           00000
C 5.    Purpose:                                                            00000
C       EOPROL writes part of the MJS log prolog for a block.               00000
C                                                                           00000
C 6.    Calling Sequence:                                                   00000
C       CALL EOPROL(IBLKCD,IPREAD,HBLOCK,HBLOCK)                            00000
C       Argument            Type    I/O    Description                      00000
C       IBLKCD              I*4     I      block type code.  See MJS        00000
C                                          Log for codes.                   00000
C       IPREAD              I*4     I      block number of the previous     00000
C                                          block of similar code.           00000
C       HBLOCK(32)          I*2     I/O    block to have prolog written.    00000
C                                                                           00000
C 7.    Notes:                                                              00000
C       7a.  Restrictions:  None                                            00000
C                                                                           00000
C       7b.  Special Features:  None                                        00000
C                                                                           00000
C 8.    Variables:                                                          00000
C       8a.  Local                                                          00000
C            Variable          Type    Description                          00000
C            None                                                           00000
C                                                                           00000
C       8b.  COMMON -                                                       00000
C            COMMON            Variables                                    00000
C            None                                                           00000
C                                                                           00000
C 9.    I/O Information:                                                    00000
C       Unit No.         Use Description                                    00000
C       None                                                                00000
C                                                                           00000
C10.    Error Handling:  None                                               00000
C                                                                           00000
C11.    Subroutines Called:                                                 00000
C       Subroutine       Description                                        00000
C       KTIME            SACC - gets current system time.                   00000
C       YDMD             SACC - converts current system time to             00000
C                         year, month and day.                              00000
C                                                                           00000
C12.    Called By:                                                          00000
C       Routine          Description                                        00000
C       MJS Log Utility                                                     00000
C                                                                           00000
C13.    Method:                                                             00000
C       Get block identifier code                                           00000
C       Get creation date                                                   00000
C       Get address of previous block of this type                          00000
C       Zero address of next block of this type                             00000
C       Return                                                              00000
C                                                                           00000
C14.    Reference:                                                          00000
C       Format for MJS Log                                                  00000
C                                                                           00000
C15.    Programmer and Date:                                                00000
C       Eunice Eng, Computer Sciences Corp., Sept. 1977                     00000
C                                                                           00000
C16.    Modifications:  None                                                00000
C                                                                           00000
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   00000
```

*** 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,HDIRBK,HDIRBK,ISATAD,RTCODE)
```

The list below describes the arguments in the calling sequence.

Argument	Type	I/O	Description
NUNIT	I*4	I	Unit number of EDR Log
HDIRBK(32)	I*2	O	Directory block
ISATAD	I*4	O	Address of Satellite block
RTCODE	I*4	O	Return Code = 0, normal return = 8, problems encountered

4.2 Subroutines Called

RDIRET calls ~~SACC DAIO~~ subroutine DREAD.

4.3 COMMON Variables

RDIRET uses IMES(26) of COMMON FERMSG to wirte an I/O error message from called subroutine DREAD. RDIRET updates ISATCD ØF 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 and 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 = AAAAAAAAAAAA NOT FOUND.
RDIRET ***** I/O ERROR ENCOUNTERED.
```

The I/O error message is followed by data from COMMON FERMSG. For description of the data see SACC DAIO documentation.

```
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C 1.  Routine:  Subroutine RDIRET
C
C 2.  System, Satellite, Version:
C     MJS Automatic Log Interface, Voyager-1 and -2, Ver. 1
C
C 3.  English Name:  Read Directory
C
C 4.  Language:
C         FORTRANG or FORTRANH   level 21.6    360/91/75    OS/MVT
C
C 5.  Purpose:
C     RDIRET reads the directory block and returns the satellite
C     block address and the directory block.
C
C 6.  Calling Sequence:
C     CALL RDIRET(NUNIT,HDIRBK,HDIRBK,ISATAD,RTCODE)
C     Argument              Type     I/O     Description
C     NUNIT                 I*4      I       unit number of MJS Log
C     HDIRBK(32)            I*2      O       directory block
C     ISATAD                I*4      O       address of satellite block
C     RTCODE                I*4      O       return code
C                                            0 = normal return
C                                            8 = problems encountered
C
C 7.  Notes:
C     7a.   Restrictions:  None
C
C     7b.   Special Features:  None
C
C 8.  Variables:
C     8a.   Local
C           Variable            Type    Description
C           None
C
C     8b.   COMMON
C           COMMON              Variables
C           FERMSG              IMES(26)  - I/O
C           LOGDAT              ISATCD  - I/O
C
C 9.  I/O Information:
C     Unit No.          Use Description
C     NUNIT             MJS Log
C
C10.  Error Handling:
C     Directory block             write error message, set return
C      not found                   code to 8 and return
C     Satellite id                write error message, set return
C      not found                   code to 8 and return
C     I/O error                   write error message, set return
C                                  code to 8 and return
C
C11.  Subroutines Called:
C     Subroutine        Description
C     LREAD             reads MJS Log doing its own deblocking
C
C12.  Called By:
C     Routine           Description
C     MJS Log Utility
C
C13.  Method:
C     Set Return Code = 8
C     Read Directory block
C        IF (I/O error)
C           Write error message
C           EXIT to RETURN
C        IF (directory block not read)
C           Write error message
C           EXIT to RETURN
C        IF (satellite id not found)
C           Write error message
C           EXIT to RETURN
C        ELSE
C           Update satellite ID in COMMON LOGDAT
C           Get satellite block address
C           Reset Return Code = 0
C     Return
```

```
C                                                                         00000
C 14.  Reference:                                                         00000
C      Format for MJS Log                                                 00000
C      COMMON LOGDAT                                                      00000
C      FTIO  COMMON FERMSG                                                00000
C                                                                         00000
C 15.  Programmer and Date:                                               00000
C      Eunice Eng, Computer Sciences Corp., Sept. 1977                    00000
C                                                                         00000
C 16.  Modifications:  None                                               00000
C                                                                         00000
Cccccccccccccccccccccccccccccccccccccccccccccccccccccc                    00000
*** 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

Eunice 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.

Argument	Type	I/O	Description
IVOSER	I*4	I	Binary Volume- Serial of tape
QVOSER(6)	L*1	O	EBCDIC vol-ser
QMSKAD(6)	L*1	I	Mask for vol-ser

4.2 Subroutines Called

BLDSER 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.

```
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C 1.  Routine:  Subroutine BLDSER
C
C 2.  System, Satellite, Version:
C     MJS Automatic Log Interface, Voyager-1 and -2, Ver. 1
C
C 3.  English Name:  Build Serial
C
C 4.  Language:
C          FORTRANH   level 21.6    360/01/75   OS/MVT
C
C 5.  Purpose:
C     BLDSER builds a 6 digit, EBCDIC, tape volume-serial
C     number given a 4-byte integer.
C
C 6.  Calling Sequence:
C     CALL BLDSER(IVOSER,QVOSER,QMSKAD)
C     Argument          Type    I/O    Description
C     IVOSER            I*4     I      binary volume-serial of tape
C     QVOSER(6)         L*1     O      EBCDIC vol-ser
C     QMSKAD(6)         L*1     I      mask for vol-ser
C
C 7.  Notes:
C     7a.  Restrictions:  Compile under FORTRANH w/o parm=XL for
C        for FORTRAN Subroutine LOR.
C
C     7b.  Special Features:  None
C
C 8.  Variables:
C     8a.  Local
C          Variable          Type    Description
C          None
C
C     8b.  COMMON
C          COMMON            Variables
C          None
C
C 9.  I/O Information:
C     Unit No.        Use Description
C     None
C
C10.  Error Handling:  None
C
C11.  Subroutines Called:
C     Subroutine      Description
C     INCORE          SACC - converts binary to EBCDIC
C     LOR             SACC - logical OR
C
C12.  Called By:
C     Routine         Description
C     MJS Log Utility
C
C13.  Method:
C     Convert binary vol-sr to EBCDIC vol-ser
C     Get vol-ser using the vol-ser mask
C
C14.  Reference:
C     A Tape vol-ser has 6 characters.  The first 3 define
C        the satellite and process.  The last 3 are used as
C        a decimal serial.
C
C15.  Programmer and Date:
C     Eunice Eng, Computer Sciences Corp., Sept. 1977
C
C16.  Modifications:  None
C
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
*** 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,RTCODE)
```

The following list describes the arguments in the calling sequence.

