WYLBUR Edit Format Utility Package

January 2001

National Institutes of Health
Center for Information Technology
NIH Computer Center
12 South Drive MSC 5607
Bethesda, Maryland 20892-5607

Publication No. CIT004

Table of Contents

1 INTRODUCTION

This manual describes facilities at the NIH Computer Center, which is administered by the Center for Information Technology (CIT).

1.1 WYLBUR Facilities

WYLBUR is a computer software system that can be used as an online interactive text editor, document formatter, programmer's tool, and programming language. Users communicate with WYLBUR through network connections and workstations/terminals connected by telephone lines to the OS/390 systems at the NIH Computer Center.

WYLBUR's **editing facility** provides an easy-to-use mechanism for creating, changing, searching for, and displaying all kinds of text, such as computer programs and data, letters, proposals, reports, manuals, and lists. WYLBUR's **document formatting facility** allows users to format documents with even right margins, indexes, tables of contents, headings, footings, and automatic page numbering. WYLBUR's **batch processing facility** allows the user to submit a computer program to the batch job stream, and examine the output at the terminal. WYLBUR's **command procedure facility** provides commands to evaluate expressions, execute collections of stored commands and perform comparisons and branching. All of these facilities use predefined instructions or commands.

The version of WYLBUR described in this document, known as NIH WYLBUR, was completely designed, developed and implemented at the National Institutes of Health in Bethesda, Maryland. It is a successor to the original WYLBUR, which was developed at the Campus Facility of the Stanford University Computation Center. It has been modified and extended by the NIH Computer Center to meet the needs of the NIH user community.

This document describes a main program, associated cataloged procedure, and a collection of subroutines that can be used to read and write edit format data sets from programs executing in batch and TSO. Edit format is the standard format used by WYLBUR to store data sets. It is designed to conserve disk space by compressing strings of blanks. Each edit format line has an associated line number that is stored in the logical record, separately from the text of the line.

1.2 Using This Manual

This manual, the *Edit Format Utility Package*, completely defines the facilities available to read and write edit format data sets in batch jobs and TSO. It is organized to serve both as a learning text and a reference manual. This manual assumes that the user is familiar with the standards established by the Computer Center for running jobs. These standards are documented in the *NIH Computer Center User's Guide*. Users are encourage to read *Interface,* the technical newsletter that describes services and facilities provided by the Center for Information Technology to NIH and other government agencies.

1.3 Information Resources for WYLBUR Users

The Center for Information Technology offers publications, training, online documentation and telephone and personal consultation to help you learn more about WYLBUR. For more information, go to:

http://cit.nih.gov

Consulting is offered to all registered users, without charge, through the CIT Technical Assistance and Support Center (TASC). Contact TASC at 301-594-6248.

2 EDSUTIL MAIN PROGRAM

2.1 Program Description

The EDSUTIL program is used to copy or list sequential data sets. Edit format data sets may be converted to non-edit format and non-edit format data sets may be converted to edit format. DD statements are used to define the data sets to be copied or listed. Members of a partitioned data set may be processed by specifying the member name on the DD statement. Multiple copying operations may be used to combine multiple data sets and copy to one or more data sets. The data sets being copied or the data sets created by the copying mechanism may be listed. There are options provided to permit the user to describe the method used for handling line numbers. Also, the user may call programs to be executed, and upon completion of execution of each program, return control to the EDSUTIL program. This program uses the subroutines provided for users to handle edit format data sets (EDINIT, EDOPEN, EDGET, EDPUT, EDCLOS, EDTERM, etc.). Commands, which are described below, are used to specify the processing to be performed on the data set.

The main purpose of the EDSUTIL program is to convert data sets between edit and non-edit format. The program also allows copying data sets from edit to edit format, or from non-edit to non-edit format. The EDSUTIL program may be used to change the record format when copying non-edit format data sets. In general, however, it may be more efficient to use programs other than EDSUTIL when not converting between edit and non-edit format. This is because EDSUTIL handles records on a logical record basis, which is not as efficient as a program which handles records on a physical block basis.

2.2 Job Control Language (JCL)

Each data set referenced by a command must have a DD statement supplied by the user (or, in the case of TSO, an appropriate ALLOCATE command must be issued). The ddname specified for a data set on the DD statement is used in the commands to indicate the data set to be processed. If a data set to be accessed by EDSUTIL has the DCB attributes of RECFM=U and BLKSIZE greater than or equal to 1022, it is assumed to be an edit format data set. In the case of writing a data set, if no DCB parameters are specified on the associated DD statement and the data set is not directed to a SYSOUT class, the data set is written in edit format.

A DD statement should be included in the JCL to obtain a listing of the commands specified for the EDSUTIL program and to obtain messages issued by the EDSUTIL program. The default ddname of this DD statement is SYSPRINT. The PARM field (described below) may be used to specify another ddname. If there is no DD statement provided for the listing, and an error occurs during EDSUTIL processing, an NIH2001 message describing the error will be issued, followed by an ABEND with a user completion code of 2001.

The default ddname of the input data set containing the commands is SYSIN. This may be overridden by specifying another input ddname in the PARM field (described below). EDSUTIL assumes that line numbers are stored in the data set containing the commands. For

non-edit format data sets, the columns where line numbers are stored depends on the format of the data set:

F format	last 8 positions of each record
U format	first 8 positions of each record
V format	first 8 positions of each record

To find out how to specify that line numbers are not stored in the data set containing the commands, see the description of the PARM field.

The EDSUTIL program sets step completion codes indicating the results of its processing:

0	Processing completed normally.
4	Processing completed. However, some minor errors occurred. Messages were produced describing the conditions.
8	Processing completed. However, some errors that may be serious have occurred. Messages were produced describing the conditions.
12	A command was terminated before it was completed. Messages were produced describing the error.

2.3 Commands

Commands are used to provide instructions for the EDSUTIL program. The general format of the commands follows the syntax of WYLBUR commands.

Comments may be used in commands by beginning the comment with a percent sign (%). The comment is terminated by the end of the line. A blank or empty line is treated as a comment line.

More than one command may be included in a line by separating the commands with a semicolon (;). A command may be continued by including two percent signs (%%) at the location where the command is being split. The next line is used as the continuation line unless it is a blank line or a comment line. Any information that is on the same line and follows the two percent signs (%%) is treated as a comment. The percent signs must appear after a complete option; that is, if an option of a command consists of more than one word, the percents must follow the last word in the option.

The abbreviations used in WYLBUR may also be used to abbreviate any of the commands or options for the EDSUTIL program.

2.3.1 COPY Command

The COPY command is used to copy a data set to another data set. Conversion to and from edit format is supported. A data set is designated by specifying the ddname associated with the DD statement that defines the data set. The general format of the COPY command is:

```
COPY DDNAME=input-ddname TO DDNAME=output-ddname
```

where "input-ddname" is the ddname of the DD statement defining the data set to be copied and "output-ddname" is the ddname of the DD statement defining the data set to be written. For example, to copy a data set with a ddname of INPUT, to a data set with the ddname OUTPUT, the following command would be used:

```
COPY DDNAME=INPUT TO DDNAME=OUTPUT
```

If DISP=MOD is used for a data set being written in edit format, the user must insure that the line numbers of the resulting data set will all be in ascending sequence.

If the data set being written is a non-edit format data set, no line numbers will be included in the output data set unless the NUMBERED, TSO, IBM, NUMBER or TIMES options (see below) are specified.

