File Number S360-24
GC24-5039-1

Chicago
Grant Boyce

DOS

IBM Systems Reference Library

IBM System/360 Disk Operating System COBOL DASD Macros

This reference publication provides the programmer with rules for using DASD macros to handle input/output operations for direct access or indexed sequential files. The programmer should be familiar with:

1. COBOL: General Information Manual, Form F28-8053.

2. IBM System/360 Disk and Tape Operating Systems, COBOL Language Specifications, Form C24-3433.

3. IBM System/360 Disk Operating System, Data Management Concepts, Form C24-3427.

4. IBM System/360 Disk Operating System, Supervisor and Input/Output Macros, Form C24-5037.

Other related IBM publications are referenced by form number and briefly described in IBM System/360 Bibliography, Form A22-6822.

Second Edition (December 1966)

This is a reprint of C24-5039-0
incorporating changes released in
the following Technical Newsletters:

Form Number	Date
N24-5142	July 27, 1966
N24-5181	November 9, 1966

Significant changes and additions to the specifications contained in this publication will be reported in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for readers' comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Endicott, New York 13760.

CONTENTS

To facilitate disk processing using the Disk Operating System COBOL language, a set of Assembler macros is provided to perform eight direct access storage device (DASD) functions. These macros are being provided on an interim basis. That is, at a future date, these DASD functions will be incorporated into the COBOL language. Conversion to the COBOL language disk statements will consist of merely replacing a few statements in the source program and re-compiling.

For a direct access file, the functions available are:

- File Creation (LOADA)
- Sequential Retrieval (SEQDA)
- Random Retrieval (RANDA)
- Random Retrieval, Update, and Add (RUADA)

For an indexed sequential file, the functions available are:

- File Creation (LODIS)
- Sequential Retrieval and Update (SRUIS)
- Random Retrieval and Update (RRUIS)
- Random Retrieval, Update, and Add (RUAIS)

The eight macros that are provided are written in DOS Assembler language and can be assembled and cataloged into the relocatable library. Each assembled macro is a subprogram tailored to a particular file. It performs a particular function for that file. For instance, if an indexed sequential file is to be created, the parameters of that file are specified in the LODIS macro. The macro is assembled and cataloged into the relocatable library. A CALL statement in the COBOL main program is used to communicate with the generated subprogram.

At linkage-edit time, linkages between the subprogram and the main program are resolved. During execution, when the CALL statement is reached, control transfers to the subprogram which completes a DASD operation and returns control to the COBOL main program.

Relationship of COBOL Main Program and Macro Subprograms

The organization of a program in main storage is submodular (Figure 1). (Each block represents a submodule or subprogram.) Because of submodular program organization, the user can communicate data and control from module to module. The COBOL mainline program controls all logic and processing. When retrieval of a record from a DASD file is required, the COBOL mainline program calls for the services of an Assembler language subprogram which performs the operation requested, and then passes control back to the COBOL mainline.

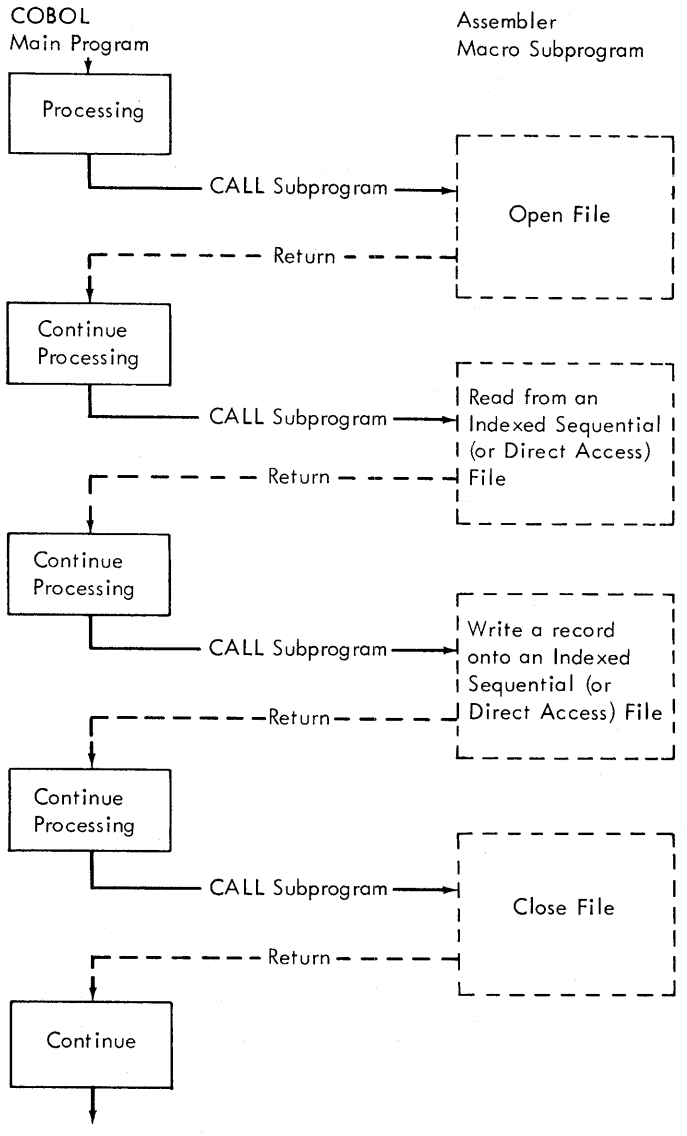


Figure 1. Submodular Organization

The parameters specified in the USING option of the CALL statement communicate the location of data areas within the COBOL main program to the subprogram performing the input/output. After an input operation, the subprogram moves the requested record into the COBOL data area. When an output operation is requested, the subprogram gets the data from the COBOL area and then performs the output operation.

DASD FUNCTIONS

DIRECT ACCESS FILES

Direct access files can be created by use of the LOADA macro (LOAD Direct Access file). They can be processed by use of the SEQDA (SEQuential), RANDA (RANdom), and RUADA (Random retrieval, Update, Add) macros.

LOADA writes user-specified records on a direct access storage device. The device must have been previously initialized by using one of the following:

1. The Initialize Disk Utility Program

2. The Clear Disk Utility Program

3. The IOCS option WRITE RZERO.

LOADA utilizes the WRITE filename,AFTER option of IOCS. The WRITE filename,AFTER option utilizes the first record (R0) on each track to maintain updated information about the records that have been written on the track. R0 contains the ID of the last record written and the number of bytes still available on the track. This information is required when additional records are to be written on the file.

SEQDA reads records from a file for sequential processing by the main COBOL program. It provides an actual key (the disk address of the next record) and a symbolic key (of the record retrieved) after each READ operation. SEQDA utilizes the READ filename,ID option of IOCS. The key should contain track and record.

RANDA randomly retrieves records for processing by the COBOL main program. It can search multiple tracks to retrieve a record, if necessary. RANDA utilizes the READ filename,KEY option of IOCS. The key can contain track or cylinder. Search for the record continues throughout the cylinder.

RUADA randomly retrieves, updates, and adds records on a direct access file. It can search multiple tracks to retrieve a record, if necessary. RUADA utilizes the READ filename,KEY option of IOCS when reading and rewriting and the WRITE filename,AFTER option when adding records. The actual key can contain track or cylinder. Search for the record continues throughout the cylinder.

Records contained within the direct access files processed by these macros must be:

- Fixed-length

- Unblocked

- Identified by an eight-byte binary key (provided by the user). For a description, see the section Direct Access Method: ID Location in the DOS Data Management Concepts publication listed on the cover.

- Identified by a symbolic key.

The form of macro statements for direct access files is:

```
modulename { SEQDA } FILEN=xxxxxx,BLKSIZE=n,KEYLEN=m
           { RANDA }
```

or

```
modulename { LOADA } FILEN=xxxxxx,BLKSIZE=n,KEYLEN=m[,VERIFY=YES]
           { RUADA }
```

modulename - Name used to identify this module. It is the module name used when cataloging the assembled module into the relocatable library. It is the entry point specified in the CALL statement in the COBOL main program. The Linkage Editor uses the modulename to resolve the linkage between the main program and the macro subprogram.

xxxxxx - Name of the file to be processed. This is the first six characters of the filename specified in the VOL statement.

n - Number of bytes contained in the record. This includes the key and data fields.

m - Number of bytes in the key.

The parameter VERIFY=YES is optional. (This is indicated by the brackets.) It is included for LOADA or RUADA if the user wants records to be checked after they are written.

INDEXED SEQUENTIAL FILES

Indexed sequential files can be created by use of the LODIS macro (LOaD Indexed Sequential file). They can be processed by use of the SRUIS (Sequential Retrieval, Update), RRUIS (Random Retrieval, Update), and RUAIS (Random retrieval, Update, Add) macros.

LODIS writes specified records onto a direct access storage device.

SRUIS retrieves records sequentially from an indexed sequential file for processing by the main COBOL program. The records can be updated and written back onto the file (in the original location).

RRUIS randomly retrieves records for processing by the COBOL main program. The records can be updated and written back onto the file (in the original location).

RUAIS randomly retrieves and updates records on an indexed sequential file. It can also add new records to the file.

Records contained within the indexed sequential files processed by these macros must be:

- Fixed-length
- Blocked or unblocked
- Identified by a key. The key in blocked records must be imbedded.

The form of macro statements for indexed sequential files is:

```
           (LODIS)                                                Col 72
modulename  SRUIS  RECSIZE=n,NRECDS=m,KEYLEN=k,KEYLOC=1,               X
            RRUIS            NAM=xxxxxx[,VERIFY=YES]
           (RUAIS)
```

modulename - Name used to identify this module. It is the module name used when cataloging the assembled module into the relocatable library. It is the entry point specified in the CALL statement in the COBOL main program. The Linkage Editor uses the modulename to resolve the linkage between the main program and the macro subprogram.

n - Number of bytes contained in a data record. This does not include a key.

m - Number of records per block.

k - Length of the key in bytes. May not exceed 255 bytes.

l - Location of the key within the record. This is the high-order position of the key within the data record. For instance, if the key is in positions 21-25 of each record in the file, then KEYLOC=21 is specified

xxxxxx - Name of the file to be processed. This is the first six characters of the filename specified in the VOL statement.

The parameter VERIFY=YES is optional. (This is indicated by the brackets.) It is included if the user wants records to be checked after they are written.