Argument	Type	I/O	Description
IREC(16)	I*4	O	directory block
QSATID(12)	L*1	I	satellite ID.
ISATAD	I*4	O	address of the satellite block
RTCODE	I*4	O	return code 0= normal 4= satellite block not found 8= I/O error

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. Proceedure.

```
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. Speical 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 = AAAAAAAAAAAA IS INVALID.

If an I/O error is encountered ~~xxxxxx~~ 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 rnge of serials.

3. Programmer and Date

Eunice Eng, January 1978

4. Interface

4.1 Calling Sequence

The FORTRAN calling sequence for ESTSER is as follows:

```
CALL FSTSER(LAST,ISPAN,DMSK,IFIRST)
```

The following list describes the arguments in the calling list.

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

4.2 Subroutines Called

Subroutine	Description
INCORE	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. Proceedure

```
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

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC          00000
C 1.    Routine:   INILOG                                                     00000
C                                                                             00000
C 2.    System, Satellite, Version:                                           00000
C       Initialize MJS Log, Voyager-1 and -2, Ver. 1                          00000
C                                                                             00000
C 3.    English Name:  Initialize Log                                         00000
C                                                                             00000
C 4.    Language:                                                             00000
C          FORTRANG or FORTRANH    level 21.6     360/91/75    OS/MVT         00000
C                                                                             00000
C 5.    Purpose:                                                              00000
C       Initialize MJS Log up through the control blocks for 2 satellites.    00000
C                                                                             00000
C 6.    Calling Sequence:                                                     00000
C       Argument             Type     I/O     Description                     00000
C       None                                                                  00000
C                                                                             00000
C 7.    Notes:                                                                00000
C       7a.   Restrictions:  Only initializes for 2 satellites.  MJS Log      00000
C         Directory block has room for 3 satellites.                          00000
C                                                                             00000
C       7b.   Special Features:  None                                         00000
C                                                                             00000
C 8.    Variables:                                                            00000
C       8a.   Local                                                           00000
C             Variable               Type     Description                     00000
C             None                                                            00000
C                                                                             00000
C       8b.   COMMON                                                          00000
C             COMMON                 Variables                                00000
C             BLOCKS                 QBLK(64,15)  - I                         00000
C                                                                             00000
C 9.    I/O Information:                                                      00000
C       Unit No.         Use Description                                      00000
C          25            MJS Log                                              00000
C                                                                             00000
C10.    Error Handling:  None                                                 00000
C                                                                             00000
C11.    Subroutines Called:                                                   00000
C       Subroutine       Description                                          00000
C       DWRITE           DAIO - write a record to disk buffer                 00000
C       DCLOSE           DAIO - sends last disk buffer to disk                00000
C       KTIME            SACC - gets system date                              00000
C       YDMD             SACC - translates system date to yr/mo/da            00000
C                                                                             00000
C12.    Called By:                                                            00000
C       Routine          Description                                          00000
C       None                                                                  00000
C                                                                             00000
C13.    Method:                                                               00000
C       Zero MJS Log                                                          00000
C       Get date of creation from the system                                  00000
C       DO UNTIL (all control blocks written)                                 00000
C         Put in creation date                                                00000
C         Write a block of block data                                         00000
C       OD                                                                    00000
C       Close disk data set                                                   00000
C       Stop                                                                  00000
C                                                                             00000
C14.    Reference:                                                            00000
C       Format for MJS Log                                                    00000
C       COMMON BLOCKS                                                         00000
C                                                                             00000
C15.    Programmer and Date:                                                  00000
C       Eunice Eng, Computer Sciences Corp., Oct. 1977                        00000
C                                                                             00000
C16.    Modifications:  None                                                  00000
C                                                                             00000
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC          00000
```

*** END OF MEMBER ***   70 RECORDS PROCESSED   ***************************

```
C
C 1.   Routine:  ALTBLK
C
C 2.   System, Satellite, Version:
C      MJS Log Maintenance, Voyager-1 and -2, Ver. 2
C
C 3.   English Name:  Alter Block
C
C 4.   Language:
C          FORTRANG or FORTRANH   level 21.6    360/91/75   OS/MVT
C
C 5.   Purpose:
C      Alter a byte in a given MJS Log block.  Usually used to
C      reset flags for processing dispositions.  Designed for
C      TSO (time sharing option) real-time execution.
C
C 6.   Calling Sequence:
C      Argument            Type    I/O    Description
C      None
C
C 7.   Notes:
C      7a.  Restrictions:  None
C
C      7b.  Special Features:  Hardcopy documents why a change was made
C           and what the change was.
C
C 8.   Variables:
C      8a.  Local
C           Variable            Type     Description
C           IBLOCK              I*4      Input card variable BLOCK.  Block
C                                         to be altered.
C           IBYTE               I*4      Input card variable BYTE.  Byte to
C                                         be altered in BLOCK.
C           QVAL                L*1      Input card hex variable QVAL.
C                                         Value put into BYTE of BLOCK.
C
C      8b.  COMMON
C           COMMON              Variables
C           FERMSG              IMES-1
C
C 9.   I/O Information:
C      Unit No.          Use Description
C         05             Input data, terminal
C         06             Output data, terminal
C         10             Output hardcopy, sysout=A
C         25             MJS Log, disk data set
C
C 10.  Error Handling:
C      I/O error on          Writes error message and terminates.
C       MJS Log
C
C 11.  Subroutines Called:
C      Subroutine        Description
C      LREAD             Reads MJS Log and returns correct record.
C       (LWRITE)         entry in LREAD.  Writes a record to MJS Log.
C
C 12.  Called By:
C      Routine           Description
C      None
C
C 13.  Method:
C         DO UNTIL (All input cards exhausted)
C            Request, Read and Record Reason for altering block
C            Request, Read and Record
C              Block to alter
C              Byte in Block to alter
C              Value that will go into the Byte
C            Read Block
C            Replace Byte
C            Write block back to MJS Log
C            Display block as altered
C         END
C         Stop
C
C 14.  Reference:
C      MJS Log Format
C
```

```
C 15.   Programmer and Date:                                     00000
C       Eunice Eng, Computer Sciences Corp., Nov. 1977           00000
C                                                                00000
C 16.   Modifications:                                           00000
C       Ver. 2:  Documentation feature of asking for a reason for 00000
C                altering was implemented.                       00000
C                                                                00000
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC       00000
*** END OF MEMBER ***     85 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 ~~XXX~~ 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

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

4.3 COMMON Variables

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

4.4 I/O Devices

Unit NO.	Description
05	terminal, input data
06	terminal, output messages
10	sysout A, hardcopy of data output
25	MJS Log

4.5 Input

VAriable	Type	Description
QSATID(12)	L*1	satellite ID. User only inputs satellite number, QSATID(9)
ISER1	I*4	first serial in the range of serials to be removed
ISERN	I*4	last serial in the range of serials to be removed

Format

1. satellite ID card

   column 12345

   1
   or
   2

2. serial range card

   column 12345678901234567890
   prompt first last

   XXX XXX
   ISER1 ISERN

5. Proceedure

```
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 ≠ ~~ISER~~ 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 (z00)
         or both done (z20))
          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
     Write remove successful message
OD

STOP
     Stop
```

6. Special Notes

6.1 Variables and Constants

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

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 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 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 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.

~~THE ABOVE MESSAGE IS PRINTED~~ 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

Subroutine	Description
KCLC	SACC - logical compare
ASSIGN	assigns slots to vol-ser in TLS
SHOW	displays the slot a vol-ser is assigned or vice-versa
LREAD (LWRITE)	deblocks and reads and/or writes a MJS Log record
DIRSAT	reads the directory ~~xxxx~~ for the satellite block number
KMVC	SACC - logical byte move

4.3 COMMON Variables

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

4.4 I/O Devices

Unit No.	Description
05	terminal, input data
06	terminal, output messages
10	sysout A, hardcopy of session
25	MJS Log

4.5 Input

Variable	Type	Description
QSATID(12)	L*1	satellite ID. Program requests only the satellite number, QSATID(9).
QTYPE(4)	L*4	type of tapes to assign. 'WORK' = assign work tapes 'CIT ' = assign CIT tapes
MAX	I*4	maximum amount of tapes to assign.