More than one data set may be copied by including the ddnames of the data sets, separated by the word AND, following COPY and preceding the TO portion of the command. When copying multiple data sets to one edit format data set, the user must insure that the line numbers for the data set written are in ascending sequence. In particular, careful attention should be made when more than one of the data sets being copied is in edit format. There are options described below which provide mechanisms for handling line numbers.

In the following example, the COPY command instructs the EDSUTIL program to copy the contents of the data set with a ddname of IN1, and then the contents of the data set with a ddname of IN2, to a data set with the ddname OUTPUT.

```
COPY DDNAME=IN1 AND DDNAME=IN2 TO DDNAME=OUTPUT
```

To copy data sets to more than one output data set, the data sets to be written are specified by including the ddnames of the output data sets, separated by AND, following the word TO. For example, to copy a data set with the ddname INPUT to a data set with the ddname OUT1 and to a data set with the ddname OUT2, the following command could be used:

```
COPY DDNAME=INPUT TO DDNAME=OUT1 AND DDNAME=OUT2
```

There are a number of options available to indicate special handling of the data sets. These options must be specified after the ddname of each applicable data set. The options may be specified in any order. If an option is repeated or if two incompatible options are coded, then only the last one encountered is used.

LIST

The contents of the data set are to be listed. Only the input or the output can be listed; input and output data sets cannot both be listed.

The listing is directed to the data set specified by the SYSPRINT DD statement or another output data set specified in the PARM field (described below).

If line numbers are part of the data set being listed, they are included in the listing. The line number appears to the left of the text of the line and is separated from the text by two blanks.

A number of options are available to specify how the listing is to be performed. These options, described in the LIST command (see below), are:

UNNUMBERED
CC
MC
INDENT n
DOUBLE
TRIPLE
SPACING n
MARKER c

NUMBERED column/column

Indicates that a line number is stored in each record of a non-edit format data set. The line numbers are located in the indicated column positions of each record. The column positions are specified by separating the first and last columns with a slash (/).

The columns may be omitted, in which case the location of the line numbers depends on the record format of the data set:

F format -- last 8 positions of the record
U format -- first 8 positions of the record
V format -- first 8 positions of the record

The numbers located in the specified positions are used as the line number of each record in the data set. Any valid WYLBUR absolute line number, with or without the period, is acceptable. If the data set is being copied, the line number is removed as each line is copied. If the data set is being written, the text in the line is moved to the right to provide positions for the line number to be stored, unless the MERGE or OVERLAY options are specified. The text of a line will be truncated on the right, if necessary, to

	make the length of the line equal to the record length. If more positions are required to write a line number, the line number is truncated on the left.
TSO column/column	The TSO option is the same as the NUMBERED option except that the line number in the specified positions does not contain a decimal point (.) and is in TSO (IBM) format. To convert a WYLBUR line number to TSO format the line number is padded on the right with zeroes, if necessary, to obtain three digits to the right of the decimal point, and is padded on the left with zeroes, if necessary, to obtain five digits to the left of the decimal point. The decimal point is then removed. To convert from TSO format to WYLBUR format, the decimal point is inserted after the fifth digit, and if appropriate, zeroes are removed from the right and left portions of the number.
IBM column/column	Synonym for TSO.
INSERT	Used with the NUMBERED, TSO or IBM options when a data set is being written to indicate that the text in and following the columns where the line number is to be stored is to be moved to the right to provide positions for the line number. If the NUMBERED, TSO or IBM option is not specified, NUMBERED is assumed. INSERT is the default line number insertion technique.
MERGE	Used with the NUMBERED, TSO or IBM options when a data set is being written to indicate that the line number is to be placed in each line only if there are only blanks in the columns where the line number is to be placed. The blanks are to be replaced with the line number. If the NUMBERED, TSO or IBM option is not specified, NUMBERED is assumed.
OVERLAY	Used with the NUMBERED, TSO or IBM options when a data set is being written to indicate that the line numbers are to replace any text in the columns where the line numbers are to be stored. If the NUMBERED, TSO or IBM option is not specified, NUMBERED is assumed.
NUMBER START line-number BY increment	NUMBER specifies how line numbers are to be generated for each line in the data set. The starting line number follows the word START. The word START may be omitted. If the START option is omitted, a starting line number of 1 is used. The increment used to generate succeeding line numbers is chosen by using a 1 in the last decimal

position of the starting line number.

The BY option may be used to specify the increment to be used to generate succeeding line numbers. The increment may range from .001 to 99999.999. If the START option is not specified and the BY option is used, the starting line number is set to the value of the increment.

If the NUMBERED or TSO options are also used, the positions specified by the NUMBERED or TSO option are removed and the line numbers are generated by following the method established by the number option.

When writing a data set, and the NUMBERED, TSO or IBM options are not specified, the NUMBERED option is assumed.

The NUMBER and TIMES options are mutually exclusive.

TIMES factor — Specifies that the line numbers, either generated or stored in each record of the data set, are multiplied by the specified factor before they are used. The factor may range from .001 to 99999.999.

The TIMES and NUMBER options are mutually exclusive.

SKIP integer — Specifies the number of lines to be skipped before a line is read or written.

TAKE integer — Specifies the number of lines to be read or written.

RECFM U — Indicates that the data set has edit format characteristics (RECFM is U and BLKSIZE is greater than or equal to 1022), but the data set is not to be treated as an edit format data set.

The following examples illustrate some uses of these options:

```
COPY DDNAME=INPUT TAKE 100 TO DDNAME=OUTPUT
```

The first 100 records of the data set with a ddname INPUT are copied to a data set with the ddname OUTPUT.

```
COPY DDNAME=IN1 LIST AND DDNAME=IN2 LIST TO DDNAME=OUTPUT NUMBER
```

The contents of data sets with ddnames IN1 and IN2 are listed to the data set designated by the SYSPRINT DD statement. The data set being written, with a ddname OUTPUT, contains the records from the data set with the ddname IN1 followed by the records from the data set with the ddname IN2. Because the NUMBER option was specified, the line

numbers of each record in the data set written are generated by using a starting line number of 1 and an increment of 1.

2.3.2 LIST Command

The LIST command is used to obtain a listing of a data set. The listing is directed to the data set specified by the SYSPRINT DD statement or the output data set specified in the PARM field (described below). If line numbers are being listed with the data set, the line number is separated from the text of the line by two blanks. A data set is designated by specifying the ddname associated with the DD statement which defines the data set following the word LIST. For example, to list the contents of the data set with a ddname of DATA, the following command would be used:

```
LIST DDNAME=DATA
```

More than one data set may be listed by including the ddnames of the data sets, separated by AND, following the word LIST. The data sets are listed in the order in which they appear in the command. For example, to list the data sets with ddnames PROGRAM and DATA, the following command could be used.

```
LIST DDNAME=PROGRAM AND DDNAME=DATA
```

The TO DDNAME=ddname option may be used to direct the listing to a specified data set. For example, the following command could be used to list the data set with a ddname of INPUT to the data set defined by a DD statement LISTING:

```
LIST DDNAME=INPUT TO DDNAME=LISTING
```

To direct the listing to more than one data set, the data sets to receive the listing are specified by including the ddnames of these data sets, separated by AND, following the word TO. For example, to list the data set with a ddname of INPUT to a data set with the ddname of LIST1 and to a data set with the ddname of LIST2, the following command could be used:

```
LIST DDNAME=INPUT TO DDNAME=LIST1 AND DDNAME=LIST2
```

There are a number of additional options available that may be specified for each output data set referenced to indicate instructions for how the data set listings are to be done. These options must be specified after the ddname of each applicable data set.