USING DASD FUNCTIONS

Certain requirements must be met when using the DASD macros.

1. Specific Logical IOCS modules must be assembled and cataloged into the relocatable library. These must be available for use by DASD macros.

2. Subprograms using DASD macros must be assembled and:

   a. cataloged into the relocatable library (Figure 2), or
   b. punched into cards (Figure 3), or
   c. executed in a compile-and-go environment (see Appendix).

3. When the main COBOL program is compiled, it must contain CALL statements referencing the desired function(s) and working storage statements defining specific data fields.

4. The compiled COBOL main program must be linkage edited (using the appropriate Linkage Editor INCLUDE statements) with the desired DASD subprogram to resolve linkage between them.

ASSEMBLING AND CATALOGING LOGICAL IOCS MODULES

Logical IOCS modules used by DASD subprograms are shown in the following table.

LIOCS Module	Needed By This Macro	Function
IJIBAZZZ	LOADA, RUADA	Creation of a direct access file (also random retrieval, update, and add)
IJIBZIZZ	SEQDA	Sequential retrieval (and updating) of a direct access file
IJIBZZZZ	RANDA	Random retrieval of a direct access file
IJHZLZZZ	LODIS	Creation of an indexed sequential file
IJHZRSZZ	SRUIS	Sequential retrieval of an indexed sequential file
IJHZRRZZ	RRUIS	Random retrieval and update for an indexed sequential file
IJHAARZZ	RUAIS	Random retrieval update, and add for an indexed sequential file

Each LIOCS module must be assembled by itself. The source statements used for assembling these modules follow.

For direct access functions:

Module Name Generated*	Operation	Operand
IJIBAZZZ	DAMOD	AFTER=YES,RECFORM=UNDEF,SEPASMB=YES
IJIBZIZZ	DAMOD	IDLOC=YES,RECFORM=UNDEF,SEPASMB=YES
IJIBZZZZ	DAMOD	RECFORM=UNDEF,SEPASMB=YES

For indexed sequential functions:

Module Name Generated*	Operation	Operand
IJHZLZZZ	ISMOD	IOROUT=LOAD,SEPASMB=YES
IJHZRSZZ	ISMOD	SEPASMB=YES,IOROUT=RETRVE,TYPEFLE=SEQNTL
IJHAARZZ	ISMOD	SEPASMB=YES,IOROUT=ADDRTR,RECFORM=BOTH,TYPEFLE=RANDOM
IJHZRRZZ	ISMOD	SEPASMB=YES,IOROUT=RETRVE,TYPEFLE=RANDOM

*These names are generated by IOCS and should not be punched in the ISMOD or DAMOD card.

When more than one function is used in a program, an indication of duplicate entry points may be given at linkage edit time. If this happens, the user must include a master LIOCS module which includes the individual modules. For example, if both LOADA and SEQDA functions are to be used by the same program, LIOCS modules IJIBAZZZ and IJIBZIZZ would be separately assembled and cataloged into the relocatable library. When they are linkage edited with the COBOL object program, an indication of duplicate entry points may be given. The user must then assemble and catalog (into the relocatable library) a master LIOCS module of which the individual modules are subsets. The name of this master module is determined in the following manner. Use the first four characters of the module name for the functions used. In this case, they would be IJIB. Then take the lowest letter of each of the next four characters in the LIOCS module names. These would be AIZZ. Thus, the name of the master LIOCS module which contains both the LOADA and SEQDA functions is IJIBAIZZ. This module (IJIBAIZZ) must be assembled and cataloged into the relocatable library. The source statement used is a combination of the source statements for the individual modules. That is,

Operation	Operand
DAMOD	AFTER=YES,IDLOC=YES,RECFORM=UNDEF,SEPASMB=YES

The master module is then included with the COBOL object program at linkage edit time instead of the individual modules (IJIBAZZZ and IJIBZIZZ).

ASSEMBLING AND CATALOGING MACRO SUBPROGRAMS

The source statements necessary to assemble DASD macro subprograms are given in the section DASD Functions. Figure 2 illustrates a typical job stream when assembling and cataloging a DASD macro to the relocatble library.

If a DASD macro has been assembled and output in punched cards, linkage editing can not be done from the relocatable library. A typical job stream is illustrated in Figure 3.

```
STEP 1

        Compile Macro Subprogram

Read
By

  R     // JOB Jobname
  R     // OPTION DECK
  R     // EXEC ASSEMBLY
  I       macro statement
  I          END
  I     /*
  R     /&

STEP 2

        Catalog Subprogram into Relocatable Library

Read
By

  R     // JOB Jobname
  R     // EXEC MAINT
  I          CATALR modname
  I          Assembled Subprogram
  I     /*
  R     /&
```

I = SYSIPT
R = SYSRDR

Figure 2. Typical Job Stream for Assembling and Cataloging DASD Subprograms into the Relocatable Library

```
Read
By

R   //    JOB  Jobname
R   //    OPTION  LINK
R         PHASE  name,origin
R         INCLUDE
I      COBOL object program
I   /*
R         INCLUDE
I      Assembled macro subprogram
I   /*
R   //    EXEC  LNKEDT
          .
          .
          .
```

I = SYSIPT
R = SYSRDR

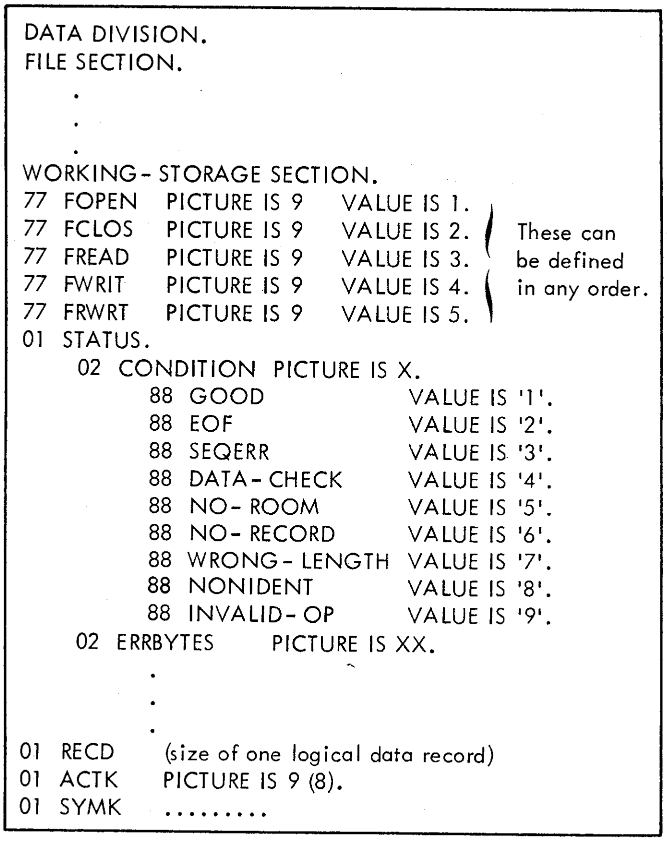
Figure 3. Linkage-Editor Job Stream Using Assembled Macro Subprogram which is not Contained in the Relocatable Library

COMMUNICATION WITH DASD MACRO SUBPROGRAMS

The COBOL facility for setting up the linkage to other modules is contained in the CALL and ENTRY statements. A summary of CALL statement parameters is in Figure 6. The general format for all communication between the main COBOL program and a DASD macro subprogram is:

```
          .
          .
          .
ENTER LINKAGE.

CALL 'modulename' USING operation,parameter1,parameter2,
                        parameter3,parameter4.

ENTER COBOL.

IF conditional statement(s)
          .
          .
          .
```

The CALL statement establishes an exit from the COBOL main program. The USING option of the CALL statement specifies the data field(s) whose location is to be communicated to the called subprogram. These parameters must be defined in the working storage section of the data division of each COBOL main program. Figure 4 illustrates these definitions. The meaning of the parameters and the order in which they must be specified follows.



```
DATA DIVISION.
FILE SECTION.
    .
    .
    .
WORKING-STORAGE SECTION.
77 FOPEN   PICTURE IS 9    VALUE IS 1.  )
77 FCLOS   PICTURE IS 9    VALUE IS 2.  |  These can
77 FREAD   PICTURE IS 9    VALUE IS 3.  |  be defined
77 FWRIT   PICTURE IS 9    VALUE IS 4.  |  in any order.
77 FRWRT   PICTURE IS 9    VALUE IS 5.  )
01 STATUS.
    02 CONDITION  PICTURE IS X.
         88 GOOD             VALUE IS '1'.
         88 EOF              VALUE IS '2'.
         88 SEQERR           VALUE IS '3'.
         88 DATA-CHECK       VALUE IS '4'.
         88 NO-ROOM          VALUE IS '5'.
         88 NO-RECORD        VALUE IS '6'.
         88 WRONG-LENGTH     VALUE IS '7'.
         88 NONIDENT         VALUE IS '8'.
         88 INVALID-OP       VALUE IS '9'.
    02 ERRBYTES       PICTURE IS XX.
         .
         .
         .
01 RECD     (size of one logical data record)
01 ACTK     PICTURE IS 9 (8).
01 SYMK     .........
```

Figure 4. Example of Definitions Required in Working Storage

'modulename' - This symbol is the same as that specified in the corresponding DASD macro. It is the entry point defined in the macro subprogram to which control is transferred.

operation - Specifies an I/O operation. The operations which can be requested are:

Open - Activate file; check label. Symbol must be defined with a value of 1.

Close - Deactivate file. Symbol must be defined with a value of 2.

Read - Read a record into storage. Symbol must be defined with a value of 3.

Write - Write a record from storage. Symbol must be defined with a value of 4.

Rewrite - Write a record (which has just been updated) back onto a random file. Symbol must be defined with a value of 5.

parameter1 - This field is used by the macro subprograms to store status and error information. The first byte is a decimal error indicator which the macro subprograms set after every I/O operation. The second and third bytes contain condition codes set by IOCS. These are in binary form. The programmer may interrogate these for a more detailed analysis of the error. Figure 5 illustrates the contents of the first byte. For bytes 2 and 3, see DOS Supervisor and Input/Output Macros.