Format

1. Satellite ID card

   column 12345

   1
   or
   2

2. Tape type card

   column 12345678901234567890

   work
   cit

3. Maximum amount card

   column 12345

   XX

   $\overbrace{\text{00 to 24}}$

5. Proceedure

```
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 maximun, 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 xxxxx serial assigned in Control
          I1SLOT = 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

Variable	Type	Description
MAX	I*4	maximum amount of assigns requested by user. Defaults 25 for WORK 24 for CIT
DMSK	R*8	EBCDIC vol-ser
QMSK(8)	L*1	equivalent to DMSK
IDID	I*4	number of assings completed
ICITA	I*4	block number of CIT Control
ISLOT	I*4	current TLS slot
IWRKA	I*4	block number of WORK Control

6.2 Data Structure and Tables

~~Skkuxkk~~ 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 = AAAAAAAAAAAA IS INVALID.

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

MAX GREATER THAN 24.

If a tape ~~kpa~~ 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 ~~wil~~ while reading or writting 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

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

4.3 COMMON Variable

Variable	type	Description
IMES(26)	I*4	DAIO I/O error data

4.4 I/O Devices

Unit No.	Description
05	terminal, input data
06	terminal, output messages
10	~~xxxxxx~~ sysout A, hardcopy of session
25	MJS Log

4.5 Input

Variable	Type	Description
QSATID(12)	L*1	satellite ID. Only the satellite number is requested QSATID(9).
ISER1	I*4	first serial in range to be removed
ISERN	I*4	last serial in range to be removed.

Format

1. satellite id card

   column 12345

   1
   or
   2

2. serial range card

```
column 12345678901234567890
prompt first  last
         XXX   XXX
         ISER1 ISERN
```

5. Proceedure

```
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

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

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 explanitory.

ISER1 = XXX IS GREATER THAN ISERN = XXX.

FIRST MUST BE ODD. $\ast \begin{cases} \text{FIRST} = \text{ISER1} \\ \text{LAST} = \text{ISERN} \end{cases}$

LAST MUST BE EVEN.

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.

ISER1 = 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 veen 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, Feburary 1978

4. Interface

4.1 Calling Sequence

NOne

4.2 Subroutines Called

Subroutine	Description
KNC	SACC - Logical AND
KOC	SACC - Logical OR
KCLC	SACC - Logical compare
KMVC	SACC - logical byte move
LREAD (LWRITE)	deblocks and reads and/or writes a MJS Log record
DIRSAT	reads the MJS Log directory block for the satellite block number
REMOVE	removes a vol-ser from a TLS slot

4.3 COMMON Variables

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

4.4 I/O Devices

Unit No.	Description
05	terminal, input data
06	terminal, output messages
10	sysout A, hardcopy of data output
25	MJS Log

4.5 Input

Variable	Type	Description
QSATID(12)	L*1	satellite ID. The program only requests the satellite number, QSATID(9).
QVOL(8)	L*1	QVOL(3)-QVOL(8), the 6 digit vol-ser of the EDR tapes to be removed, one at a time.

Format

1. Satellite id card

   column 12345

   1 or 2

2. vol-ser card

   column 123456789

   AAAAAA

   vol-ser

5. Proceedure

```
     Get satellit 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

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

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 remove~~xarexperformed~~ is performed and the program will request another vol-ser.

AAAAAA[1] VOL-SER NOT FOUND FOR AAAAAAAAAAAA[2].

AAAAAAAA[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

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

4.3 COMMON Variables

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

4.4 I/O Devices

Unit No.	Description
05	terminal, input data
06	terminal, output messages
10	sysout A, hard copy of data output
25	MJS Log

4.5 Input

Variable	Type	Description
QSATID(12)	L*1	satellite ID. User only inputs satellite number, QSATID(9)
ISER	I*4	serial to be removed

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. Proceedure

```
          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 STOP
            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

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

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 RXCEEDS LAST SERIAL ASSIGNED YYY.

XXX SERIAL NOT MARKED FOR REMOVAL.

XXX SERIAL NOT FOUND.

An I/O error encounted 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 fo the MJS Log list programs.

3. Programmer and Date.

Eunice Eng, January 1978

4. Interface.

4.1 Calling Sequence.

None

4.2 Subroutines Called.

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

4.3 COMMON Variables.

The following describes the variable in COMMON FERMSG.

Variable	Type	COMMON	Description
IMES(26)	I*4	FERMSG	DAIO COMMON that displays data of an I/O error.

4.4 I/O Devices

Unit No.	Description
05	terminal, data ~~xxx~~ input
06	terminal, output messages
10	sysout A, hardcopy, data output
11	terminal or data set, data output
25	MJS Log

4.5 Input.

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

Menu number

MENUNO I*4 number associated with a list subroutine

Format

```
column 1234567890
       xx
```

Satellite ID.

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

Format

```
                11111111112
column 12345678901234567890
       voyager-X
```

The following describes input requested for theBAKLOG and INCOMP list programs.

Response card

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

~~xITYPExxxxxxxxxxxxI*4xxxxxxxxxxx~~

Format

```
column 12345
       yes
       no
```

Type card

ITYPE 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

~~XXXXX~~

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

Format

```
column 123456789012345678901234567890
       XX XX XX XX XX XX XX XX XX XX
```

The following describes input requesed for the DMPLOG subroutine.

RANGE card

IST	I*4	first block to be dumped. Must be given.
IEND	I*4	last block to be dumped. Defaults to the last block allocated, recorded in Log directory.

5. Proceedure.

```
MENU Request subroutine by menu number
     If( no more requests) EXIT to END
     If (menu = 7) go to DMPLOG
     Request satellite ID.
     If (menu no. = 1)
          Call SHWEDR, subprogram to list EDR slots marked
          for removal
          EXIT to MENU
     If (menu no = 2)
          Call SHWCIT, subprogram to list CIT slots marked for
          removal
          EXIT to MENU
     If (menu no = 3)
          Call REMSLT, subprogram to list remaining tape slots
          in Library, Work and CIT controls.
          EXIT to MENU
     IF (menu no = 4)
          If (list of back log 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 requesed)
               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 the records~~ to the last record.
          Request and read first and last records to be read
          Call DMPLOG, subroutine that gives a hexidecimal dump
          of the records in the given range.
          EXIT to MENU
     Write error message ('Wrong menu number')
     EXIT TO ~~menu~~ 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.

Argument	Type	I/O	Description
QSATID(12)	L*1	I	satellite ID
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

Subroutine	Description
KCLC	SACC - logical compare
LREAD	deblocks and reads an MJS Log record
DIRSAT	reads and returns the MJS Log directory block, also locating the satellite block address
DLCUPK	Unpacks the R*8 FDSC for the MJS satellite

4.3 COMMON Variables

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

4.4 I/O Devices

Unit No.	Description
06	terminal, output messages
10	sysout A, hardcopy, data output
11	terminal or data set, data output
25	MJS Log

5. Proceedure

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

     Read Satellite block for CONTROL block aumbers
          IEDR, EDR cnntrol 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

Variable	Type	Description
ILIB	I*4	Library Control block number
IEDR	I*4	EDR Control block number
IWRK	I*4	Work Control block number
QTODO	L*1	z80, constant that flags a ~~j~~ process requested

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~~mxxWxxk~~ or Work blocks for a given satellite, the following messages are printed respectfully and ~~xxxxxxin~~ execution continues.

NO EDR BLOCKS FOR AAAAAAAAAAAA IN LOG AT PRESENT.

NO LIBRARY BLOCKS FOR AAAAAAAAAAAA IN LOG AT PRESENT .

NO WORK BLOCKS FOR AAAAAAAAAAAA IN LOG AT PRESENT.

AAAAAAAAAAAA = satellite ID

If an I/O error occurs while reading the MJS Log the following messgge 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.

Argument	Type	I/O	Description
ICIT	I*4	I	CIT control block number
ISATCD	I*4	I	satellite code

4.2 Subroutines called

Subroutine	Description
KNC	SACC - logical AND
KOC	SACC - logical OR
KCLC	SACC - logical compare
LREAD	deblocks and reads an MJS Log record
INCORE	SACC - changes binary data to EBCDIC

4.3 COMMON Variables

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

4.4 I/O Devices

Unit No.	Description
06	terminal, output messages
10	sysout A, hardcopy of data output
11	terminal or data set, data output
25	MJS Log

5. Proceedure

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

Variable	type	Description
QREMOV	L*1	z20, constant to flag a slot marked for removal

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 subrotine then returns to the calling routine.

I/O ERROR ON LOG.

1. Subroutine DMPLOG

2. Function

DMPLOG dumps the MJS Log in hexidecimal 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 ~~XXXXXXXXXX~~ is as follows:

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

The following list describes the arguments in the calling list.

Argument	Type	I/O	Description
IST	I*4	I	first block to be dumped.
IEND	I*4	I	last block to be dumped, default to last record allocated

4.2 Subroutines Called

Subroutine	Description
LREAD	deblocks and reads a record form the MJS Log

4.3 COMMON Variables

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

4.4 I/O Devices

Unit No.	Description
06	terminal, output messages
10	sysout A, hardcopy of data output
11	terminal or data set, data output
25	MJS Log

5. Proceedure

```
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 hexidecimal 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 desription 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~~ to ~~the~~ the calling routine.