UNNUMBERED	Line numbers are not listed .
CC	The data set is processed assuming it contains ANSI carriage control characters in the first position of each line. For more information on ANSI carriage control, see the Users Guide.

MC	The data set is processed assuming it contains machine carriage control characters in the first position of each line. For more information on machine carriage control, see the Users Guide.
INDENT n	Indicates the number of blanks to be inserted before the text of each line is listed. If line numbers are being listed with the data set, the blanks are inserted before the line number.
DOUBLE	The output is to be double spaced (i.e., one blank line before each line from the data set). To obtain this result, ANSI carriage control characters are inserted in each line. For more information on ANSI carriage control, see the *NIH Computer Center User's Guide*.
TRIPLE	The output is to be triple spaced (i.e., two blank lines before each line from the data set). To obtain this result, ANSI carriage control characters are inserted before each line. For more information on ANSI carriage control, see the *NIH Computer Center User's Guide*.
SPACING n	SPACING n, where "n" is a positive integer, indicates that n-1 blank lines should appear before each line in the data set. To obtain this result, ANSI carriage control characters are used. For more information on ANSI carriage control, see the *NIH Computer Center User's Guide*.
MARKER c	MARKER c, where "c" is a character, indicates that an eject to a new page should be performed if a line is encountered that only contains the character "c" in the first position. A line that contains a "c" in the first position and other characters in the line is listed. To perform the functions provided by this option, ANSI carriage control characters are inserted before each line. The marker character may be enclosed in either single or double quotation marks. For more information on ANSI carriage control, see the *NIH Computer Center User's Guide*.

The following options, described above for the COPY command, may be used to specify the characteristics of the input data sets:

NUMBERED column/column
TSO column/column
IBM column/column
NUMBER START line-number BY increment
TIMES factor
SKIP integer
TAKE integer

RECFM U

2.3.3 CALL Command

The CALL command is used to execute another main program. When the called program completes execution, control is returned to the EDSUTIL program and commands following the CALL command are executed. The name of the program to be executed follows the word CALL. A parameter string may be passed to the called program by specifying a series of characters, representing the parameter, enclosed in single or double quotation marks following the word WITH. WITH appears after the name of the program. The syntax of the CALL command is:

```
CALL program-name WITH 'string'
```

The ABORT option is available on the CALL command to permit specifying a condition to stop execution of the EDSUTIL program. The condition checked is the value of the return code from the program called. The syntax of the ABORT option is

```
ABORT IF RC relational-operator integer
```

The valid relational-operators are:

Operator	**Meaning**
EQ	Equal
NEQ (or NE)	Not equal
LT	Less than (same as NGE)
NLT	Not less than (same as GE)
LE	Less than or equal (same as NGT)
NLE	Not less than or equal (same as GT)
GE	Greater than or equal (same as NLT)
NGE	Not greater than or equal (same as LT)
GT	Greater than (same as NLE)
NGT	Not greater than (same as LE)

The return code is compared to the integer, using the relational operator specified. If the comparison is true, the EDSUTIL program stops execution. For example, if the following command were issued, the EDSUTIL program would stop execution if the program named CALC returned a code greater than or equal to 8:

```
CALL CALC ABORT IF RC GE 8
```

Only the word ABORT, optionally followed by the relational-operator and the integer are required. If both the relational-operator and the integer are omitted, then ABORT IF RC GT 0 is assumed.

The job control language required for the EDSUTIL program and for the other program to be executed must be included in the job.

The amount of memory (core storage) required by the EDSUTIL program is approximately 120K bytes. If the user calls another program, the amount of storage requested must include the storage required by the called program plus the 120K bytes required by the EDSUTIL program, plus the buffer space required for data sets defined by the user.

2.4 PARM Field

The input statements to the EDSUTIL program are provided by the commands described above. The ddnames of the DD statements that describe where the commands are located and where they are to be listed may be specified in the PARM field. The following options may be included in the PARM field:

SYSIN=in-ddn	The ddname of the DD statement that describes where commands are located. The default is to assume a ddname of SYSIN.
SYSPRINT=out-ddn	The ddname of the DD statement that describes where the commands will be listed and where messages issued by the EDSUTIL program will be printed. The default is to assume a ddname of SYSPRINT.
UNNUMBERED	The data set containing the commands does not contain line numbers. The default is to assume that the data set contains line numbers.
ABEND	The data management ABENDs (e.g., x13, x14, x37 ABENDs) are not to be intercepted. The default is to intercept data management ABENDs and to print a message indicating the abend.
MESSAGES=n	After the same message is printed n times, that message will no longer be printed. A summary of the number of times each message is suppressed will be listed. The only messages that can be suppressed are those that may occur many times -- for example, messages describing errors that occur while reading or writing a data set. Messages that generally do not appear in abundance (e.g., syntax error messages) are never suppressed. The word INFINITY can follow MESSAGES to indicate that no messages will ever be suppressed. The default is to suppress messages after they have printed 10 times (i.e., MESSAGES 10).
TERMINATE=m	Terminates a command when the same message has been

	printed m times. If a command is terminated, processing will continue with the next command. The only messages for which a count is kept are those that may potentially occur numerous times -- for example, messages describing errors that occur while reading or writing a data set. Messages that generally do not appear in abundance (e.g., syntax error messages) are not counted. The word INIFINITY can follow TERMINATE to indicate that a command will never be terminated because of the number of messages. The default is INFINITY.
WIDTH=w	WIDTH=w, where "w" is a positive integer, indicates that a maximum of w characters per line are to be printed to the DD statement specified by the SYSPRINT= option. A line containing more than w characters is automatically carried over onto the following line ("wraparound"). The value of w includes the carriage control character. The default value is 133 and the maximum is 1000.

One command may be included in the PARM field by specifying the command, preceded by a slash (/), as the last option in the PARM field. The slash is required preceding the command, even though the other options may be omitted. A command included in the PARM field is executed before any commands in the SYSIN data set. The general format of the PARM field is:

```
PARM='SYSIN=in-ddn,SYSPRINT=out- ddn,UNNUMBERED,ABEND,MESSAGES=n,
TERMINATE=m,WIDTH=w/cmd'
```

2.5 EDSUTIL Procedure

The EDSUTIL procedure is defined to provide some of the job control language necessary to execute the EDSUTIL program.

The following symbolic parameters available in the EDSUTIL procedure provide a method for specifying the options available in the PARM field. These symbolic parameters are optional.

Symbolic Parameter	**Option**	**Function**
UTILOPT		Used to specify the SYSIN, UNNUMBERED, SYSPRINT, ABEND, MESSAGES, TERMINATE, and WIDTH PARM field options. More than one option may be used by separating the options by blanks or commas.
	SYSIN	Code UTILOPT='SYSIN=input-ddname' to specify the ddname of the DD statement defining the data set containing the commands to the EDSUTIL program. If used, a DD statement with the specified ddname must

be included.

The default ddname is SYSIN, defining the system inputstream. A SYSIN DD statement may be included to define a different input command data set.