Code	Error/Status Condition	Indicator Can Be Set By	Corresponds To Indicator(s) In
1	Good operation	LOADA, SEQDA, RANDA, RUADA	Byte 2, Bits 1,4 Byte 3, Bits 0-6
		LODIS, SRUIS, RRUIS, RUAIS	Byte 2, Bits 0-6
2	End-of-file	SEQDA	Byte 3, Bit 5
		SRUIS	Byte 2, Bit 2
3	Invalid key	LODIS	Byte 2, Bits 5,6
		SRUIS, RRUIS, RUAIS	Byte 2, Bits 3,4,5
4	Data check	LOADA, SEQDA,	Byte 3, Bit 0
		RANDA, RUADA	Byte 3, Bit 3
		LODIS, SRUIS, RRUIS, RUAIS	Byte 2, Bits 0,1
5	No room found	LOADA, RUADA	Byte 2, Bit 4
		LODIS	Byte 2, Bits 2,3
		RUAIS	Byte 2, Bit 6
6	No record found	RANDA	Byte 3, Bits 2,4
		RUADA	Byte 3, Bits 2,4
		SEQDA	Byte 3, Bit 4
7	Wrong-length record	RANDA, RUADA, SEQDA	Byte 2, Bit 1
8	Error condition not identifiable	All macros	Any bits in bytes 2 and/or 3, not indicated above
9	Invalid operation specified for this file	All macros	

Figure 5. First Byte of Three-Byte Status Indicator

parameter2 - I/O area in COBOL main program. It is the length of one logical data record. This is used for communication to and from the macro subprogram.

parameter3 - Actual key (disk address). For a description, refer to the section Direct Access Method: ID Location in the DOS Data Management Concepts publication listed on the cover.

parameter4 - Symbolic key. See the DOS Data Management Concepts publication listed on the cover.

Macro Function	Operation				
	Open	Close	Read	Write	Rewrite
Direct Access File					
File Creation (LOADA)	STATUS	STATUS	X	STATUS RECD* ACTK* SYMK*	X
Sequential Retrieval (SEQDA)	STATUS	STATUS	STATUS RECD## ACTK*## SYMK##	X	X
Random Retrieval (RANDA)	STATUS	STATUS	STATUS RECD## ACTK* SYMK*	X	X
Random Retrieval, Update, Add (RUADA) See Note.	STATUS	STATUS	STATUS RECD## ACTK* SYMK*	STATUS RECD* ACTK* SYMK*	STATUS* RECD ACTK SYMK
Indexed Sequential File					
File Creation (LODIS)	STATUS	STATUS	X	STATUS RECD* SYMK*	X
Sequential Retrieval, Update (SRUIS)	STATUS SYMK*	STATUS	STATUS RECD ## SYMK##	X	STATUS RECD
Random Retrieval, Update (RRUIS) See Note.	STATUS	STATUS	STATUS RECD## SYMK*	X	STATUS RECD
Random Retrieval, Update, Add (RUAIS) See Note.	STATUS	STATUS	STATUS RECD ## SYMK*	STATUS RECD* SYMK*	STATUS RECD*

X Operation not valid for this macro function.
* Must be loaded before the CALL statement is issued.
Available to the user after the operation is complete.

Note: Before any rewrite operation, a read operation must have taken place.

STATUS	= parameter 1	ACTK	= parameter 3
RECD	= parameter 2	SYMK	= parameter 4

Figure 6. Summary of CALL Statement Parameters

Examples of Calls to the LOADA Subprogram

The OPEN statement for the LOADA subprogram is:

```
     CALL 'modulename' USING FOPEN, STATUS.
```

To write a record on a file, load the constants RECD, ACTK, and SYMK. Then issue this statement:

```
     CALL 'modulename' USING FWRIT, STATUS, RECD, ACTK, SYMK.
```

The CLOSE statement for the LOADA subprogram is:

```
     CALL 'modulename' USING FCLOS, STATUS.
```

Examples of Calls to the SEQDA Subprogram

The OPEN statement for the SEQDA subprogram is:

```
     CALL 'modulename' USING FOPEN, STATUS.
```

To read a record from a sequential file, load ACTK with the actual key (in binary) and then issue this statement:

```
     CALL 'modulename' USING FREAD, STATUS, RECD, ACTK, SYMK.
```

After the read operation is completed, RECD, ACTK, and SYMK are available to the user. ACTK contains the actual key of the next sequential record in the file. SYMK contains the symbolic key. The CLOSE statement for the SEQDA subprogram is:

```
     CALL 'modulename' USING FCLOS, STATUS.
```

Examples of Calls to the RANDA Subprogram

The OPEN statement for the RANDA subprogram is:

```
     CALL 'modulename' USING FOPEN, STATUS.
```

To read a record from a random file, load ACTK with the actual key (in binary) and SYMK with the symbolic key. Then issue this statement:

```
     CALL 'modulename' USING FREAD, STATUS, RECD, ACTK, SYMK.
```

After the read operation is completed, RECD is available to the user. The CLOSE statement for the RANDA subprogram is:

```
     CALL 'modulename' USING FCLOS, STATUS.
```

Examples of Calls to the RUADA Subprogram

The OPEN statement for the RUADA subprogram is:

```
CALL 'modulename' USING FOPEN, STATUS.
```

To retrieve a record from a random file, load ACTK with the actual key (in binary) and SYMK with the symbolic key. Then issue this statement:

```
CALL 'modulename' USING FREAD, STATUS, RECD, ACTK, SYMK.
```

After the read operation is completed, RECD is available to the user. To write an updated record back onto the random file, use:

```
CALL 'modulename' USING FRWRT, STATUS, RECD, ACTK, SYMK.
```

To write a new record on a random file, load RECD, ACTK, and SYMK. Then issue this statement:

```
CALL 'modulename' USING FWRIT, STATUS, RECD, ACTK, SYMK.
```

The CLOSE statement for the RUADA subprogram is:

```
CALL 'modulename' USING FCLOS, STATUS.
```

Examples of Calls to the LODIS Subprogram

The OPEN statement for the LODIS subprogram is:

```
CALL 'modulename' USING FOPEN, STATUS.
```

To write a record on a file, load the constants RECD and SYMK. Then issue this statement:

```
CALL 'modulename' USING FWRIT, STATUS, RECD, SYMK.
```

The CLOSE statement for the LODIS subprogram is:

```
CALL 'modulename' USING FCLOS, STATUS.
```

Examples of Calls to the SRUIS Subprogram

Before issuing the OPEN statement for SRUIS, load SYMK with the symbolic key of the record in the file where sequential retrieval is to begin. Load SYMK with zeros if sequential retrieval is to begin at the starting address of the file. The OPEN statement is:

```
CALL 'modulename' USING FOPEN, STATUS, SYMK.
```

To read a record from an indexed sequential file, load SYMK as previously described and issue the following statement:

```
CALL 'modulename' USING FREAD, STATUS, RECD, SYMK.
```

After the read operation is completed, RECD and SYMK are available to the user. To write an updated record back onto the indexed sequential file, use:

```
CALL 'modulename' USING FRWRT, STATUS, RECD.
```

The CLOSE statement for the SRUIS subprogram is:

```
CALL 'modulename' USING FCLOS, STATUS.
```

Examples of Calls to the RRUIS Subprogram

The OPEN statement for the RRUIS subprogram is:

```
     CALL 'modulename' USING FOPEN, STATUS.
```

To read a record, load SYMK with the symbolic key. Then issue this statement:

```
     CALL 'modulename' USING FREAD, STATUS, RECD, SYMK.
```

After the read operation is completed, RECD is available to the user. To write a record back onto the indexed sequential file, use the statement:

```
     CALL 'modulename' USING FRWRT, STATUS, RECD.
```

The CLOSE statement for the RRUIS subprogram is:

```
     CALL 'modulename' USING FCLOS, STATUS.
```

Examples of Calls to the RUAIS Subprogram

The OPEN statement for the RUAIS subprogram is:

```
     CALL 'modulename' USING FOPEN STATUS.
```

To read a record from the indexed sequential file, load SYMK and issue the following statement:

```
     CALL 'modulename' USING FREAD, STATUS, RECD, SYMK.
```

To write an updated record back onto the file, use:

```
     CALL 'modulename' USING FRWRT, STATUS, RECD.
```

To write a new record on an indexed sequential file, load SYMK and RECD and issue this statement:

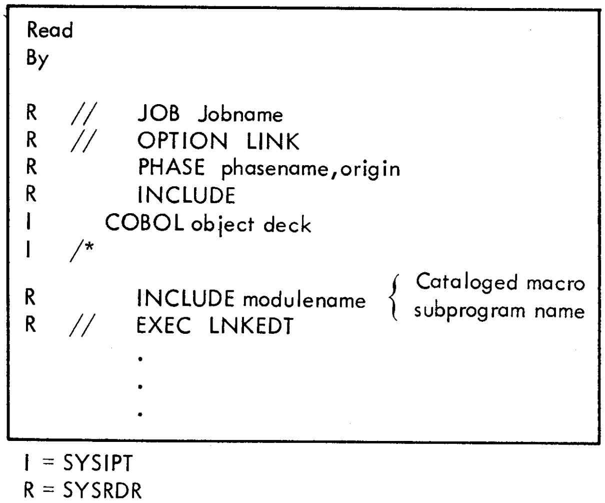
```
     CALL 'modulename' USING FWRIT STATUS, RECD, SYMK.
```

The CLOSE statement for the RUAIS subprogram is:

```
     CALL 'modulename' USING FCLOS, STATUS.
```

LINKAGE EDITING COBOL MAIN PROGRAM WITH MACRO SUBPROGRAM

Figure 7 illustrates the statements necessary to complete the linkage between a COBOL main program and a macro subprogram which is cataloged in the relocatable library.