I/O ERROR ON LOG.

1. Subroutine EDRRMV

2. Function

EDRRMV displays ~~k~~ 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 DMPLOG.

3 - 6 See Subroutine DMPLOG.

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.

Argument	Type	I/O	Description
QSATID(12)	L*1	I	satellite ID
IBLOCK(10)	I*4	I	list of ~~block~~ block types to be listed

4.2 Subroutines Called

Subroutine	Description
LCIT	displays CIT records
LEDR	displays EDR records
LENC	displays Encyclopedia records
LLIB	displays Library records.
LWRK	displays Work records
LATTR	displays Attribrute blocks
LREAD	deblocks 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 Devices

Unit NO.	Description
06	terminal, output messages
10	sysout A, hardcopy of data output

5. Proceedure

Set-up truth ~~kapa~~ table which will call subroutines

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

Variable	Type	Description
QCALL(10)	L*1	logical array to call appropriate list all subroutines T = call F = do not call

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 alogn with DAIO error data passed in COMMON FERMSG. Then a ~~xxxxxxxx~~ return is taken to the ~~xxxx~~ 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 ~~arguax~~ arguments in the calling sequence ~~is desxirbed~~ are described in the list below.

Argument	Type	I/O	Description
ICCITB	I*4	I	Cit control block number
RTCODE	I*4	O	Return code = 0, normal = 8, I/O error encountered

4.2 Subroutines Called

Subroutine	Description
LREAD	deblocks 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 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. Proceedure

Reset return code, RTCODE = 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.

Argument	Type	I/O	Description
ICATTB	I*4	I	first Attribute block number
RTCODE	I*4	O	return code = 0, normal = 8, I/O error encountered

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	FERMSG	DAIO I/O error data

4.4 I/O Device

Unit No.	Description
06	terminal, output messages
10	sysout A, hardcopy of data output
11	terminal or data set, data output
25	MJS Log

5. Proceedure

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 writeen 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 FERMSG. 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.

Argument	Type	I/O	Description
QSATID(12)	L*1	I	satellite ID

4.2 Subroutines Called

Subroutine	Description
LREAD	deblocks and reads a MJS Log ~~xxxxx~~ record
DIRSAT	reads and returns the directory block. Also checks the satellite id.

4.3 COMMON Variables

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

4.4 I/O Devices

Unit No.	Description
06	terminal, output messages
10	sysout A, hardcopy, data output
11	terminal or data set, data output
25	MJS Log

5. Proceedure

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 ~~which~~ while reading the MJS Log the following message is written along with the DAIO I/O data passed through COMMON FERMSG. 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 ~~RMVCIT~~ CITRMV 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.

Argument	Type	I/O	Description
QSAITD(12)	L*1	I	satellite ID

4.2 Subroutines Called

Subroutine	Description
KCLC	SACC - logical compare
LREAD	deblocks and reads a MJS Log record
CITRMV	displays the CIT tapes marked for removal

4.3 COMMON Variables

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

4.4 I/O Device

Unit No.	Description
06	terminal, output messages
10	sysout A, hardcopy of data output

5. Proceedure

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~~ program

RETURN
Return

6. Special Notes

6.1 Major Variables and Constants

Variable	Type	Description
ICIT	I*4	Cit Control block number
ISATCD	I*4	satellite code passed to subroutine CITRMV to help build vol-ser
QSATCD(4)	L*1	equivalent to ISATCD

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 = AAAAAAAAAAAA INVALID.

If an I/O error is encountered while reading ~~zbzMz~~ 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~~ actual listing.
The subroutine is similar to SHWCIT.

3. - 6.3 see Subroutine SHWCIT

~~ALTBLK~~
INTLOG

User's guide

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

2) Alter data COMMON BLOCKS if nessary

3) Compile, link, load and run INTLOG

a) Time est. 100 400

```
ALTBLK
*
     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.
END OF DATA
```

'ZBEKE. LIB. CLIST (ALTBLK)'

```
00010 PROC 0
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
```

```
00010 ************************************************************
 0020   LOGTLS - MJS LOG  and TLS UPDATE PROGRAMS
00030     There are four functions in the MJS Log and TLS Update
00040     system.  At present they are expected to run on TSO
00050     real-time.
00060 *
00070       I.  Function.
00080           A.  RMVEDR - remove EDR tapes marked for removal,
00090                        one at a time from MJS Log and TLS.
00100           B.  RMVWRK - remove Work tapes with CIT and Merge
00110                        processes marked done from TLS, one
00120                        user specified serial at a time.
00130           C.  RMVCIT - remove CIT tapes, within a user given
00140                        range from Log and TLS.
00150           D.  ASSIGN - assign Work or CIT tapes, up to a user
00160                        specified maximum (LE 5) to fill
00170                        unassigned slots allotted in TLS to the
00180                        given MJS process.
00190           E.  Backup - all above functions give the user the option
00200                        of backing up the MJS Log 'SBMJS.LOG.DATA'
00210                        onto a one day scratch data set 'ZBEKE.LOG.
00220                        DATA'.  If a backup does not exist in 'ZBEKE.
00230                        LOG.DATA', one will be done automatically
00240                        once per day of use.
00250           F.  Hardcopy - all above functions also provide the user
00260                        with a hard copy of his/her session.
00270      II.  Restrictions.
00280           A.  These programs must be run using user id = SBMJS, since
00290               these tapes are TLS userid protected.
00300           B.  The user must use TSO with the following region
00310               requirements:
00320                    91  any region (only one region size at present)
00330                    75  large region size(192), meaning 192k
00340           C.  One satellite is processed at a time, except for ASSIGN.
00350           D.  The user should know how to use command tlsupdte on TSO
00360               incase of fixups.
00370     III.  Syntax to invoke routines.
00380           A.  To run RMVEDR:
00390                    ex 'ZBEKE.LIB.CLIST(RMVEDR)'
00400           B.  To run RMVWRK:
00410                    ex 'ZBEKE.LIB.CLIST(RMVWRK)'
00420           C.  To run RMVCIT:
00430                    ex 'ZBEKE.LIB.CLIST(RMVCIT)'
00440           D.  To run ASSIGN:
00450                    ex 'ZBEKE.LIB.CLIST(ASSIGN)'
00460           E.  To close data sets incase of an 80A abend (see error)
00470                    ex 'ZBEKE.LIB.CLIST(CLOGTLS)'
00480           F.  To list Log:
00490                    ex 'ZBEKE.LIB.CLIST(LISTALL)'
00500                         or background batch
```

```
00510                    stab YYY,t(h00h00)
00520                    job card
00530                    =:'zbeke.lib.cntl(listall)'
00540                    // exec notifyts
00550                    endinput
00560          G.  To invoke TLS update programs only:
00570                    tlsupdte
00580     IV.  Error Conditions.
00590          A.  General - If the execution terminates abnormally, i.e.
00600                        system 80A, execute 'ZBEKE.LIB.CLIST(CLOGTLS)'
00610                        to close data sets properly.  Then list the
00620                        Log using one of the above mentioned methods,
00630                        to check the state of the Log.  The user must
00640                        then decide what kind of fixups are needed if
00650                        any.
00660          B.  RMVEDR:
00670              1.  XXXXXX   NOT MARKED FOR REMOVAL.
00680              Response:
00690                  a.  check typing
00700                  b.  if vol-ser was entered twice, only the last
00710                      entered will get marked.  Must contact offical
00720                      person to alter the MJS Log manually to get to
00730                      the first entered vol-ser.
00740                  c.  continue
00750              2.  I/O ERROR ENCOUNTERED ON LOG.
00760              Response
00770                  a.  try listing Log
00780                  b.  save messge and following lines of data to help
00790                      when consulting the SACC DAIO manual.
00800          C.  RMVWRK:
00810              1.  SERIAL REQUESTED FOR REMOVAL XXX EXCEEDS LAST SERIAL
00820                  ASSIGNED YYY.
00830              Response:
00840                  a.  check typing
00850                  b.  get latest Log listing
00860                  c.  continue
00870              2.  XXX SERIAL NOT MARKED FOR REMOVAL.
00880              Response:
00890                  a.  check typing
00900                  b.  get latest Log listing
00910                  c.  continue
00920              3.  XXX SERIAL NOT FOUND.
00930              Response:
00940                  a.  check typing
00950                  b.  continue
00960              4.  I/O ERROR ENCOUNTED ON LOG.
00970              Response:  See RMVEDR above
00980          D.  RMVCIT:
00990              1.  ISER1 = XXX IS GREATER THAN ISERN = YYY.
01000              Response:
```

```
01010                   a.  Check typing.  ISER1 = first serial,
01020                       ISERN = last serial.
01030                   b.  restart CLIST to continue
01040               2.  FIRST MUST BE ODD.
01050               Response:
01060                   a.  restart CLIST to continue.  This time
01070                       first serial must be odd.
01080               3.  LAST MUST BE EVEN.
01090               Response:
01100                   a.  restart CLIST to continue.  This time
01110                       last serial must be even.
01120               4.  ***** NONE OF THE SERIALS ARE ASSIGNED TO A TLS SLOT.
01130               Response:
01140                   a.  check latest Log listing.  No more removals
01150                       can be done.
01160               5.  THE SERIALS IN THE REMOVE REQUEST IS OUT OF RANGE
01170                   (XXX to YYY).
01180                   XXX first log assigned serial
01190                   YYY last log assigned serial
01200               Response:
01210                   a.  check latest Log listing
01220                   b.  restart CLIST to continue
01230               6.  I/O ERROR ENCOUNTED ON LOG.
01240               Response:
01250                   see RMVEDR above
01260           E.  ASSIGN:
01270               1.  MAX GREATER THAN 5.
01280               Response:
01290                   a.  check typing
01300                   b.  restart CLIST to continue
01310               2.  INVALID QTYPE - AAAA - CHECK INPUT.
01320               Response:
01330                   a.  check typing
01340                   b.  restart CLIST to continue
01350               3.  LAST SERIAL = XXX NOW EXCEEDS 999.  UPDATING CONTROL
01360                   AND TERMINATING.
01370               Response:
01380                   a.  the last serial assigned was 999.  Must
01390                       allocate and initialize the next sequential
01400                       Log.
01410               4.  I/O ERROR ENCOUNTED ON LOG.
01420               Response:  See RMVEDR above.
01430       V.  Load Modules:
01440           A.  RMVEDR - 'ZBEKE.VOYAGER.LOAD(REDR)'
01450           B.  RMVWRK - 'ZBEKE.VOYAGER.LOAD(RWORK)'
01460           C.  RMVCIT - 'ZBEKE.VOYAGER.LOAD(RCIT)'
01470           D.  ASSIGN - 'ZBEKE.VOYAGER.LOAD(ASSIGN)'
01480           E.  LISTALL - 'ZBEKE.LIST.LOAD(LISTER)'
01490      VI.  TSO CLISTS:
01500           A.  RMVEDR - 'ZBEKE.LIB.CLIST(RMVEDR)'
```

```
01510            B.  RMVWRK - 'ZBEKE.LIB.CLIST(RMVWRK)'
01520            C.  RMVCIT - 'ZBEKE.LIB.CLIST(RMVCIT)'
01530            D.  ASSIGN - 'ZBEKE.LIB.CLIST(ASSIGN)'
01540            E.  CLOGTLS - 'ZBEKE.LIB.CLIST(CLOGTLS)'
01550            F.  LISTALL - 'ZBEKE.LIB.CLIST(LISTALL)'
01560      VII.  Background PROC:
01570            A.  LISTALL - 'ZBEKE.LIB.CNTL(LISTALL)'
```