UNNUMBERED — Code UTILOPT='UNNUMBERED' to indicate that the data set containing the commands does not contain line numbers.

The default is to assume that the command input data set contains line numbers.

SYSPRINT — Code UTILOPT='SYSPRINT=output-ddname' to specify the ddname of the DD statement defining the data set where the commands will be listed and where messages issued by the EDSUTIL program will be printed. If used, a DD statement with the specified ddname must be included.

The default ddname is SYSPRINT, defining the standard printer. The SYSPRINT DD statement in the procedure may be overridden to define another data set to receive messages. If added, the SYSPRINT DD statement must appear before any additional DD statements being supplied.

ABEND — Code UTILOPT='ABEND' to indicate that data management ABENDs (e.g., x13, x14, x37ABEND) are not to be intercepted.

The default is to intercept data management ABENDs and issue a message indicating the error.

The default is to suppress each message after it has been printed 10 times.

MESSAGES — Code UTILOPT='MESSAGES=n' to indicate the number of times the same messages is to be listed. Code MESSAGES=INFINITY to list all of the messages.

The default is to suppress each message after it has been printed 10 times.

	TERMINATE	Code UTILOPT='TERMINATE=n' to indicate that a command is to terminate after the same message has been generated n times. If a command is terminated, processing will continue with the next command. Code TERMINATE=INFINITY to never terminate a command because of the number of messages. The default is to never terminate a command because of the number of messages (i.e., INFINITY).
	WIDTH	Code UTILOPT='WIDTH=n' to indicate that a maximum of n characters per line are to be printed to the DD statement specified by the SYSPRINT= option. A line containing more than n characters is automatically carried over onto the following line ("wraparound"). The value of n includes the carriage control character. The default is to wrap to a new line after 133 characters have been printed.
COMMAND		To specify the first command to be executed before the commands in the command input data set, use COMMAND='command'. The default is to assume that the command input data set contains all the commands.

2.6 Examples

To obtain a listing of a cataloged edit format data set named AAAAIII.TEXT, the following job could be executed. The job uses the EDSUTIL procedure and the symbolic parameter COMMAND to specify the command. A DD statement with the name TEXT defines the data set to be listed.

```
//     (Class A JOB Statement)
//         EXEC EDSUTIL,COMMAND='LIST DDNAME=TEXT'
//TEXT      DD  DSNAME=AAAAIII.TEXT,DISP=SHR
```

The following example copies an edit format data set to a non-edit format data set and produces a listing of the copied data set. The procedure EDSUTIL is used. The ddname of the data set to be copied is IN and the ddname of the data set to be written is OUT.

```
//         EXEC EDSUTIL
//IN        DD  DSNAME=AAAAIII.IN,DISP=SHR
//OUT       DD  DSNAME=AAAAIII.OUT,UNIT=TMP,
//              DISP=(NEW,CATLG),
```

```
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=11440),
//             SPACE=(TRK,(10,5),RLSE)
COPY  DDNAME=IN TO DDNAME=OUT LIST
```

In the previous example, the COPY command is the only command given to the EDSUTIL program. The command is specified in the SYSIN input data set (the system generates a //SYSIN DD * statement when one is not present). The same result could have been obtained by using the COMMAND symbolic parameter to specify the command instead of using the input data set.

The next example illustrates the use of the EDSUTIL program to convert an edit format data set to a non-edit format data set, call a program that reads the data set copied and writes an output data set, and then converts the data set written by the program to an edit format data set. The following information is useful in understanding the example:

- The STEPLIB DD statement informs the system where the program called by the CALL command is stored.
- The SYSPRINT DD statement designates that the output will be printed on the standard printers. This is also where all messages from the EDSUTIL program will be listed.
- The first command copies the data set designated on the DD statement INPUT to the temporary data set &&INPUT as specified on the DD statement FT01F001. This data set is converted from edit format to non-edit format as it is copied. Line numbers are not included in the output, non-edit format, data set.
- The program CALC is called and reads input from the temporary data set &&INPUT. Output is written to the non-edit format temporary data set &&OUTPUT as specified on the DD statement FT22F001.
- The temporary data set &&OUTPUT is copied to an edit format data set as specified on the OUTPUT DD statement.
- The last command produces a listing of the original input data set and the final output data set. The listing is printed on the standard printers as described by the SYSPRINT DD statement
- The FT06F001 DD statement is included to obtain errors occurring during the execution of the Fortran program CALC, and the SYSUDUMP DD statement is included to obtain a system dump in case a system ABEND error condition arises during the execution of this job step.

```
//        EXEC PGM=EDSUTIL,PARM='SYSIN=CMDIN'
//STEPLIB  DD  DSNAME=AAAAIII.PGM,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//FT01F001 DD  DSNAME=&&INPUT,UNIT=SYSDA,DISP=(NEW,DELETE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=11440),
//             SPACE=(CYL,(1,1))
//FT15F001 DD  SYSOUT=A
```

```
//FT22F001 DD  DSNAME=&&OUTPUT,
//             UNIT=SYSDA,DISP=(NEW,DELETE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=11440),
//             SPACE=(CYL,(1,1))
//INPUT    DD  DSNAME=AAAAIII.INPUT,DISP=SHR
//OUTPUT   DD  DSNAME=AAAAIII.OUTPUT,UNIT=TMP,
//             DISP=(NEW,CATLG),SPACE=(TRK,(10,5),RLSE)
//SYSUDUMP DD  SYSOUT=A
//CMDIN    DD  *
COPY DDNAME=INPUT TO DDNAME=FT01F001
CALL CALC
COPY DDNAME=FT22F001 TO DDNAME=OUTPUT
LIST DDNAME=INPUT AND DDNAME=OUTPUT
```

2.7 Abbreviations

Word	Abbreviations
AND	&
BY	
CALL	
CC	
COPY	COP, CPY
DDNAME	DD, DDN
DOUBLE	DBL
IBM	
INDENT	IND
INFINITY	INF
INSERT	INS
LIST	LIS, (L)
MARKER	MAR, MARK
MERGE	
MESSAGES	MSGS
MC	
NUMBER	NUM
NUMBERED	NUMD
RECFM	
SKIP	
SPACING	SPN
START	
TAKE	
TERMINATE	TERM
TIMES	
TO	
TRIPLE	TRI, TPL
TSO	

UNNUMBERED	UNN
WIDTH	WID
WITH	

Abbreviations in parenthesis are only valid for commands, not options.

3 EDIT FORMAT DATA SET (ED) ROUTINES

3.1 General Information

These subroutines provide a mechanism for programs to directly read and write edit and non-edit format sequential data sets and members of partitioned data sets. The ability to handle both edit and non-edit format data sets provides the user with the flexibility of not needing to know the format of the data set to be accessed. If only non-edit format data sets are to be handled, the facilities available in the programming language should be used instead of these routines.