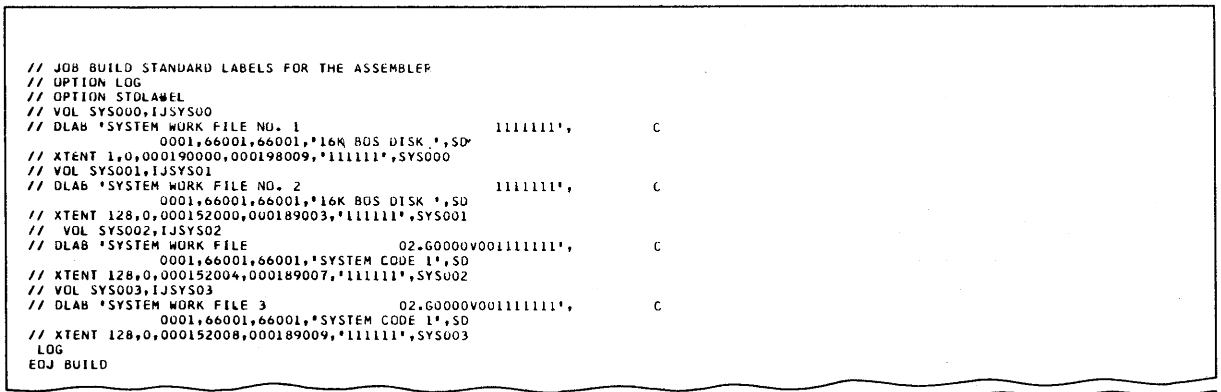


```
Read
By

R   //    JOB  Jobname
R   //    OPTION  LINK
R         PHASE  phasename,origin
R         INCLUDE
I      COBOL object deck
I   /*
                                    { Cataloged macro
R         INCLUDE modulename        { subprogram name
R   //    EXEC  LNKEDT
          .
          .
          .
```

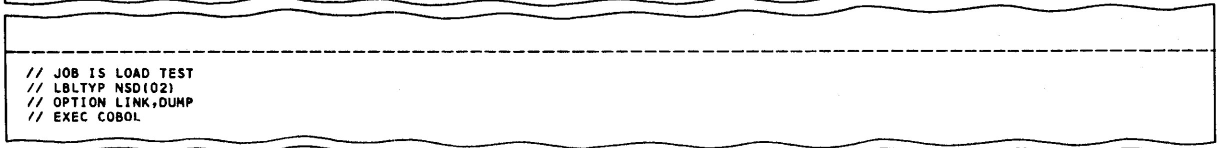
I = SYSIPT
R = SYSRDR

Figure 7. Typical Job Stream for Linkage Editing COBOL Main Program and Macro Subprogram

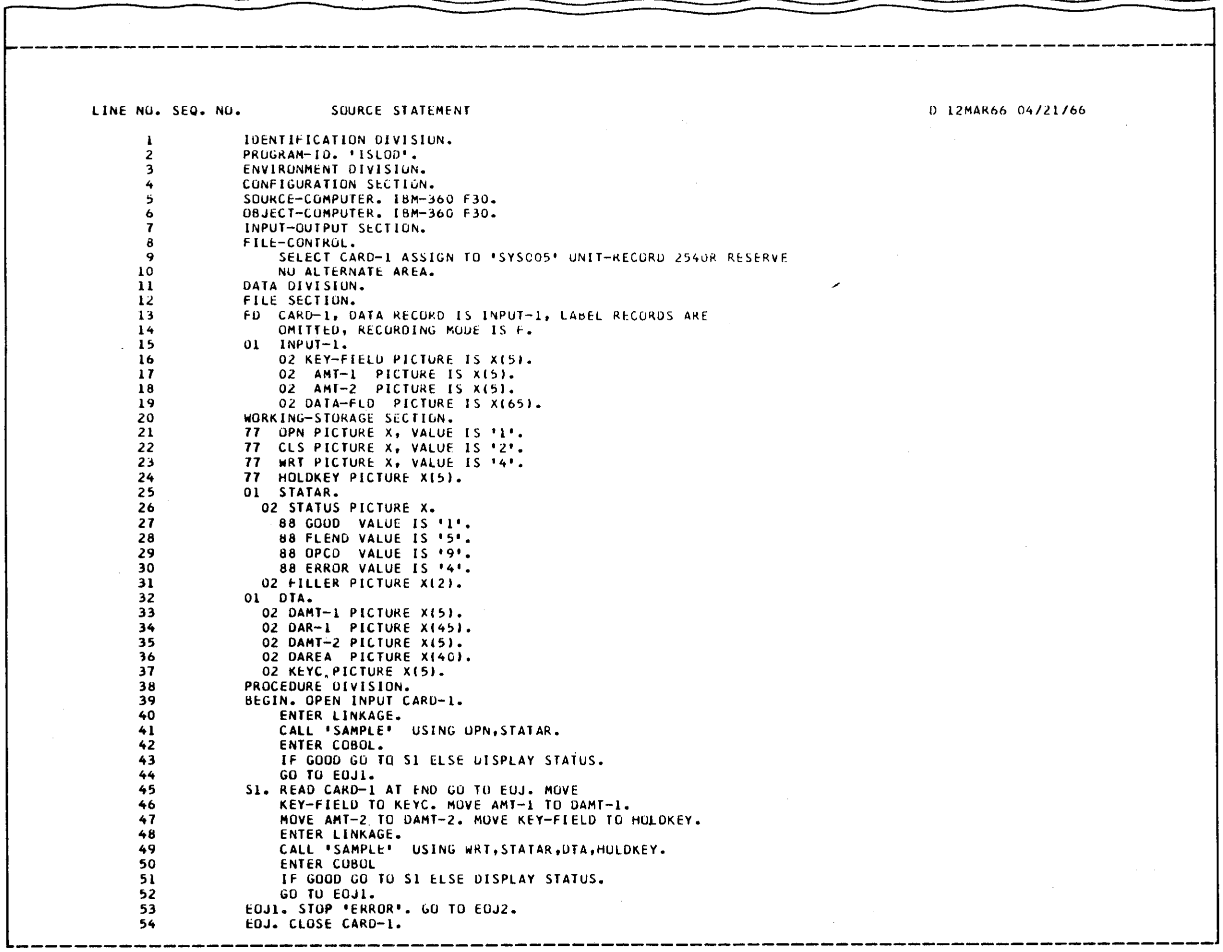
APPENDIX: EXAMPLE USING LODIS



```
// JOB BUILD STANDARD LABELS FOR THE ASSEMBLER
// OPTION LOG
// OPTION STDLABEL
// VOL SYS000,IJSYS00
// DLAB 'SYSTEM WORK FILE NO. 1                          1111111',          C
               0001,66001,66001,'16K BOS DISK ',SD
// XTENT 1,0,000190000,000198009,'111111',SYS000
// VOL SYS001,IJSYS01
// DLAB 'SYSTEM WORK FILE NO. 2                          1111111',          C
               0001,66001,66001,'16K BOS DISK ',SD
// XTENT 128,0,000152000,000189003,'111111',SYS001
//  VOL SYS002,IJSYS02
// DLAB 'SYSTEM WORK FILE                     02.G0000V001111111',          C
               0001,66001,66001,'SYSTEM CODE 1',SD
// XTENT 128,0,000152004,000189007,'111111',SYS002
// VOL SYS003,IJSYS03
// DLAB 'SYSTEM WORK FILE 3                   02.G0000V001111111',          C
               0001,66001,66001,'SYSTEM CODE 1',SD
// XTENT 128,0,000152008,000189009,'111111',SYS003
 LOG
EOJ BUILD
```

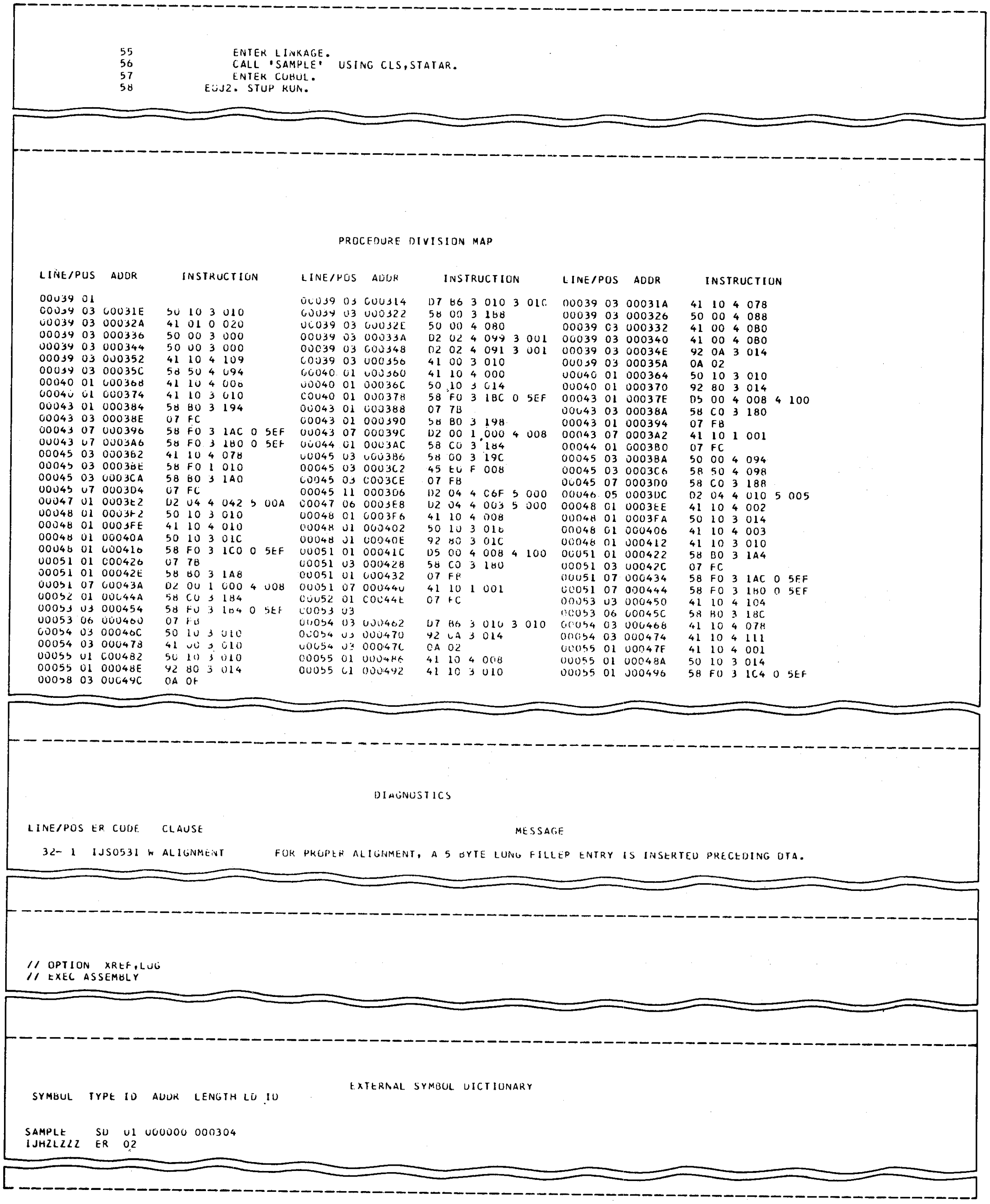


```
// JOB IS LOAD TEST
// LBLTYP NSD(02)
// OPTION LINK,DUMP
// EXEC COBOL
```



```
LINE NO. SEQ. NO.         SOURCE STATEMENT                                                       D 12MAR66 04/21/66

     1           IDENTIFICATION DIVISION.
     2           PROGRAM-ID. 'ISLOD'.
     3           ENVIRONMENT DIVISION.
     4           CONFIGURATION SECTION.
     5           SOURCE-COMPUTER. IBM-360 F30.
     6           OBJECT-COMPUTER. IBM-360 F30.
     7           INPUT-OUTPUT SECTION.
     8           FILE-CONTROL.
     9               SELECT CARD-1 ASSIGN TO 'SYS005' UNIT-RECORD 2540R RESERVE
    10               NO ALTERNATE AREA.
    11           DATA DIVISION.
    12           FILE SECTION.
    13           FD  CARD-1, DATA RECORD IS INPUT-1, LABEL RECORDS ARE
    14               OMITTED, RECORDING MODE IS F.
    15           01  INPUT-1.
    16               02 KEY-FIELD PICTURE IS X(5).
    17               02  AMT-1  PICTURE IS X(5).
    18               02  AMT-2  PICTURE IS X(5).
    19               02 DATA-FLD  PICTURE IS X(65).
    20           WORKING-STORAGE SECTION.
    21           77  OPN PICTURE X, VALUE IS '1'.
    22           77  CLS PICTURE X, VALUE IS '2'.
    23           77  WRT PICTURE X, VALUE IS '4'.
    24           77  HOLDKEY PICTURE X(5).
    25           01  STATAR.
    26             02 STATUS PICTURE X.
    27               88 GOOD  VALUE IS '1'.
    28               88 FLEND VALUE IS '5'.
    29               88 OPCD  VALUE IS '9'.
    30               88 ERROR VALUE IS '4'.
    31             02 FILLER PICTURE X(2).
    32           01  DTA.
    33             02 DAMT-1 PICTURE X(5).
    34             02 DAR-1  PICTURE X(45).
    35             02 DAMT-2 PICTURE X(5).
    36             02 DAREA  PICTURE X(40).
    37             02 KEYC PICTURE X(5).
    38           PROCEDURE DIVISION.
    39           BEGIN. OPEN INPUT CARD-1.
    40               ENTER LINKAGE.
    41               CALL 'SAMPLE'  USING OPN,STATAR.
    42               ENTER COBOL.
    43               IF GOOD GO TO S1 ELSE DISPLAY STATUS.
    44               GO TO EOJ1.
    45           S1. READ CARD-1 AT END GO TO EOJ. MOVE
    46               KEY-FIELD TO KEYC. MOVE AMT-1 TO DAMT-1.
    47               MOVE AMT-2 TO DAMT-2. MOVE KEY-FIELD TO HOLDKEY.
    48               ENTER LINKAGE.
    49               CALL 'SAMPLE'  USING WRT,STATAR,DTA,HOLDKEY.
    50               ENTER COBOL
    51               IF GOOD GO TO S1 ELSE DISPLAY STATUS.
    52               GO TO EOJ1.
    53           EOJ1. STOP 'ERROR'. GO TO EOJ2.
    54           EOJ. CLOSE CARD-1.
```