RMVEDR

```
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) 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(F 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
```

RMVWRK

```
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) SYSOUT
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.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(F 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(RWORK)'
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
```

RMVCIT

```
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) 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(F 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(RCIT)'
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
```

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.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) 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(F 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 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(SYSIN,SYSPRINT,VSN,PVTLIBDD,SLOT)
00530 ALLOC F(SYSIN) 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

```
00010   Lister describes the list routines of the Voyasr Automatic
        Log.
00030 *
00040      I.  Function:
00050          List all or parts of the Voyaser Automatic Log.
00060          A hard copy of the TSO session is returned to the user.
00070 *
00080     II.  Restrictions:
00090          It is highly recommended that the user submit a backsound
00100          job when requesting a list all type list of the entire
00110          Voyaser Log.  (Use 'ZBEKE.LIB.CNTL.(LISTAL)')
00120 *
00130    III.  Syntax:
00140          To start a TSO, selective list of the Log.
00150                ex 'zbeke.lib.clist(lister)' 'log(''sbmJs.log.data'') -
00160                    out11(*)'
00170            A.  log = Voyaser 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
          To terminate session type in
              /*
              when prompted for another menu number.
00290          To submit a backsround batch job from TSO, the
00300              following job stream is nessary
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 listing 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
```

```
00470                1.  NO EDR BLOCKS FOR AAAAAAAAAAAA IN LOG AT PRESENT.
00480                    NO LIBRARY BLOCKS FOR AAAAAAAAAAAA IN LOG AT PRESENT.
00490                    NO WORK BLOCKS FOR AAAAAAAAAAAA IN LOG AT PRESENT.
00500                    where AAAAAAAAAAAA = satellite id
00510                    Check typing.  Program will prompt for next menu
00520                    number.
00530                2.  SATELLITE ID, QSATID = AAAAAAAAAAAA IS INVALID.
00540                    Check typing.  Program will prompt for next menu
00550                    number
00560                3.  I/O ERROR ON LOG.
00570                    Check Log.  Program will terminate.
00580            C.  DMPLOG
00590                1.  I/O ERROR ON LOG.
00600                    see BAKLOG
00610            D.  SHWCIT, display CIT tapes marked for removal.
00620                1.  I/O ERROR ON LOG.
00630                    see BAKLOG
00640                2.  SATELLITE ID, QSATID = AAAAAAAAAAAA IS INVALID.
00650                    see BAKLOG
00660            E.  SHWEDR, display EDR tapes marked for removal.
00670                1.  I/O ERROR ON LOG.
00680                    see BAKLOG
00690                2.  SATELLITE ID, QSATID = AAAAAAAAAAAA IS INVALID.

  00                     see BAKLOG
00710            F.  INCOMP
00720                similar to BAKLOG, see BAKLOG.
00730            G.  LISTAL
00740                1.  I/O error on LISTAL or any of its Subroutines
00750                    see BAKLOG
00760            H.  REMSLT
00770                1.  I/O ERROR ON LOG
00780                    see BAKLOG
00790                2.  SATELLITE ID, QSATID = AAAAAAAAAAAA IS INVALID.
00800                    see BAKLOG
00810 *
00820       V.   Load Modules
00830            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
```

'ZBEKE.LIB.CLIST (LISTER)'

```
00010 PROC 0 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       SHWEDR     DISPLAYS EDR TAPES MARKED FOR REMOVAL'

00240 TPRINT '    02       SHWCIT     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       INCOMP     INCOMPLETE PROCESSES'
00262 TPRINT '    06       LISTAL     LISTALL 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)
END OF DATA
```

User's Guide to log list programs LISTER

The following is a Clist (of TSO) used to invoke the list routines. The program is interactive with the user at the terminal.