These subroutines are callable from any of the major languages supported at the Computer Center (i.e., Assembler Language, COBOL, FORTRAN, and PL/I). These subroutines are stored in the cataloged partitioned data set named NIH.UTILITY. To allow the use of PL/I character string variables, a separate set of subroutines has been provided that may only be called from PL/I. These subroutines are denoted by the letter "P" suffixed to the name of the subroutine. For example, programs written in Assembler Language, COBOL, or FORTRAN would call subroutine EDINIT, whereas a PL/I program must call EDINITP. For the most part, the calling sequences for the subroutines with and without the "P" suffix are exactly the same. The only difference is that the subroutines with a "P" suffix take advantage of the fact that PL/I keeps track of the length of character strings for the programmer. Consequently, it is not necessary for the PL/I user to pass this length as an argument. Since the documentation for the PL/I callable subroutines is nearly identical to that for the non-PL/I callable subroutines, a special notation is used in this document to indicate both subroutine names. For example, EDINIT(P) refers to both EDINIT (which is callable from Assembler Language, COBOL, and FORTRAN), and EDINITP (which is callable from PL/I).

3.1.1 Summary of ED Subroutines

The subroutines are divided into two groups: those that are required, and those which provide optional functions.

Required Subroutines

EDINIT(P)—This subroutine is called first for each data set to be read or written. It establishes an internal work area for the other subroutines.

EDOPEN(P)— This subroutine opens a data set for processing.

EDGET(P)— This subroutine is called to read a logical record from a data set. If the records read are from an edit format data set, they are converted to non-edit format.

EDPUT(P)— This subroutine is called to write a logical record into a data set. If the data set has edit format characteristics, the records are converted to edit format as they are written.

EDMSG(P)— This subroutine returns the text of an error message associated with the return code returned by the previous subroutine call.

EDCLOS(P)— This subroutine is called for each data set processed in order to properly close the data set.

EDTERM(P)— This subroutine is called to terminate processing and free the work area established by EDINIT(P).

Optional Subroutines

EDNCOL(P)—This subroutine is used to specify the column positions for line numbers in a non-edit format data set.

EDNGEN(P)—This subroutine is used to designate how line numbers are to be generated for each line in the data set.

EDSET(P)—This subroutine is used to set information about the data set (e.g., RECFM of the data set).

EDSHOW(P)— This subroutine is used to show information (e.g., RECFM of the data set).

EDGETL(P)— This subroutine is a version of EDGET(P) that returns the address of the location where the characters of the line read are stored.

EDNCNV(P)— This subroutine converts line numbers between integer and character representation.

3.1.2 Arguments

Common Declarations

Many of these subroutines have arguments that must be declared to be binary full word (4 byte) integer (fixed point) variables. In this document, these variables will be referred to as simply integer variables. To declare an integer variable named VAR, the following could be used:

Language	**Declaration**
COBOL	VAR PIC S9(8) COMPUTATIONAL
FORTRAN	INTEGER*4 VAR
PL/I	DECLARE VAR FIXED BINARY(31)
Assemble	VAR DS F

Arguments containing character strings are also used. The following is one method which may be used to declare these variables. VAR is used as the variable name and "n" is the number of characters in the character string.

Language	Declaration
COBOL	VAR PIC X(n)
FORTRAN	LOGICAL*1 VAR(n) or DIMENSION VAR(m) where m is the smallest integer that is greater than or equal to n divided by 4
PL/I	DECLARE VAR CHARACTER(n) VARYING or DECLARE VAR CHARACTER(n)
Assembler	VAR DS CLn

For PL/I, to copy declarations for all subroutine entry point and argument types, use %INCLUDE EDDCLS statements where the entry points would normally be declared. If this is done when using the standard PL/I compile procedures (i.e., PLIXCOMP), several additions must be made in the EXEC statement. When using the PL/I Optimizing compiler (i.e., PLIXCOMP), add OPTIONS=INCLUDE and LIBNAME='NIH.MACLIB'.

Control Word
All subroutines have as a first argument a control word, referred to in this documentation as CNTRL. CNTRL is used to reference a work area that is acquired by subroutine EDINIT(P).

CNTRL must be an integer variable. The call to EDINIT(P) stores a value in the location referenced by CNTRL, which is used to identify the work area established for the data set being processed. The value of CNTRL is used by other subroutines to identify the proper work area for the data set being processed.

If the user fails to call EDINIT(P) to initialize the variable CNTRL, or if the contents of CNTRL is altered after the call to EDINIT(P), the next subroutine call using the variable CNTRL will cause the program to ABEND. If the program uses linkage editor overlays, CNTRL should be in the root segment (segment 1).

Return Code
The second argument for all subroutines is a return code, referred to in this documentation as RETURN. This argument is used to return information concerning the processing of the subroutine. RETURN must be an integer variable. The subroutine stores an integer in the variable RETURN that gives the result of the subroutine processing.

Return Code Range	Meaning
0	Normal completion
1 - 99	Informational only
100 – 199	Warning; processing may continue
200 – 299	Error; processing may continue
300 – 399	Error; further processing of the same data set will cause an ABEND

3.1.3 Line Numbers

Each line in an edit format data set has a line number. Some of the subroutines reference these line numbers. There are two ways that line numbers are represented:

Character string form of a line number

WYLBUR line number

This number is represented by 0 to 5 digits, a decimal point, and 0 to 3 digits. At least one digit must be specified. If the line number is an integer, the decimal point may be omitted. Leading blanks and zeroes are ignored. Trailing blanks are ignored; trailing zeroes are treated as digits. It is an error if blanks appear in the number. Examples of valid WYLBUR line numbers: 53.9, 112, 0.781, .3

TSO (IBM) line number

This number is formed by taking the WYLBUR representation, multiplying it by 1000, and optionally padding it with zeroes or blanks on the left. Leading zeroes and blanks are ignored. Examples of valid TSO (IBM) line numbers: 00053900, 112000, 00781, 00000300. Notice that these numbers are the same ones used in the example of WYLBUR line numbers.

Character string line numbers are passed to and from the subroutines in character string arguments.

Binary integer form of a line number

The binary integer form of a line number is created by taking the WYLBUR character string form, multiplying it by 1000, and converting it to a binary full word (4 byte) integer.

Binary integer line numbers are passed to and from the subroutines in integer variable arguments.

3.1.4 ABEND Conditions

If an error is encountered for which it is unreasonable to expect a user's program to be able to recover, the subroutine will ABEND. The ABEND will have a user code of 2001. The subroutine will also issue a message describing the error. The message will have an identification number of NIH2001. For a batch program, the NIH2001 message will be displayed on the JES2 job log; for a program executing under TSO, the message will be displayed on the terminal.

3.2 SUBROUTINE EDINIT(P)

Purpose: This subroutine establishes and initializes an internal work area for the subroutines that process the data set. It must be called before any of the other subroutines.

Arguments: CNTRL, RETURN

Name	**Function**
CNTRL	A control word.
RETURN	Return code. The possible values are: 0 Processing completed normally.

3.3 SUBROUTINE EDOPEN(P)

Purpose: This subroutine opens the data set for processing. A DD statement for the data set must be included in the JCL for the job step (or, an appropriate ALLOCATE command issued under TSO).

EDOPEN(P) acquires an additional internal work area which includes buffers where the records read or written are stored. The size of these buffers depends on the block size associated with the data set, and thus the amount of storage used by this subroutine varies according to the block size of the data set being processed.

A data set that has the DCB attributes RECFM=U and BLKSIZE greater than or equal to 1022 is assumed to be an edit format data set. The EDSET(P) routine may be called to specify that a data set with these characteristics is to be treated as a non-edit format data set.