Appendix. Example Using LODIS (Part 1 of 6)



```
           55            ENTER LINKAGE.
           56            CALL 'SAMPLE'  USING CLS,STATAR.
           57            ENTER COBOL.
           58        EOJ2. STOP RUN.
```

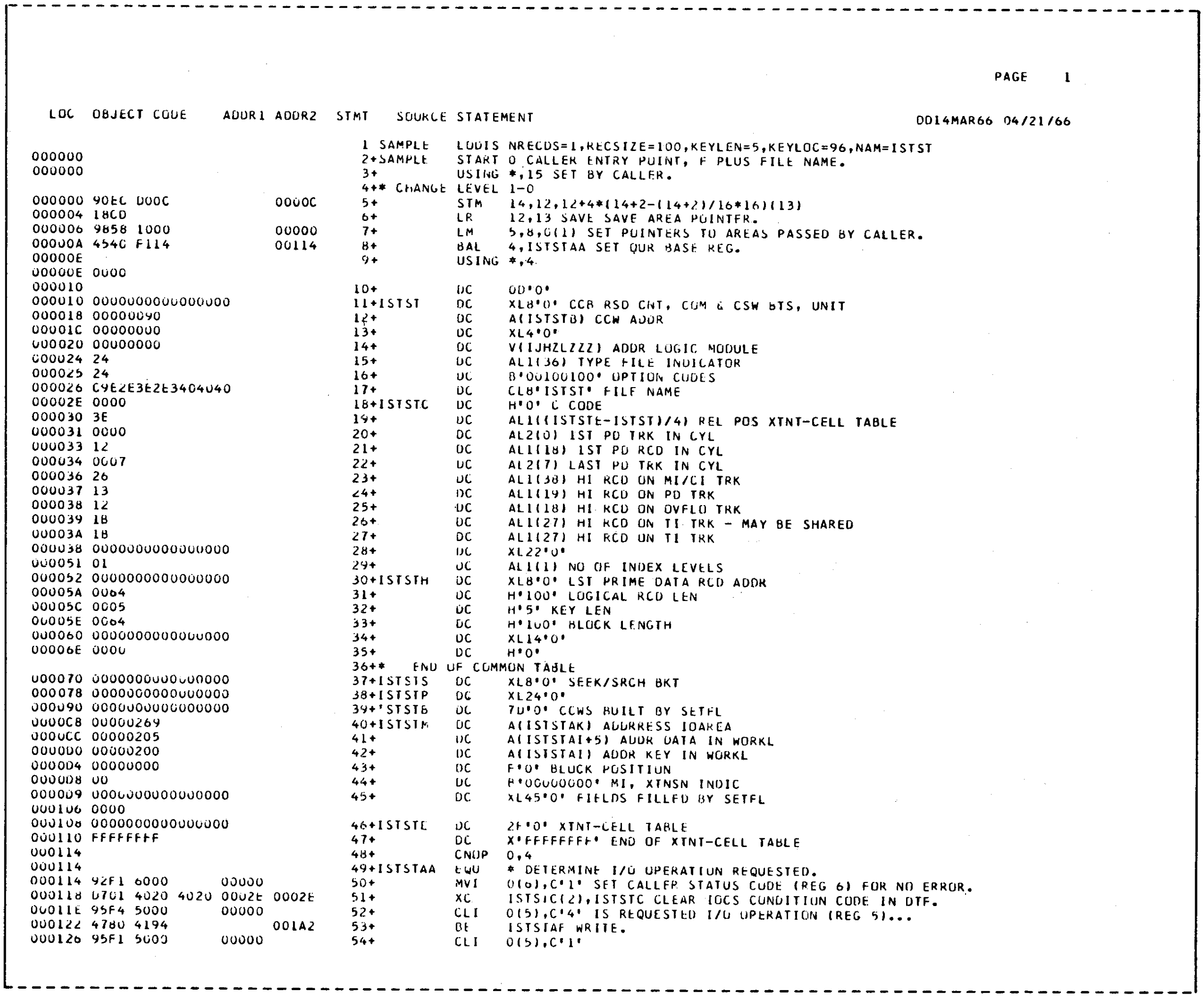
```
                                          PROCEDURE DIVISION MAP

 LINE/POS   ADDR       INSTRUCTION         LINE/POS   ADDR       INSTRUCTION         LINE/POS   ADDR       INSTRUCTION

 00039 01                                  00039 03 000314   D7 B6 3 010 3 01C       00039 03 00031A   41 10 4 078
 00039 03 00031E   50 10 3 010             00039 03 000322   58 00 3 188             00039 03 000326   50 00 4 088
 00039 03 00032A   41 01 0 020             00039 03 00032E   50 00 4 080             00039 03 000332   41 00 4 080
 00039 03 000336   50 00 3 000             00039 03 00033A   D2 02 4 099 3 001       00039 03 000340   41 00 4 080
 00039 03 000344   50 00 3 000             00039 03 000348   D2 02 4 091 3 001       00039 03 00034E   92 0A 3 014
 00039 03 000352   41 10 4 109             00039 03 000356   41 00 3 010             00039 03 00035A   0A 02
 00039 03 00035C   58 50 4 094             00040 01 000360   41 10 4 000             00040 01 000364   50 10 3 010
 00040 01 000368   41 10 4 006             00040 01 00036C   50 10 3 014             00040 01 000370   92 80 3 014
 00040 01 000374   41 10 3 010             00040 01 000378   58 F0 3 1BC 0 5EF       00043 01 00037E   D5 00 4 008 4 100
 00043 01 000384   58 80 3 194             00043 01 000388   07 7B                   00043 03 00038A   58 C0 3 180
 00043 03 00038E   07 FC                   00043 01 000390   58 B0 3 198             00043 01 000394   07 FB
 00043 07 000396   58 F0 3 1AC 0 5EF       00043 07 00039C   D2 00 1 000 4 008       00043 07 0003A2   41 10 1 001
 00043 07 0003A6   58 F0 3 1B0 0 5EF       00044 01 0003AC   58 C0 3 184             00044 01 0003B0   07 FC
 00045 03 0003B2   41 10 4 078             00045 03 0003B6   58 00 3 19C             00045 03 0003BA   50 00 4 094
 00045 03 0003BE   58 F0 1 010             00045 03 0003C2   45 E0 F 008             00045 03 0003C6   58 50 4 098
 00045 03 0003CA   58 B0 3 1A0             00045 03 0003CE   07 FB                   00045 07 0003D0   58 C0 3 188
 00045 07 0003D4   07 FC                   00045 11 0003D6   D2 04 4 C6F 5 000       00046 05 0003DC   D2 04 4 010 5 005
 00047 01 0003E2   D2 04 4 042 5 00A       00047 06 0003E8   D2 04 4 003 5 000       00048 01 0003EE   41 10 4 002
 00048 01 0003F2   50 10 3 010             00048 01 0003F6   41 10 4 008             00048 01 0003FA   50 10 3 014
 00048 01 0003FE   41 10 4 010             00048 01 000402   50 10 3 016             00048 01 000406   41 10 4 003
 00048 01 00040A   50 10 3 01C             00048 01 00040E   92 80 3 01C             00048 01 000412   41 10 3 010
 00048 01 000416   58 F0 3 1C0 0 5EF       00051 01 00041C   D5 00 4 008 4 100       00051 01 000422   58 B0 3 1A4
 00051 01 000426   07 7B                   00051 03 000428   58 C0 3 180             00051 03 00042C   07 FC
 00051 01 00042E   58 B0 3 1A8             00051 01 000432   07 FB                   00051 07 000434   58 F0 3 1AC 0 5EF
 00051 07 00043A   D2 00 1 000 4 008       00051 07 000440   41 10 1 001             00051 07 000444   58 F0 3 1B0 0 5EF
 00052 01 00044A   58 C0 3 184             00052 01 00044E   07 FC                   00053 03 000450   41 10 4 104
 00053 03 000454   58 F0 3 1B4 0 5EF       00053 03                                  00053 06 00045C   58 B0 3 18C
 00053 06 000460   07 FB                   00054 03 000462   D7 B6 3 010 3 010       00054 03 000468   41 10 4 078
 00054 03 00046C   50 10 3 010             00054 03 000470   92 0A 3 014             00054 03 000474   41 10 4 111
 00054 03 000478   41 00 3 010             00054 03 00047C   0A 02                   00055 01 00047F   41 10 4 001
 00055 01 000482   50 10 3 010             00055 01 000486   41 10 4 008             00055 01 00048A   50 10 3 014
 00055 01 00048E   92 80 3 014             00055 01 000492   41 10 3 010             00055 01 000496   58 F0 3 1C4 0 5EF
 00058 03 00049C   0A 0F
```

```
                                                    DIAGNOSTICS

LINE/POS ER CODE   CLAUSE                                       MESSAGE

  32- 1  IJS0531 W ALIGNMENT       FOR PROPER ALIGNMENT, A 5 BYTE LONG FILLER ENTRY IS INSERTED PRECEDING DTA.
```

```
 // OPTION  XREF,LOG
 // EXEC ASSEMBLY
```