CLIST location: ZBEKE.LIB.CLIST(LISTER)

```
FREE O LOG('SBMJS.LOG.DATA') OUT11(*)
CFREE F(FT05F001,FT06F001,FT10F001,FT11F001,FT25F001)
CFREE ATTR(OUT,LOG)
ATTR OUT BLKSIZE(2400) LRECL(120) RECFM(F B)
ATTR LOG BLKSIZE(7232) LRECL(7232) RECFM(F)
ALLOC F(FT05F001) DA(*)
ALLOC F(FT06F001) DA(*)
ALLOC F(FT10F001) SYSOUT
ASK 'DID YOU NAME A DSN FOR OUT11?'
SYSRC (EQ 4) GO TO LABEL TERM
SRCHCS &OUT11
WHEN SYSRC(LE 4)
ALLOC DA(&OUT11.) NEW SPACE(60) TRACKS
ALLOC F(FT11F001) DA(&OUT11.) USING(OUT) MOD
GO TO LABEL FT25
LABEL TERM
ALLOC F(FT11F001) DA(&OUT11.)
LABEL FT25
ALLOC F(FT25F001) DA(&LOG.) USING(LOG) OLD
ASK 'DO YOU WANT TO SEE THE LIST MENU?'
SYSRC (EQ 4) GO TO LABEL CALL
TPRINT 'MENU NO.     NAME       DESCRIPTION'
TPRINT '    01       SHWEDR     DISPLAYS EDR TAPES MARKED FOR REMOVAL'
TPRINT '    02       SHWCIT     DISPLAYS CIT TAPES MARKED FOR REMOVAL'
TPRINT '    03       REMSLT     # TAPE SERIALS REMAINING IN SOME CONTROLS'
TPRINT '    04       BAKLOG     BACK LOG ON A GIVEN PROCESS'
TPRINT '    05       INCOMP     INCOMPLETE PROCESSES'
TPRINT '    06       LISTAL     LISTALL BY BLOCKS'
TPRINT '    07       DMPLOG     HEX DUMP OF THE LOG'
LABEL CALL
CALL 'ZBEKE.LIST.LOAD(LISTER)'
FREE F(FT10F001) SYSOUT(A)
TPRINT 'OUPUT TO FT11F001 ON &OUT11.'
FREE F(FT25F001)
```

ex 'ZBEKE.LIB.CLIST(LISTER)' 'LOG("____.LOG._____") OUT11(" .LIST.DATA

Note LOG( ) gives the dsn of the log to be listed default is set to 'SBMJS.LOG.DATA'

OUT11( ) gives the dsn of a data output, ~~temporary~~ scratch data set. This is to be used if large output is expected. Else the default is set to the terminal.

Logical Units

Unit No.	Description
5	Terminal, data input
6	Terminal, prompter and error messages
10	Sysout (a), hard copy of session
11	Terminal or Data Set, data output
25	Log

Load Module Location:

'ZBEKE.LIB.CNTL(LISTALL)'

```
00010 //* 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=SBMJS.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
```

```
*************************************************************
*  BIOCKS IS A BLOCK DATA USED TO INTIALIZE THE LOG
*
*************************************************************
*
BIOCKS    CSECT
*                    ***** BLOCK 1 ***** DIRECTORY *****
QBLK      DC    F'0',F'0'          BLOCK ID CODE = 0
          DC    H'5650'            # OF BLOCKS IN DATA SET
          DC    H'15'              LAST BLOCK ALLOCATED
          DC    H'2'               # OF SATELLITES
          DC    H'3'               MAX. # OF SATELLITES
*                              ---------------------------
          DC    X'01'              S/C CODE
          DC    X'09'              # OF CHAR. IN ID
          DC    CL12'VOYAGER-1'    S/C ID
          DC    H'2'               @ OF SATELLITE BLOCK
*                              ---------------------------
          DC    X'00'              S/C CODE
          DC    X'09'              # OF CHAR. IN ID.
          DC    CL12'VOYAGER-2'    S/C ID.
          DC    H'9'               @ OF SATELLITE BLOCK
*                              ---------------------------
          DC    8H'0'              SPARE
*
```

```
*                    ***** BLOCK 2 ***** SATELLITE -1 *****
*
         DC    X'01'                 BLOCK ID. CODE
         DC    7XL1'00'
         DC    H'3'                  @ OF 1ST ENCY-ATTRIBUTE BLOCK
         DC    H'4'                  @ OF EDR CONTROL
         DC    H'5'                  @ OF LIBRARY CONTROL
         DC    H'6'                  @ OF WORK CONTROL
         DC    H'7'                  @ OF ENCY CONTROL
         DC    H'8'                  @ OF CITENCY CONTROL
         DC    11F'0'
*
*                    ***** BLOCK 3 ***** ENCY ATTR 1 *****
*
         DC    X'02'                 BLOCK IDENTIFIER CODE
         DC    7XL1'00'
         DC    H'900'                RECORD VOLUME LEN IN SEC.
         DC    X'0C'                 DQA ON MASK
         DC    X'00'                 DQA OFF MASK
         DC    X'FFFFFFF0'           VERSE PRESENCE MASK
         DC    H'0'                  @ OF 1ST WORK BLOCK
         DC    H'0'                  @ PF 1ST ENCY BLOCK
         DC    CL4'    '             4 CHAR MNEMONIC
         DC    H'0'                  @ OF LAST WORK BLOCK
         DC    H'14'                 # OF CHAR IN ENCY DSN
         DC    CL18'V1.MASTER.ENCY'    ENCY DSN
         DC    H'0'                  @ OF LAST ENCY BLOCK
         DC    4F'0'
*
*                    ***** BLOCK 4 ***** EDR CONTROL 1 *****
*
         DC    X'03'                 BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                  @ OF FIRST EDR BLOCK
         DC    H'0'                  @ OF LAST EDR BLOCK
         DC    H'0'                  @ OF LAST EDR BLOCK MODIFIED
         DC    H'0'                    SPARE
         DC    F'64590'              1ST EDR SLOT
         DC    F'64614'              LAST EDR SLOT
         DC    10F'0'
*
*                    ***** BLOCK 5 ***** LIBRARY CONTROL 1 *****
*
         DC    X'04'                 BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                  @ OF FIRST LIBRARY BLOCK
         DC    H'0'                  @ OF LAST LIBRARY BLOCK
         DC    H'0'                  @ OF LAST LIBRARY BLOCK MODIFIED
         DC    H'0'                    SPARE
         DC    H'1'                  1ST LIBRARY SERIAL
         DC    H'50'                 LAST LIBRARY SERIAL
         DC    F'60640'              1ST LIBRARY SLOT
         DC    F'60689'              LAST LIBRARY SLOT
         DC    H'1'                  LAST VOL-SER WRITTEN ON
         DC    H'0'                  LAST DATA FILE SEQUENCE
         DC    H'0'                  AMT. OF FEET USED ON VOL-SER
         DC    H'1800'               MAX. AMT. OF FEET TO BE USED/TAPE
         DC    CL18'V1.'             MODEL DSN FOR LIBRARY TAPES
         DC    CL6'M1L000'           MODEL VOL-SER
         DC    F'0'
*
*                    *** BLOCK 6 ***** WORK CONTROL 1 *****
*
         DC    X'05'                 BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                  @ OF FIRST WORK BLOCK
         DC    H'0'                  @ OF LAST WORK BLOCK
         DC    H'1'                  1ST WORK SERIAL
         DC    H'999'                LAST WORK SERIAL
         DC    F'60410'              1ST WORK SLOT
         DC    F'60434'              LAST WORK SLOT
         DC    CL18'V1.WORK.ENCY'      WORK DSN
         DC    CL6'M1W000'           MODEL VOL-SER
         DC    H'0'                  @ OF PREF WORK VOLUME
         DC    7H'0'
*
*                    ***** BLOCK 7 ***** ENCY CONTROL 1 *****
*
```

```
         DC    X'06'                BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                 @ OF 1ST ENCY BLOCK
         DC    H'0'                 @ OF LAST ENCY BLOCK
         DC    H'1'                 1ST ENCY SERIAL
         DC    H'150'               LAST ENCY SERIAL
         DC    F'60690'             1ST ENCY SLOT
         DC    F'60719'             LAST ENCY SLOT
         DC    CL6'M1E000'          MODEL VOL-SER
         DC    H'0'                 @ OF FREE ENCY VOLUME
         DC    H'2000'              MAX. AMT. OF FEET TO BE USED/TAPE
         DC    H'0'
         DC    7F'0'
*
*                 ***** BLOCK 8 ***** CITENCY CONTROL 1 *****
*
         DC    X'07'                BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                 @ OF FIRST CITENCY BLOCK
         DC    H'0'                 @ OF LAST CITENCY BLOCK
         DC    H'1'                 1ST CITENCY SERIAL
         DC    H'999'               LAST CITENCY SERIAL
         DC    F'60360'             1ST CITENCY SLOT
         DC    F'60384'             LAST CITENCY SLOT
         DC    CL6'M1C000'          VOL-SER MODEL
         DC    H'0'                 @ OF FREE CITENCY VOLUME
         DC    H'2000'              MAX AMT OF FEET/TAPE
         DC    CL18'V1000000. 000000'    MODEL DSN
         DC    3F'0'
*
*                 ***** BLOCK 9 ***** SATELLITE -2 *****
*
         DC    X'01'                BLOCK ID. CODE
         DC    7XL1'00'
         DC    H'10'                @ OF 1ST ENCY-ATTRIBUTE BLOCK
         DC    H'11'                @ OF EDR CONTROL
         DC    H'12'                @ OF LIBRARY CONTROL
         DC    H'13'                @ OF WORK CONTROL
         DC    H'14'                @ OF ENCY CONTROL
         DC    H'15'                @ OF CITENCY CONTROL
         DC    11F'0'
*
*                 ***** BLOCK 10 ***** ENCY-ATTR 2 *****
*
         DC    X'02'                BLOCK IDENTIFIER CODE
         DC    7XL1'00'
         DC    H'900'               RECORD VOLUME LEN IN SEC.
         DC    X'00'                DQA ON MASK
         DC    X'00'                DQA OFF MASK
         DC    X'FFFFFFF0'          VERSE PRESENCE MASK
         DC    H'0'                 @ OF 1ST WORK BLOCK
         DC    H'0'                 @ PF 1ST ENCY BLOCK
         DC    CL4'    '            4 CHAR MNEMONIC
         DC    H'0'                 @ OF LAST WORK BLOCK
         DC    H'14'                # OF CHAR IN ENCY DSN
         DC    CL18'V2.MASTER.ENCY'     ENCY DSN
         DC    H'0'                 @ OF LAST ENCY BLOCK
         DC    4F'0'
*
*                 ***** BLOCK 11 ***** EDR CONTROL 2 *****
*
         DC    X'03'                BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                 @ OF FIRST EDR BLOCK
         DC    H'0'                 @ OF LAST EDR BLOCK
         DC    H'0'                 @ OF LAST EDR BLOCK MODIFIED
         DC    H'0'                   SPARE
         DC    F'64615'             1ST EDR SLOT
         DC    F'64639'             LAST EDR SLOT
         DC    10F'0'
*
*                 ***** BLOCK 12 ***** LIBRARY CONTROL 2 *****
*
         DC    X'04'                BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                 @ OF FIRST LIBRARY BLOCK
         DC    H'0'                 @ OF LAST LIBRARY BLOCK
         DC    H'0'                 @ OF LAST LIBRARY BLOCK MODIFIED
```

```
         DC    H'0'                    SPARE
         DC    H'1'                 1ST LIBRARY SERIAL
         DC    H'50'                LAST LIBRARY SERIAL
         DC    F'60840'             1ST LIBRARY SLOT
         DC    F'60889'             LAST LIBRARY SLOT
         DC    H'1'                 LAST VOL-SER WRITTEN ON
         DC    H'0'                 LAST DATA FILE SEQUENCE
         DC    H'0'                 AMT. OF FEET USED ON VOL-SER
         DC    H'1800'              MAX. AMT. OF FEET TO BE USED/TAPE
         DC    CL18'V2.'            MODEL DSN FOR LIBRARY TAPES
         DC    CL6'M2L000'          MODEL VOL-SER
         DC    F'0'
*
*                   *** BLOCK 13 ***** WORK CONTROL 2 *****
*
         DC    X'05'                BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                 @ OF FIRST WORK BLOCK
         DC    H'0'                 @ OF LAST WORK BLOCK
         DC    H'1'                 1ST WORK SERIAL
         DC    H'999'               LAST WORK SERIAL
         DC    F'60435'             1ST WORK SLOT
         DC    F'60459'             LAST WORK SLOT
         DC    CL18'V2.WORK.ENCY'      WORK DSN
         DC    CL6'M2W000'          MODEL VOL-SER
         DC    H'0'                 @ OF FREE WORK VOLUME
         DC    7H'0'
*
*                   ***** BLOCK 14 ***** ENCY CONTROL 2 *****
*
         DC    X'06'                BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                 @ OF 1ST ENCY BLOCK
         DC    H'0'                 @ OF LAST ENCY BLOCK
         DC    H'1'                 1ST ENCY SERIAL
         DC    H'150'               LAST ENCY SERIAL
         DC    F'64890'             1ST ENCY SLOT
         DC    F'65039'             LAST ENCY SLOT
         DC    CL6'M2E000'          MODEL VOL-SER
         DC    H'0'                 @ OF FREE ENCY VOLUME
         DC    H'2000'              MAX. AMT. OF FEET TO BE USED/TAPE
         DC    H'0'
         DC    7F'0'
*
*                   ***** BLOCK 15 ***** CITENCY CONTROL 2 *****
*
         DC    X'07'                BLOCK ID CODE
         DC    7XL1'00'
         DC    H'0'                 @ OF FIRST CITENCY BLOCK
         DC    H'0'                 @ OF LAST CITENCY BLOCK
         DC    H'1'                 1ST CITENCY SERIAL
         DC    H'999'               LAST CITENCY SERIAL
         DC    F'60385'             1ST CITENCY SLOT
         DC    F'60409'             LAST CITENCY SLOT
         DC    CL6'M2C000'          VOL-SER MODEL
         DC    H'0'                 @ OF FREE CITENCY VOLUME
         DC    H'2000'              MAX AMT OF FEET/TAPE
         DC    CL18'V2000000. 000000'     MODEL DSN
         DC    3F'0'
*
         END
```