In the case of writing a data set, the data set will be in edit format unless DCB parameters are specified on the associated DD statement or via the EDSET(P) subroutine, or the data set is directed to a SYSOUT class. Additional considerations also apply when writing a non-edit data set. When requesting a record format other than RECFM=U, the user must also specify a value for LRECL. For blocked data sets when no block size is specified, a block size less than or equal to 11476 will be chosen which is compatible with the RECFM and LRECL specification. If the data set is directed to a SYSOUT class and no record format is specified, RECFM=U is assumed; if no block size is specified, a block size of 254 is used.

Arguments: CNTRL,RETURN,DDNAME,TYPE

NAME	FUNCTION
CNTRL	The same control word that was specified when EDINIT(P) was called.
RETURN	Return code. The possible values are:

100	Subroutine EDNCOL(P) was called to specify the column positions of line numbers for a non-edit format data set. However, the data set opened was edit format. The call to EDNCOL(P) will be ignored.
200	Column positions specified by a previous call to EDNCOL(P) are inconsistent with the attributes of the data set. The information set by EDNCOL(P) is ignored.
300	An ABEND has been intercepted. The intercepted IBM ABEND code may be obtained by calling EDSHOW(P). A descriptive message may be obtained by calling EDMSG(P). The data set was not opened.
301	The data set was not opened. There is no DD statement for the data set referenced.

DDNAME — The DDNAME associated with the DD statement of the data set to be processed must be stored in this 8 byte character variable. The DDNAME must be padded on the right with blanks, if necessary, to completely fill the area.

For subroutine EDOPENP, this argument must be a PL/I character string variable and need not be padded with blanks.

TYPE — An integer indicating the type of processing to be performed must be stored into this integer variable before EDOPEN(P) is called. The acceptable values are:

1	A data set is to be read.
101	A data set is to be written.

102	Records are to be added at the end of an existing data set, without using a disposition of MOD on the data set's DD statement.

Notes:

If the value of TYPE is invalid or an attempt is made to open a data set when another data set is already open using the same control word, the subroutine will ABEND.

3.4 SUBROUTINE EDGET(P)

Purpose: This subroutine is called to read a logical record from a data set. If the records read are from an edit format data set, they are converted to non-edit format. The subroutine assumes that the data set has already been opened by the EDOPEN(P) subroutine. The records are read in sequential order.

Arguments: CNTRL,RETURN,LINENO,LINE,LENMAX,LENACT

Name	Function
CNTRL	The same control word that was specified when the data set was opened by the EDOPEN(P) subroutine call.
RETURN	Return code. Unless otherwise documented, a record is returned. The possible values are:

0	Processing completed normally.
1	There are no more records in the data set to be read (an end of file condition was encountered). No record is returned. The EDCLOS(P) subroutine should be called to close the data set properly. If no more data sets are to be processed, then EDTERM(P) should be called to release work area storage for use by the remainder of the program.
100	The record was truncated on the right to a length equal to the number of characters specified by the value of LENMAX (or the length of the PL/I character string variable when EDGETP is called). For EDGET, the actual length of the record, after truncation, is returned in LENACT. The length of the record before truncation can be obtained by calling EDSHOW(P).
101	A concatenated data set with attributes different from the previously processed data set was opened. One of the following changes occurred: (1) the maximum record length changed, (2) RECFM changed, (3) the format

changed between edit and non-edit format. No record from the data set is returned. The next call to EDGET(P) will process records starting with the first record in the concatenated data set. The maximum record size may be obtained by calling EDSHOW(P). Previous calls to EDNCOL(P) are no longer used.

102 When opening a concatenated data set, the line number of the first line is less than the line number of the last line from the previous data set. To obtain the value of the previous line number, call EDSHOW(P). The line number obtained by the next call to EDGET(P) will be compared against the line number returned from this call to EDGET(P).

200 An I/O error occurred. The record in error is returned, and the processing of the data set may continue. A message describing the I/O error may be obtained by calling EDMSG(P).

201 For a non-edit format data set, where EDNCOL(P) had been called, an invalid line number was encountered. LINENO is set to -1. The characters to be used as the line number, as well as the value of the last valid line number, may be obtained by calling EDSHOW(P).

202 A physical block that was not in edit format was encountered in an edit format data set. No record from the data set is returned. The next call to EDGET(P) will process records starting with the next physical block.

203 In an edit format data set, a line that is not in edit format was encountered. No line is returned. The next call to EDGET(P) will process records starting with the next line.

204 The line number exceeded the maximum number allowed. LINENO is set to -1.

205 The line number is less than or equal to the previous line number. To obtain the value of the previous line number, call EDSHOW(P). The line number obtained by the next call to EDGET(P) will be compared against the line number returned from this call to EDGET(P).

206 A non-edit format data set with a record format of U or V is being processed. EDNCOL(P) has been called. A record

is encountered whose length is too short to contain the line number specified by the call to EDNCOL(P). LINENO is set to -1. The characters representing the partial line number may be obtained by calling EDSHOW(P).

207 A wrong length record has been read from a non-edit format data set. The record was either shorter than the minimum allowed or longer than the maximum allowed length. A message showing the minimum and maximum allowed lengths may be obtained by calling EDMSG(P). No record from the data set is returned. The next call to EDGET(P) will process records starting with the next record.

300 An ABEND has been intercepted. The intercepted IBM ABEND code may be obtained by calling EDSHOW(P). A descriptive message may be obtained by calling EDMSG(P). No record is returned. Further data set processing (other than EDCLOS(P) and EDTERM(P)) are not allowed.

LINENO A binary integer line number is stored by EDGET(P) in this full word (4 byte) area.

When reading an edit format data set, LINENO is normally set equal to the line number obtained from the record read. However, if the EDNGEN(P) subroutine has been called to generate line numbers, then LINENO will be set equal to the line number that was generated for the record read.

When reading a non-edit format data set, LINENO is normally set equal to -1. There are two exceptions: (1) if the EDNCOL(P) subroutine has been called to specify that line numbers are in the data set, then LINENO will be set equal to the line number obtained from the record read; (2) if subroutine EDNGEN(P) has been called to generate line numbers, then the LINENO will be set equal to the line number which was generated for the record read.

LINE Subroutine EDGET(P) stores the characters of the line read into this character variable. If necessary, the line read will be padded on the right with blanks or truncated on the right to fit.

For subroutine EDGETP, this argument must be a PL/I

character string variable. If LINE is a fixed length character string variable, the line read will be padded on the right with blanks. For a varying length character string, the current length is set to the actual length of the line read.

LENMAX	The maximum number of characters that can be placed in location LINE must be stored in this integer variable before EDGET is called. For subroutine EDGETP, this argument should be omitted.
LENACT	Subroutine EDGET stores into this integer variable the actual number of characters stored into the variable LINE, after truncation, but before padding with blanks. For subroutine EDGETP, this argument should be omitted.

Note:
If the data set is not open, the subroutine will ABEND.

3.5 SUBROUTINE EDPUT(P)

Purpose: This subroutine is called to write a logical record into a data set. If the data set has edit format characteristics, the records are converted to edit format as they are written. The subroutine assumes that the data set has already been opened by the EDOPEN(P) subroutine. The buffers established by EDOPEN(P) are used to store the records to be written. Records are actually written to the data set when the buffers become full. The data set should be closed using the EDCLOS(P) subroutine when processing of the data set is completed in order to insure that the last buffer of information is written.