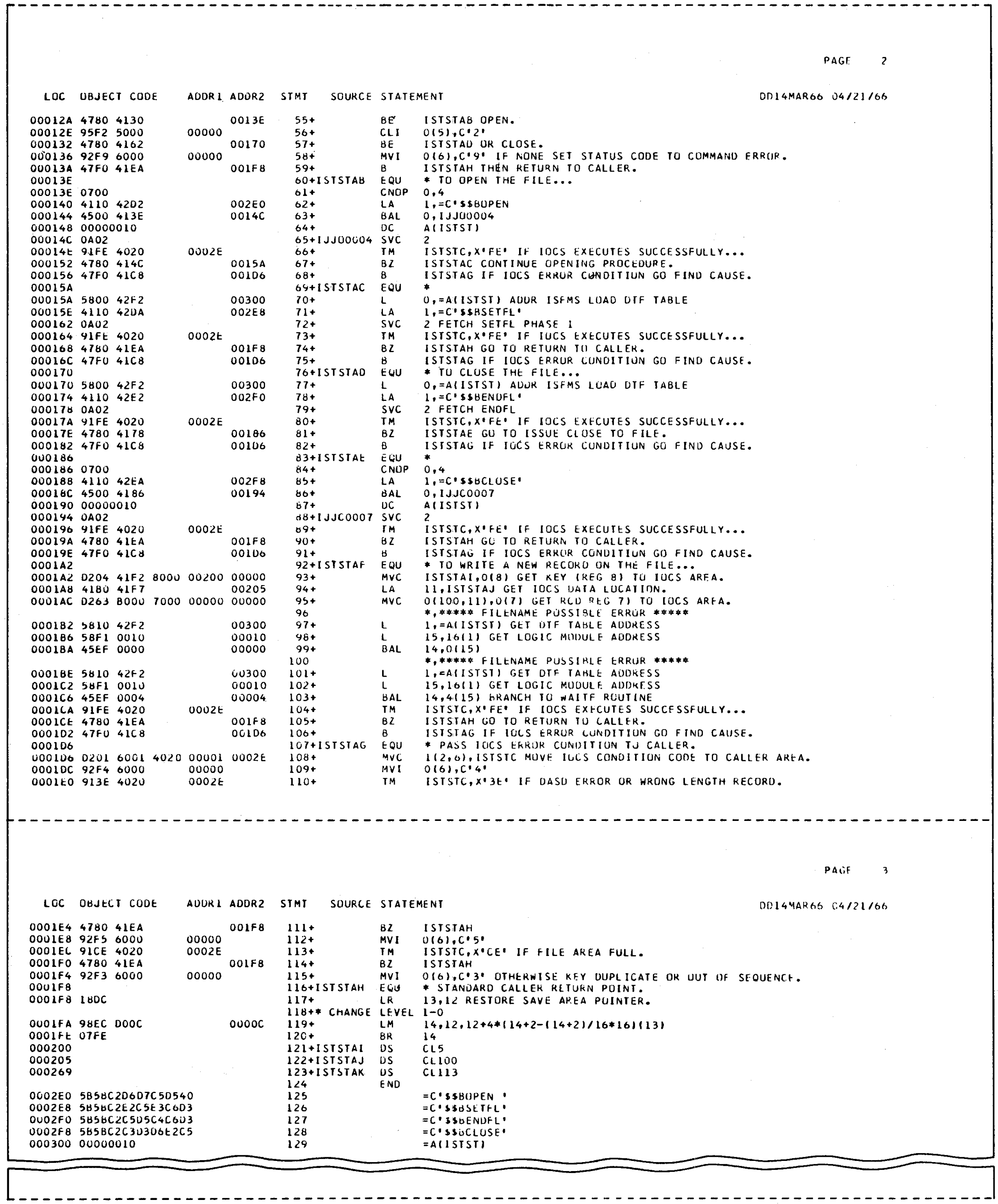
```
                                               EXTERNAL SYMBOL DICTIONARY
  SYMBOL  TYPE ID  ADDR  LENGTH LD ID

 SAMPLE     SD  01 000000 000304
 IJHZLZZZ   ER  02
```

Appendix. Example Using LODIS (Part 2 of 6)



```
                                                                                                  PAGE    1

  LOC  OBJECT CODE     ADDR1 ADDR2  STMT    SOURCE STATEMENT                                   DD14MAR66 04/21/66

                                     1 SAMPLE   LODIS NRECDS=1,RECSIZE=100,KEYLEN=5,KEYLOC=96,NAM=ISTST
000000                               2+SAMPLE   START 0 CALLER ENTRY POINT, F PLUS FILE NAME.
000000                               3+         USING *,15 SET BY CALLER.
                                     4+* CHANGE LEVEL 1-0
000000 90EC D00C             0000C   5+         STM   14,12,12+4*(14+2-(14+2)/16*16)(13)
000004 18CD                          6+         LR    12,13 SAVE SAVE AREA POINTER.
000006 9858 1000             00000   7+         LM    5,8,0(1) SET POINTERS TO AREAS PASSED BY CALLER.
00000A 4540 F114             00114   8+         BAL   4,ISTSTAA SET OUR BASE REG.
00000E                               9+         USING *,4
00000E 0000
000010                              10+         DC    0D'0'
000010 0000000000000000             11+ISTST    DC    XL8'0' CCB RSD CNT, COM & CSW BTS, UNIT
000018 00000090                     12+         DC    A(ISTSTB) CCW ADDR
00001C 00000000                     13+         DC    XL4'0'
000020 00000000                     14+         DC    V(IJHZLZZZ) ADDR LOGIC MODULE
000024 24                           15+         DC    AL1(36) TYPE FILE INDICATOR
000025 24                           16+         DC    B'00100100' OPTION CODES
000026 C9E2E3E2E3404040             17+         DC    CL8'ISTST' FILF NAME
00002E 0000                         18+ISTSTC   DC    H'0' C CODE
000030 3E                           19+         DC    AL1((ISTSTE-ISTST)/4) REL POS XTNT-CELL TABLE
000031 0000                         20+         DC    AL2(0) 1ST PD TRK IN CYL
000033 12                           21+         DC    AL1(18) 1ST PD RCD IN CYL
000034 0007                         22+         DC    AL2(7) LAST PD TRK IN CYL
000036 26                           23+         DC    AL1(38) HI RCD ON MI/CI TRK
000037 13                           24+         DC    AL1(19) HI RCD ON PD TRK
000038 12                           25+         DC    AL1(18) HI RCD ON OVFLO TRK
000039 1B                           26+         DC    AL1(27) HI RCD ON TI TRK - MAY BE SHARED
00003A 1B                           27+         DC    AL1(27) HI RCD ON TI TRK
00003B 0000000000000000             28+         DC    XL22'0'
000051 01                           29+         DC    AL1(1) NO OF INDEX LEVELS
000052 0000000000000000             30+ISTSTH   DC    XL8'0' LST PRIME DATA RCD ADDR
00005A 0064                         31+         DC    H'100' LOGICAL RCD LEN
00005C 0005                         32+         DC    H'5' KEY LEN
00005E 0064                         33+         DC    H'100' BLOCK LENGTH
000060 0000000000000000             34+         DC    XL14'0'
00006E 0000                         35+         DC    H'0'
                                    36+*   END OF COMMON TABLE
000070 0000000000000000             37+ISTSTS   DC    XL8'0' SEEK/SRCH BKT
000078 0000000000000000             38+ISTSTP   DC    XL24'0'
000090 0000000000000000             39+'STSTB   DC    7D'0' CCWS BUILT BY SETFL
0000C8 00000269                     40+ISTSTM   DC    A(ISTSTAK) ADDRRESS IOAREA
0000CC 00000205                     41+         DC    A(ISTSTAI+5) ADDR DATA IN WORKL
0000D0 00000200                     42+         DC    A(ISTSTAI) ADDR KEY IN WORKL
0000D4 00000000                     43+         DC    F'0' BLOCK POSITION
0000D8 00                           44+         DC    B'00000000' MI, XTNSN INDIC
0000D9 0000000000000000             45+         DC    XL45'0' FIELDS FILLED BY SETFL
000106 0000
000108 0000000000000000             46+ISTSTE   DC    2F'0' XTNT-CELL TABLE
000110 FFFFFFFF                     47+         DC    X'FFFFFFFF' END OF XTNT-CELL TABLE
000114                              48+         CNOP  0,4
000114                              49+ISTSTAA  EQU   * DETERMINE I/O OPERATION REQUESTED.
000114 92F1 6000       00000        50+         MVI   0(6),C'1' SET CALLER STATUS CODE (REG 6) FOR NO ERROR.
000118 D701 4020 4020 0002E 0002E   51+         XC    ISTSTC(2),ISTSTC CLEAR IOCS CONDITION CODE IN DTF.
00011E 95F4 5000       00000        52+         CLI   0(5),C'4' IS REQUESTED I/O OPERATION (REG 5)...
000122 4780 4194             001A2  53+         BE    ISTSTAF WRITE.
000126 95F1 5000       00000        54+         CLI   0(5),C'1'
```