*** END OF MEMBER *** 294 RECORDS PROCESSED **********

COMMON/LOADAT/

VARIABLE	Type	Lenth	Disp	
QSATID (12)	L*1	12	0	Satellite id., (alphanumeric)
ISATCD	I*4	4	12	Satellite code
EDRSER (6)	L*1	6	16	EDR vol-ser = XXXXXX, jpl number
EDRDSN (6)	L*1	6	22	EDR DSN = CRS
IEDRSL	I*4	4	28	EDR slot
NRREDR	I*4	4	32	No. EDR records
NEREDR	I*4	4	36	No. EDR errors
IEDRBK	I*4	4	40	EDR block address
HEDRCH	I*2	2	44	2 char. of EDR, JPL-ser
HLBSER	I*2	2	46	Lib. vol-ser (binary)
DSTFDS	R*8	8	48	Start FDSC
DENFDS	R*8	8	56	End FDSC
LIBSER (6)	L*1	6	64	Lib. vol-ser (EBCDIC) = M(1/2)LXXX
LIB DSN (18)	L*1	18	70	Lib. DSN (EBCDIC) = V(A/B).AAXXXXXX EDR VOL-SER
I1SLOT	I*4	4	88	Lib. 1st slot
INSLOT	I*4	4	92	Lib. Last slot
ILIBBK	I*4	4	96	Lib. block address
HLIBSQ	I*2	2	100	Lib. tape file sequence
HFEET	I*2	2	102	amt. of feet used on present tape
HMXFT	I*2	2	104	max. feet on tape
HWKSER	I*2	2	106	Work VOL-SER (binary)
IATTBK	I*4	4	108	ENCY-ATTR block ~~address~~
IWRKBK	I*4	4	112	WORK block address
WRKDSN (18)	L*1	18	116	WORK DSN, (EBCDIC) = V{A/B}.
WRKSER (6)	L*1	6	134	WORK Vol-Ser, (EBCDIC) = M(1/2)WXXX

Variable	Type	Length	Disp	
I1WSLT	I*4	4	140	1st WORK slot
INWSLT	I*4	4	144	last WORK slot
I1RVOL	I*4	4	148	1st record volume number on work block
INRVOL	I*4	4	152	last record volume number on work block
			156	

MJS LOG

The Log is basically of two parts, the control blocks and the blocks themselves. Into all blocks is build a forward and backward link.

THE CONTROL BLOCKS:

The directory block notes 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 blocks for a given satellite.

The attribute blocks describe the conditions under which the ENCBEN procedure was run.

The EDR, Library, Work, Ency and CIT Ency control blocks basically keep track of the first and last assigned blocks and range of tape slots and serials. In the case of ~~EDR~~ Library, Work, Ency and CIT Ency tape limits are also described.

The Blocks

The EOR, Library, Work, Ency. and CIT Ency blocks maintain forward and backward links and processing status and define the data covered and ~~their~~ tape location.

The following Tables ~~should~~ will give a more detailed description of the fields inside a Log block.