Arguments: CNTRL,RETURN,LINENO,LNOUT,LINE,LENGTH

Name	**Function**
CNTRL	The same control word that was specified when the data set was opened by the EDOPEN(P) subroutine call.
RETURN	Return code. Unless otherwise stated, a record was written. The possible values are:

0	Processing completed normally.
100	For non-edit format data sets, an attempt was made to write a record longer than the maximum record length. The line was written, but was truncated on the right to the maximum

length.

101	More positions were required to write the line number than were specified by EDNCOL(P). The line was written, but the line number was truncated on the left.
102	In a non-edit format data set the line number was less than or equal to the previous line number. The value of the last valid line number may be obtained by calling EDSHOW(P). The line number used in the next call to EDPUT(P) will be compared against the number used in this call to EDPUT(P).
200	An I/O error occurred. The processing of the data set may continue. A message describing the I/O error may be obtained by calling EDMSG(P).
201	The line number exceeded the maximum number allowed. The line associated with the line number was ignored and no line was written. The value of the last valid line number may be obtained by calling EDSHOW(P).
202	In an edit format data set the line number was less than or equal to the previous line number. The line was not written. The value of the last valid line number may be obtained by calling EDSHOW(P).
300	An ABEND has been intercepted. The intercepted IBM ABEND code may be obtained by calling EDSHOW(P). A descriptive message may be obtained by calling EDMSG(P). No line is written. Further data set references (other than EDCLOS(P) and EDTERM(P)) are not allowed

LINENO — A binary integer line number to be associated with the record to be written must be stored in this integer variable before EDPUT(P) is called.

If LINENO is -1 and a line number is needed, the line number will automatically be generated. Line numbers are generated by examining the last line number written and selecting the next

highest integer line number. If no line number has been written, the line number 1 is used. If EDNGEN(P) has been called, the increment specified by EDNGEN(P) is used to generate line numbers, including the starting line number.

If a non-edit format data set is being written, and the EDNCOL(P) subroutine is not called to indicate the location for line numbers, then no line numbers are written and the value of LINENO is ignored.

LNOUT	EDPUT(P) stores into this full word (4 byte) area a binary integer line number equal to the line number written. If a non-edit format data set is being written, and the EDNCOL(P) subroutine is not called to indicate the location for the line numbers, then no line numbers are written and the value of LNOUT is set to –1 .
LINE	The characters of the line to be written must be stored in LINE before EDPUT(P) is called. If necessary, lines written to data sets with RECFM=F or RECFM=FB will be padded on the right with blanks or truncated on the right. For subroutine EDPUTP, this argument must be a PL/I character string variable.
LENGTH	The actual number of characters to be written from location LINE must be stored into this integer variable before EDPUT is called. For subroutine EDPUTP, this argument should be omitted.

Note:
If the data set is not open, the subroutine will ABEND.

3.6 SUBROUTINE EDMSG(P)

Purpose: This routine returns the text of a message associated with the return code returned by the last subroutine call.

Arguments: CNTRL,RETURN,MSG,LENMAX,LENACT

Name	Function
CNTRL	The same control word which was specified when EDINIT(P) was called.

RETURN — Return code. The possible values are:

- 0 — Processing completed normally.
- 100 — The message has been truncated on the right to a length equal to the number of characters specified by the value of LENMAX (or the length of the PL/I character string variable when EDMSGP is called). The actual length of the message, after truncation, is returned in LENACT.

MSG — The characters representing the error message are stored into MSG by the EDMSG(P) subroutine. The message returned will not exceed 132 characters in length. If necessary, the message will be padded on the right with blanks or truncated on the right to fit into the area.

For subroutine EDMSGP, this argument must be a PL/I character string variable. If MSG is a fixed-length character string variable, the message will be padded on the right with blanks. For a varying length character string, the current length is set to the actual length of the message.

LENMAX — The maximum number of characters that can be placed in location MSG must be stored into this integer variable before EDMSG is called.

For subroutine EDMSGP, this argument should be omitted.

LENACT — Subroutine EDMSG stores into this integer variable the actual number of characters in the message, after truncation, but before padding with blanks.

For subroutine EDMSGP, this argument should be omitted.

3.7 SUBROUTINE EDCLOS(P)

Purpose: This subroutine must be called for each data set processed in order to properly close the data set. If the data set is being written, the last few records stored in the buffer portion of the work area are written to the data set.

The internal work area established by EDINIT(P) is returned to the state it was in immediately following the call to EDINIT(P). Thus, a call to subroutine EDCLOS(P) "erases" the following information concerning the data set being processed:

a) All information about the column positions for line numbers and how they are to be generated, that was specified by calls to EDNCOL(P) and/or EDNGEN(P).

b) All DCB characteristics.

Therefore, if another data set (or the same data set) is to be processed, this information may need to be specified again before EDOPEN(P) is called.

If no more data sets are to be processed, EDTERM(P) should be called.

Arguments: CNTRL,RETURN

Name	Function
CNTRL	The same control word which was specified when the data set was opened by the EDOPEN(P) subroutine call.
RETURN	Return code. The possible values are:
	0 Processing completed normally.
	100 No data set was open.
	200 An I/O error has occurred. The processing of the job will continue. A message describing the I/O error may be obtained by calling EDMSG(P). The data set was closed.
	300 An ABEND has been intercepted. The intercepted IBM ABEND code may be obtained by calling EDMSG(P) or EDSHOW(P). The data set was closed.

3.8 SUBROUTINE EDTERM(P)

Purpose: This subroutine is called to terminate processing with the ED subroutines. The work areas set up by the EDINIT(P) subroutine are freed, thus making the storage used by the work areas available to other parts of the program being executed. Subroutine EDCLOS(P) should have been called previously to close the data set.

Arguments: CNTRL,RETURN

Name	Function
CNTRL	The same control word that was specified when EDINIT(P) was called.
RETURN	Return code. The possible values are:
	0 Processing completed normally.
	100 Subroutine EDCLOS(P) had not been called to close the data set. EDTERM(P) first closes the data set and then

frees the work areas.

3.9 Examples

The following examples include programs for each language that can call these subroutines: COBOL, FORTRAN, PL/I and Assembler. Each program performs
the same function. An edit format data set is read and printed. There are at most 80 characters in each record of the data set. A sample DD statement for the input data set is:

```
//GO.IN    DD  DSNAME=AAAAIII.INPUT,DISP=SHR
```

For COBOL and PL/I, output is directed to the SYSPRINT DD statement; for Assembler, output is directed to the SYSOUT DD statement; and for FORTRAN, output is directed to the FT06F001 DD statement. These DD statements are defined as the printer in the Computer Center standard procedures.