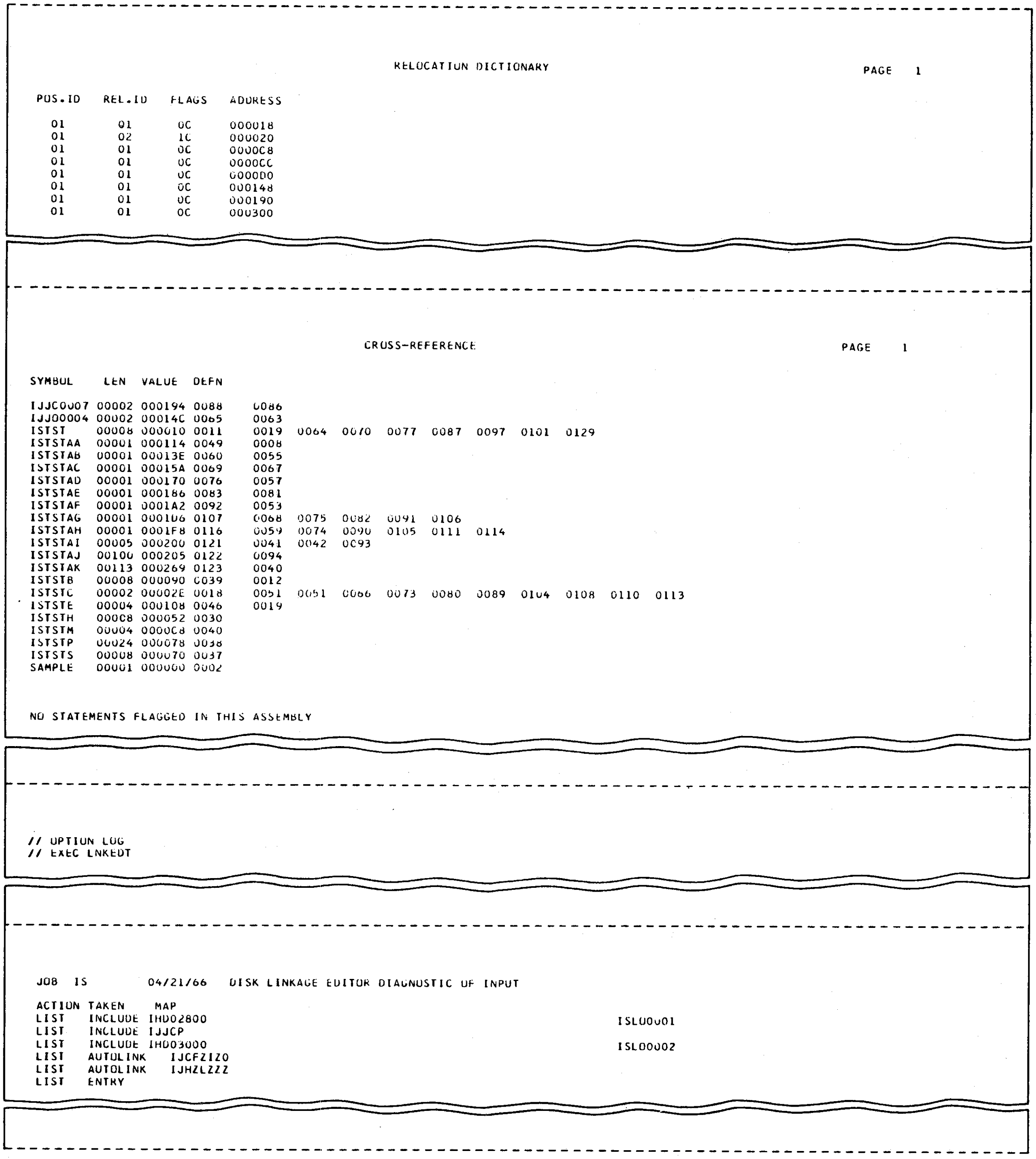
Appendix. Example Using LODIS (Part 3 of 6)



```
                                                                                                                PAGE    2

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                DD14MAR66 04/21/66

00012A 4780 4130            0013E    55+          BE    ISTSTAB OPEN.
00012E 95F2 5000      00000          56+          CLI   0(5),C'2'
000132 4780 4162            00170    57+          BE    ISTSTAD OR CLOSE.
000136 92F9 6000      00000          58+          MVI   0(6),C'9' IF NONE SET STATUS CODE TO COMMAND ERROR.
00013A 47F0 41EA            001F8    59+          B     ISTSTAH THEN RETURN TO CALLER.
00013E                               60+ISTSTAB   EQU   * TO OPEN THE FILE...
00013E 0700                          61+          CNOP  0,4
000140 4110 42D2            002E0    62+          LA    1,=C'$$BOPEN
000144 4500 413E            0014C    63+          BAL   0,IJJ00004
000148 00000010                      64+          DC    A(ISTST)
00014C 0A02                          65+IJJ00004  SVC   2
00014E 91FE 4020      0002E          66+          TM    ISTSTC,X'FE' IF IOCS EXECUTES SUCCESSFULLY...
000152 4780 414C            0015A    67+          BZ    ISTSTAC CONTINUE OPENING PROCEDURE.
000156 47F0 41C8            001D6    68+          B     ISTSTAG IF IOCS ERROR CONDITION GO FIND CAUSE.
00015A                               69+ISTSTAC   EQU   *
00015A 5800 42F2            00300    70+          L     0,=A(ISTST) ADDR ISFMS LOAD DTF TABLE
00015E 4110 42DA            002E8    71+          LA    1,=C'$$BSETFL'
000162 0A02                          72+          SVC   2 FETCH SETFL PHASE 1
000164 91FE 4020      0002E          73+          TM    ISTSTC,X'FE' IF IOCS EXECUTES SUCCESSFULLY...
000168 4780 41EA            001F8    74+          BZ    ISTSTAH GO TO RETURN TO CALLER.
00016C 47F0 41C8            001D6    75+          B     ISTSTAG IF IOCS ERROR CONDITION GO FIND CAUSE.
000170                               76+ISTSTAD   EQU   * TO CLOSE THE FILE...
000170 5800 42F2            00300    77+          L     0,=A(ISTST) ADDR ISFMS LOAD DTF TABLE
000174 4110 42E2            002F0    78+          LA    1,=C'$$BENDFL'
000178 0A02                          79+          SVC   2 FETCH ENDFL
00017A 91FE 4020      0002E          80+          TM    ISTSTC,X'FE' IF IOCS EXECUTES SUCCESSFULLY...
00017E 4780 4178            00186    81+          BZ    ISTSTAE GO TO ISSUE CLOSE TO FILE.
000182 47F0 41C8            001D6    82+          B     ISTSTAG IF IOCS ERROR CONDITION GO FIND CAUSE.
000186                               83+ISTSTAE   EQU   *
000186 0700                          84+          CNOP  0,4
000188 4110 42EA            002F8    85+          LA    1,=C'$$BCLOSE'
00018C 4500 4186            00194    86+          BAL   0,IJJC0007
000190 00000010                      87+          DC    A(ISTST)
000194 0A02                          88+IJJC0007  SVC   2
000196 91FE 4020      0002E          89+          TM    ISTSTC,X'FE' IF IOCS EXECUTES SUCCESSFULLY...
00019A 4780 41EA            001F8    90+          BZ    ISTSTAH GO TO RETURN TO CALLER.
00019E 47F0 41C8            001D6    91+          B     ISTSTAG IF IOCS ERROR CONDITION GO FIND CAUSE.
0001A2                               92+ISTSTAF   EQU   * TO WRITE A NEW RECORD ON THE FILE...
0001A2 D204 41F2 8000 00200 00000    93+          MVC   ISTSTAI,0(8) GET KEY (REG 8) TO IOCS AREA.
0001A8 41B0 41F7            00205    94+          LA    11,ISTSTAJ GET IOCS DATA LOCATION.
0001AC D263 B000 7000 00000 00000    95+          MVC   0(100,11),0(7) GET RCD REG 7) TO IOCS AREA.
                                     96                 *,***** FILENAME POSSIBLE ERROR *****
0001B2 5810 42F2            00300    97+          L     1,=A(ISTST) GET DTF TABLE ADDRESS
0001B6 58F1 0010            00010    98+          L     15,16(1) GET LOGIC MODULE ADDRESS
0001BA 45EF 0000            00000    99+          BAL   14,0(15)
                                     100                *,***** FILENAME POSSIBLE ERROR *****
0001BE 5810 42F2            00300    101+         L     1,=A(ISTST) GET DTF TABLE ADDRESS
0001C2 58F1 0010            00010    102+         L     15,16(1) GET LOGIC MODULE ADDRESS
0001C6 45EF 0004            00004    103+         BAL   14,4(15) BRANCH TO WAITF ROUTINE
0001CA 91FE 4020      0002E          104+         TM    ISTSTC,X'FE' IF IOCS EXECUTES SUCCESSFULLY...
0001CE 4780 41EA            001F8    105+         BZ    ISTSTAH GO TO RETURN TO CALLER.
0001D2 47F0 41C8            001D6    106+         B     ISTSTAG IF IOCS ERROR CONDITION GO FIND CAUSE.
0001D6                               107+ISTSTAG  EQU   * PASS IOCS ERROR CONDITION TO CALLER.
0001D6 D201 6001 4020 00001 0002E    108+         MVC   1(2,6),ISTSTC MOVE IOCS CONDITION CODE TO CALLER AREA.
0001DC 92F4 6000      00000          109+         MVI   0(6),C'4'
0001E0 913E 4020      0002E          110+         TM    ISTSTC,X'3E' IF DASD ERROR OR WRONG LENGTH RECORD.
```

```
                                                                                                                PAGE    3

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                DD14MAR66 04/21/66

0001E4 4780 41EA            001F8    111+         BZ    ISTSTAH
0001E8 92F5 6000      00000          112+         MVI   0(6),C'5'
0001EC 91CE 4020      0002E          113+         TM    ISTSTC,X'CE' IF FILE AREA FULL.
0001F0 4780 41EA            001F8    114+         BZ    ISTSTAH
0001F4 92F3 6000      00000          115+         MVI   0(6),C'3' OTHERWISE KEY DUPLICATE OR OUT OF SEQUENCE.
0001F8                               116+ISTSTAH  EQU   * STANDARD CALLER RETURN POINT.
0001F8 18DC                          117+         LR    13,12 RESTORE SAVE AREA POINTER.
                                     118+* CHANGE LEVEL 1-0
0001FA 98EC D00C            0000C    119+         LM    14,12,12+4*(14+2-(14+2)/16*16)(13)
0001FE 07FE                          120+         BR    14
000200                               121+ISTSTAI  DS    CL5
000205                               122+ISTSTAJ  DS    CL100
000269                               123+ISTSTAK  DS    CL113
                                     124          END
0002E0 5B5BC2D6D7C5D540              125                =C'$$BOPEN '
0002E8 5B5BC2E2C5E3C6D3              126                =C'$$BSETFL'
0002F0 5B5BC2C5D5C4C6D3              127                =C'$$BENDFL'
0002F8 5B5BC2C3D3D6E2C5              128                =C'$$BCLOSE'
000300 00000010                      129                =A(ISTST)
```