MJS Log

TYPES of BLOCKS and CODE ASSIGNMENT

Block Identifier Code	Description
00	Directory Block
01	Satellite block
02	Ency-Attribute block
03	EDR Control block
04	Library Control block
05	Work Control block
06	Encyclopedia Control Block
07	CITENCY Control block
13	EDR block
14	Library block
15	Work block
16	Encyclopedia block
17	CITENCY block

Structure of Processing Disposition Byte

type	Description			
0	marked for processing	80		
1	processing has begun		E0	E1
2	processing was completed			
3	spare			
4-7	processing completion code =0 no errors =1 errors dectected =15 serious error			

Structure of Slot Allocation Status Byte

Bit	Description			
0	slot allocation requested	C0		
1	slot allocated		E0 complete	F0 removed
2	slot is removable			
3-7	spare			

* bits are logically ORed on.

DIRECTORY BLOCK

Byte	Byte Length	Description
0	1	block identifier code, = 0
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type, = 0
6	2	Address of next block of this type, = 0
8	2	Number of blocks in data set
10	2	Address of last block allocated
12	2	Number of satellite entries
14	2	Maximum number of satellite entries, = 3
16	1	S/C id. code
17	1	Number of characters in satellite id.
18	12	Satellite id. (alphanumeric)
30	2	Address of Satellite block for this satellite
32-47	16	Same as 16-31 above for second satellite
48-63	16	Same as 16-31 above for third satellite

SATELLITE BLOCK

Byte	Byte Length	Description
0	1	Block identifier code, = 1
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type, = 0
6	2	Address of next block of this type, = 0
8	2	Address of first Ency-Attribute block
10	2	Address of EDR Control block
12	2	Address of Library Control block
14	2	Address of Work Control block
16	2	Address of Encyclopedia Control block
18	2	Address of CITENCY Control block

ENCY-ATTRIBUTE BLOCK

Byte	Byte Length	Description
0	1	block identifier code, = 2
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type
6	2	Address of next block of this type
8	2	Record volume length in seconds
10	1	Data Quality Acceptance ON mask
11	1	Data Quality Acceptance OFF mask
12	4	Verse presence mask
16	2	Address of first work block
18	2	Address of first encyclopedia block
20	4	4 Character mnemonic for this encyclopedia
24	2	Address of last work block
26	2	unused
28	4	First encyclopedia volume
32	4	Last encyclopedia volume
36	2	Last encyclopedia block
38	2	Number of characters in title *
40	–	Character title of Encyclopedia *

* not actively used at present

EDR CONTROL BLOCK

Byte	Byte Length	Description
0	1	block identifier code, = 3
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type, = 0
6	2	Address of next block of this type, = 0
8	2	Address of first EDR block
10	2	Address of last EDR block
12	2	Address of EDR block last modified
14	2	
16	4 6	First slot assigned to EDR's
20	4 6	Last slot assigned to EDR's

Byte	Byte Length	Description
0	1	block identifier code, = 4
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type, = 0
6	2	address of next block of this type, = 0
8	2	address of first library block
~~10~~	~~2~~	~~address of last library block~~
12	2	address of last library block modified
14	2	
16	2	first library volume serial
18	2	last library volume serial
20	4	first library slot
24	4	last library slot
28	2	last library volume serial written on
30	2	last data set sequence number on above
32	2	feet used on volume corresp. to last library serial above
34	2	Maximum amount of tape to be written (in feet) = 1800 feet
36	18	Model volume DSN (17 char) $V\binom{A}{B}$.AAXXXXXX (2 JPL characters; JPL number)
54	6	Model volume serial = $M\binom{1}{2}$WØØØ

Byte	Byte Length	Description
0	1	block identifier code, = 5
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type, = 0
6	2	Address of next block of this type, = 0
8	2	Address of 1st work block
10	2	Address of last work block
12	2	first work volume serial
14	2	last work volume serial
16	4	first work slot
20	4	last work slot
24	18	Model volume DSN (17 char)
		V(A/B).
42	6	Model volume serial
		M(1/2)WØØØ
48	2	LAST SERIAL USED

ENCYCLOPEDIA CONTROL BLOCK

Byte	Byte Length	Description
0	1	block identifier code, = 6
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type, = 0
6	2	Address of next block of this type, = 0
8	2	Address of 1st Encylopedia block
10	2	Address of last Encyclopedia block
12	2	first encyclopedia volume serial
14	2	last encyclopedia volume serial
16	4	first encyclopedia slot
20	4	last encyclopedia slot
24	6	Model Volume name (6 char)
30	2	LAST ENCY SERIAL USED
32	2	MAXIMUM NUMBER OF BLOCKS
34	18	MODEL VOLUME DSN (18 CHAR)

Byte	Byte Length	Description
0	1	block identifier code, = 7
1	1	year of creation
2	1	month of creation
3	1	day of creation
4	2	Address of previous block of this type, = 0
6	2	Address of next block of this type, = 0
8	2	Address of first CITENCY block
10	2	Address of last CITENCY block
12	2	First CITENCY volume serial
14	2	Last CITENCY volume serial
16	4	First CITENCY slot
20	4	Last CITENCY slot
24	6	Model volume name (6 char)
30	2	Address of free CITENCY volume block
32	2	Maximum amount of tape to be written on

EDR BLOCK

Byte	Byte Length	Description
0	1	block identifier code, = 13
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type
6	2	Address of next block of this type
8	6	Volume Serial (JPL number)
14	2	2 Characters in JPL number
16	1	Slot allocation status
17	1	Year received
18	1	Month received
19	1	Day received
20	4	Slot number
24	2	Address of Library entry
26	2	Number of records
28	2	Number of errors
30	2	maximum # of files
32	2	
34	2	Number of entries that follow that were processed
36	1	Processing disposition
37	1	Year of processing
38	1	Month of processing
39	1	Day of processing
40-43	4	Same as 36-39 above
44-47	4	Same as 36-39 above
48-51	4	Same as 36-39 above

LIBRARY BLOCK

Byte	Byte Length	Description
0	1	block identifier code, = 14
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type
6	2	Address of next block of this type
8	8 (R*8)	~~Start FDSC~~ Start seconds since Jan 1, 1977
16	8 (R*8)	~~End FDSC~~ End seconds since Jan 1, 1977
24	2	Library Tape volume serial
26	2	Library tape file sequence
28	2	Address of EDR block
30	2	Number of entries to follow that processing was completed
32	1	Processing disposition
33	1	Year of processing
34	1	Month of processing
35	1	Day of processing
36	2	Address of Ency-Attribute block
38	2	Address of work block
40-47	8	Same as 32-39 above
48-55	8	Same as 32- 39 above
56-63	8	Same as 32-39 above

WORK BLOCK

Byte	Byte Length	Description
0	1	block identifier code, = 15
1	1	Year of creation
2	1	Month of creation
3	1	Day of creation
4	2	Address of previous block of this type,
6	2	Address of next block of this type
8	2	Address of Ency-Attribute block
10	2	Address of work block of same Ency-Attribute
12	2	Address of next work block of same attribute
14	2	Number of library blocks
16	4	Start record volume number
20	4	End record volume number
24	1	Merge processing disposition
25	1	Spare
26	2	Address of Encyclopedia block
28	1	CITENCY processing disposition
29	1	spare
30	2	Address of CITENCY block
32	~~32~~ 309-17	Address of library blocks corresponding to byte 14 above, each address takes 2 bytes
62	2	work tape serial number

Byte	Byte Length	Description
0	1	block identifier code, = 16
1	1	year of creation
2	1	month of creation
3	1	day of creation
4	2	Address of previous block of this type
6	2	Address of next block of this type
8	2	Address of attribute block
10	2	Address of previous encyclopedia block for same Ency-Attribute
12	2	Address of next encyclopedia block for same Ency-Attribute
14	2	Current Block count for this serial *
16	4	Start record volume number
20	4	End record volume number
24	2	Volume serial of encyclopedia tape
26	2	@ 1 of this Ency block *
28 + 2 = (29)	1	Status of this block (00 = active 01 = dead) *
30 (Byte 31, 32)	2	slot number freed by RMVENC
31	4	cartridge slot # freed by RMVENC (MOD 1/89 KAW)

* documentation added 11/10/78 EWR

CITENCY BLOCK

Byte	Byte Length	Description
0	1	block identifier code, = 17
1	1	year of creation
2	1	month of creation
3	1	day of creation
4	2	Address of previous block of this type
6	2	Address of next block of this type
8	2	Volume serial of 'raw verse only' encyclopedia
10	2	Volume serial of 'all but raw rates' encyclopedia
12	4	Slot of 'raw verse only'
16	4	Slot of 'all but raw rates'
20	1	slot allocation status
21	1	year sent
22	1	month sent
23	1	day sent
24	3	Date acknowledged, year month day
27	1	spare
28	2	Address of work block