Appendix. Example Using LODIS (Part 4 of 6)



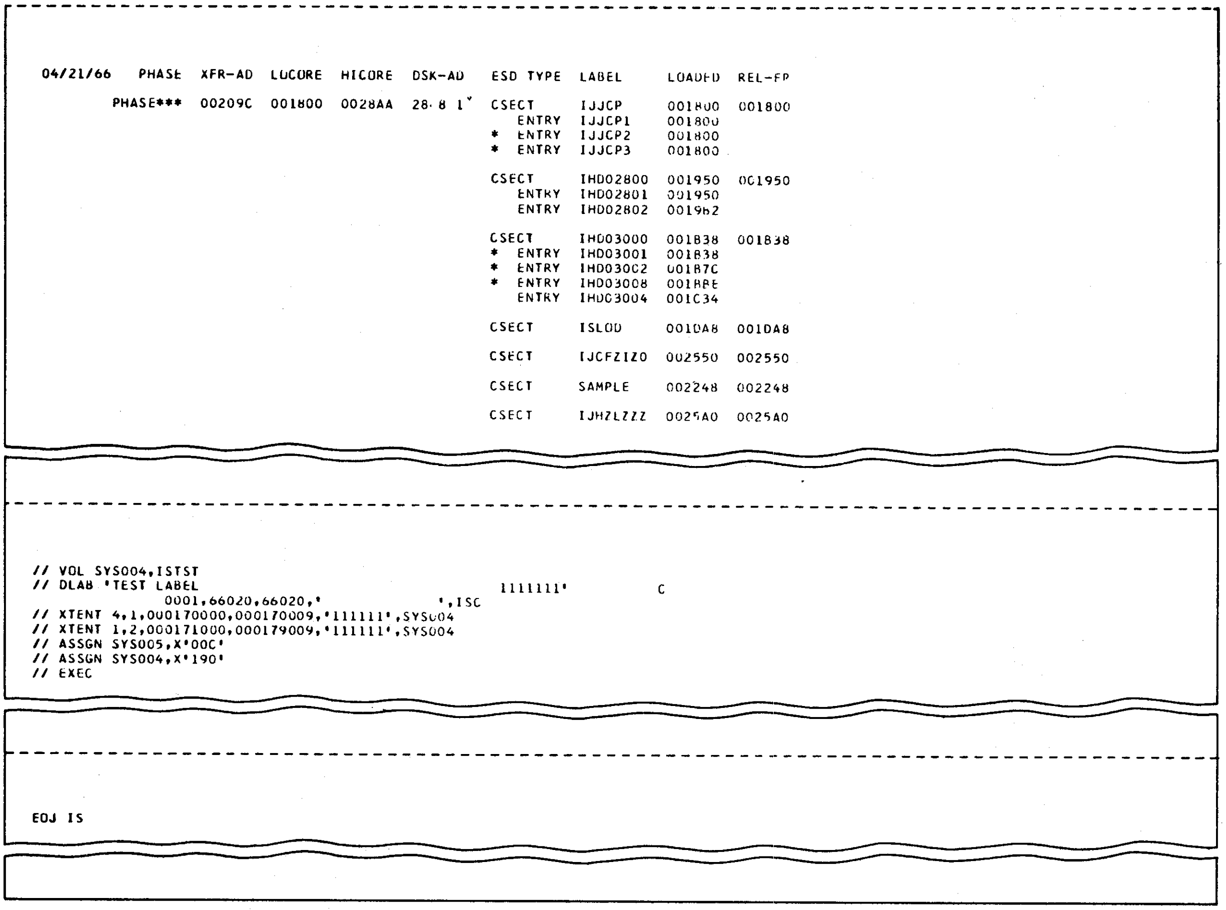
```
                                      RELOCATION DICTIONARY                                    PAGE    1

   POS.ID    REL.ID    FLAGS    ADDRESS

     01        01        0C      000018
     01        02        1C      000020
     01        01        0C      0000C8
     01        01        0C      0000CC
     01        01        0C      0000D0
     01        01        0C      000148
     01        01        0C      000190
     01        01        0C      000300
```

```
                                  CROSS-REFERENCE                                            PAGE    1

   SYMBOL     LEN  VALUE  DEFN

   IJJC0007 00002 000194 0088     0086
   IJJ00004 00002 00014C 0065     0063
   ISTST    00008 000010 0011     0019  0064  0070  0077  0087  0097  0101  0129
   ISTSTAA  00001 000114 0049     0008
   ISTSTAB  00001 00013E 0060     0055
   ISTSTAC  00001 00015A 0069     0067
   ISTSTAD  00001 000170 0076     0057
   ISTSTAE  00001 000186 0083     0081
   ISTSTAF  00001 0001A2 0092     0053
   ISTSTAG  00001 0001D6 0107     0068  0075  0082  0091  0106
   ISTSTAH  00001 0001F8 0116     0059  0074  0090  0105  0111  0114
   ISTSTAI  00005 000200 0121     0041  0042  0C93
   ISTSTAJ  00100 000205 0122     0094
   ISTSTAK  00113 000269 0123     0040
   ISTSTB   00008 000090 0039     0012
   ISTSTC   00002 00002E 0018     0051  0051  0066  0073  0080  0089  0104  0108  0110  0113
   ISTSTE   00004 000108 0046     0019
   ISTSTH   00008 000052 0030
   ISTSTM   00004 0000C8 0040
   ISTSTP   00024 000078 0038
   ISTSTS   00008 000070 0037
   SAMPLE   00001 000000 0002


   NO STATEMENTS FLAGGED IN THIS ASSEMBLY
```

```
   // OPTION LOG
   // EXEC LNKEDT
```

```
    JOB   IS        04/21/66   DISK LINKAGE EDITOR DIAGNOSTIC OF INPUT

    ACTION TAKEN     MAP
    LIST    INCLUDE IHD02800                                              ISL00001
    LIST    INCLUDE IJJCP
    LIST    INCLUDE IHD03000                                              ISL00002
    LIST    AUTOLINK    IJCFZIZO
    LIST    AUTOLINK    IJHZLZZZ
    LIST    ENTRY
```

Appendix. Example Using LODIS (Part 5 of 6)



```
04/21/66   PHASE   XFR-AD  LOCORE  HICORE  DSK-AD   ESD TYPE  LABEL     LOADED  REL-FR

        PHASE***  00209C  001800  0028AA  28 8 1   CSECT     IJJCP     001800  001800
                                                      ENTRY  IJJCP1    001800
                                                   *  ENTRY  IJJCP2    001800
                                                   *  ENTRY  IJJCP3    001800

                                                   CSECT     IHD02800  001950  001950
                                                      ENTRY  IHD02801  001950
                                                      ENTRY  IHD02802  001962

                                                   CSECT     IHD03000  001B38  001B38
                                                   *  ENTRY  IHD03001  001B38
                                                   *  ENTRY  IHD03002  001B7C
                                                   *  ENTRY  IHD03008  001BBE
                                                      ENTRY  IHD03004  001C34

                                                   CSECT     ISLOD     001DA8  001DA8

                                                   CSECT     IJCFZIZ0  002550  002550

                                                   CSECT     SAMPLE    002248  002248

                                                   CSECT     IJHZLZZZ  0025A0  0025A0
```

```
// VOL SYS004,ISTST
// DLAB 'TEST LABEL                                          1111111'         C
                 0001,66020,66020,'                 ',ISC
// XTENT 4,1,000170000,000170009,'111111',SYS004
// XTENT 1,2,000171000,000179009,'111111',SYS004
// ASSGN SYS005,X'00C'
// ASSGN SYS004,X'190'
// EXEC
```

```
EOJ IS
```

Appendix. Example Using LODIS (Part 6 of 6)

INDEX

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Disk Operating System
COBOL DASD Macros

GC24-5039-1

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM Branch Offices.

	Yes	No
● Does this publication meet your needs?	□	□
● Did you find the material:		
Easy to read and understand?	□	□
Organized for convenient use?	□	□
Complete?	□	□
Well illustrated?	□	□
Written for your technical level?	□	□

- What is your occupation? ____________________
- How do you use this publication?

As an introduction to the subject?	□	As an instructor in a class?	□
For advanced knowledge of the subject?	□	As a student in a class?	□
For information about operating procedures?	□	As a reference manual?	□

Other ____________________

- Please give specific page and line references with your comments when appropriate.

COMMENTS:

- Thank you for your cooperation. No postage necessary if mailed in the U. S. A.

GC24-5039-1

Staple

Fold Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P. O. Box 6
Endicott, N. Y. 13760

Attention: Programming Publications, Dept. 157

Fold Fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Additional Comments:

Cut Along Line