3.9.1 COBOL

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. ED.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  CNTRL               PIC X(4).
 01  DDNAME              PIC X(8).
 01  LINE-VAR            PIC X(80).
 01  MSG-VAR             PIC X(132).
 01  RETURN-VALUE        PIC S9(8) COMPUTATIONAL.
 01  LINENO              PIC S9(8) COMPUTATIONAL.
 01  TYP                 PIC S9(8) COMPUTATIONAL.
 01  LENMAX              PIC S9(8) COMPUTATIONAL.
 01  LENACT              PIC S9(8) COMPUTATIONAL.
 PROCEDURE DIVISION.
*
* SET UP INPUT DATA SET
*
     MOVE 'IN      ' TO DDNAME.
     CALL 'EDINIT' USING CNTRL RETURN-VALUE.
     MOVE 1 TO TYP.
     CALL 'EDOPEN' USING CNTRL RETURN-VALUE DDNAME TYP.
     IF RETURN-VALUE NOT EQUAL 0 GO TO CLOSE-SECTION.
*
* READ AND PRINT DATA SET
*
     MOVE 80 TO LENMAX.
 READ-PRINT-SECTION.
```

```
    CALL 'EDGET' USING CNTRL RETURN-VALUE LINENO LINE-VAR
       LENMAX LENACT.
    IF RETURN-VALUE NOT EQUAL 0 GO TO CLOSE-SECTION.
    DISPLAY LINE-VAR.
    GO TO READ-PRINT-SECTION.
*
* CLOSE DATA SET
*
 CLOSE-SECTION.
     MOVE 132 TO LENMAX.
     CALL 'EDMSG' USING CNTRL RETURN-VALUE MSG-VAR
        LENMAX LENACT.
     DISPLAY MSG-VAR.
     CALL 'EDCLOS' USING CNTRL RETURN-VALUE.
     CALL 'EDTERM' USING CNTRL RETURN-VALUE.
  EXITING.
      STOP RUN.
```

3.9.2 FORTRAN

```
C
C DECLARATIONS
C
      INTEGER*4 CNTRL,RETURN
      INTEGER*4 LINENO,LINE(20),LENMAX,LENACT,MSG(33)
      REAL*8 DDNAME
      DATA DDNAME/'IN      '/
C
C SET UP DATA SET TO BE READ
C
      CALL EDINIT(CNTRL,RETURN)
      CALL EDOPEN(CNTRL,RETURN,DDNAME,1)
      IF (RETURN.NE.0) GO TO 600
C
C READ AND PRINT DATA SET
C
100   CALL EDGET(CNTRL,RETURN,LINENO,LINE,80,LENACT)
      IF (RETURN.NE.0) GO TO 600
      WRITE(6,1000) LINE
1000  FORMAT(1X,20A4)
      GO TO 100
C
C CLOSE DATA SET
C
600   CALL EDMSG(CNTRL,RETURN,MSG,132,LENACT)
      WRITE(6,2000) MSG
```

```
2000  FORMAT(1X,33A4)
      CALL EDCLOS(CNTRL,RETURN)
      CALL EDTERM(CNTRL,RETURN)
700   STOP
      END
```

3.9.3 PL/I

```
 ED: PROCEDURE OPTIONS(MAIN);

    %INCLUDE EDDCLS;

    DCL (CNTRL,RETURN,LINENO) FIXED BINARY(31);
    DCL DDNAME CHARACTER(8) INIT('IN');
    DCL LINE CHARACTER(80) VARYING;
    DCL MSG CHARACTER(132) VARYING;

    /*  SET UP INPUT DATA SET  */

    CALL EDINITP(CNTRL,RETURN);
    CALL EDOPENP(CNTRL,RETURN,DDNAME,1);
    IF RETURN ^= 0 THEN GO TO CLOSE;

    /*  READ AND PRINT DATA SET  */

    DO WHILE ('1'B);
       CALL EDGETP(CNTRL,RETURN,LINENO,LINE);
       IF RETURN ^= 0 THEN GO TO CLOSE;
       PUT EDIT(LINE) (A) SKIP;
       END;

    /*  CLOSE DATA SETS  */

    CLOSE:
       CALL EDMSGP(CNTRL,RETURN,MSG);
       PUT EDIT(MSG) (A) SKIP;
       CALL EDCLOSP(CNTRL,RETURN);
       CALL EDTERMP(CNTRL,RETURN);
    EXIT:
       END ED;
```

3.9.4 Assembler

```
ED       CSECT
         SAVE  (14,12),,*
```

```
         BALR  12,0
         USING *,12
         ST    13,SAVEAREA+4
         LA    11,SAVEAREA
         ST    11,8(0,13)
         LR    13,11
         OPEN  (PUTDCB,(OUTPUT))
*
* SET UP INPUT DATA SET
*
         CALL  EDINIT,(CNTRL,RETURN)
         CALL  EDOPEN,(CNTRL,RETURN,DDNAME,ONE)
         CLC   RETURN,=F'0'
         BNE   CLOSE
*
* READ AND WRITE DATA SET
*
LOOP     CALL
EDGET,(CNTRL,RETURN,LINENO,LINE,LENMAX,LENACT)
         CLC   RETURN,=F'0'
         BNE   CLOSE
         PUT   PUTDCB,LINE
         B     LOOP
*
* CLOSE DATA SET
*
CLOSE    CALL  EDMSG,(CNTRL,RETURN,LINE,LENMAX2,LENACT)
         PUT   PUTDCB,LINE
         CALL  EDCLOS,(CNTRL,RETURN)
         CALL  EDTERM,(CNTRL,RETURN)
EXIT     CLOSE (PUTDCB)
         L     13,SAVEAREA+4
         RETURN (14,12),RC=0
*
* SET UP INITIAL VALUES AND WORK AREAS
*
CNTRL    DS    F
DDNAME   DC    CL8'IN'
LINE     DC    CL132' '
LENMAX   DC    F'80'
LENMAX2  DC    F'132'
LENACT   DS    F
RETURN   DS    F
LINENO   DS    F
ONE      DC    F'1'
PUTDCB   DCB
DDNAME=SYSOUT,DSORG=PS,MACRF=(PM),RECFM=FB,LRECL=132,   X
```

```
                BLKSIZE=132
SAVEAREA DC     18A(0)
         END
```

3.10 Special Considerations for Data Set Handling

3.10.1 Processing Multiple data sets:

In general, the processing of multiple data sets may be divided into two categories:

- Only one data set is open at any given time:

  Only one call to EDINIT(P) is necessary. After the EDINIT(P) call, the first data set is processed by calling EDOPEN(P) and whatever other routines are necessary. When processing of the first data set has been completed, it is closed by a call to EDCLOS(P), but EDTERM(P) should not be invoked. The second data set is processed by again calling EDOPEN(P) and whatever other routines are necessary. When processing of this data set is completed, it is closed by another call to EDCLOS(P). This procedure may be repeated for as many data sets as necessary. When all of the data sets have been processed, subroutine EDTERM(P) is called.

- Several data sets are to be opened simultaneously:

  These subroutines are reentrant and refreshable since they do not modify themselves. This means that the subroutines may be used to process more than one data set by establishing more than one work area with multiple calls to EDINIT(P) using different control words.

  There must be one call to EDINIT(P) for each data set which will be open at the same time. Each of these calls to EDINIT(P) must specify a unique variable name for the control word argument (CNTRL).

3.10.2 Adding onto an Existing Edit Format Data Set:

Records may be added to an edit format data set by using DISP=MOD or by setting TYPE equal to 102 in the call to EDOPEN(P), but it is the responsibility of the user to insure that the line numbers used for the data set will all be in ascending order.

3.11 SUBROUTINE EDNCOL(P)

Purpose: This subroutine specifies the column positions where line numbers are stored in a non-edit format data set. No line numbers are used when processing a non-edit format data set unless EDNCOL(P) or EDNGEN(P) are called.

If the data set is being read, the positions in each record that contain the line number are always removed as each line is processed. If the data set is being written, the text in the line normally is moved to the right, if necessary, to provide positions for the line number to be stored. The EDSET(P) subroutine may be called to have the line numbers overlay or be merged with text in the specified columns. The text of a line will be truncated on the right, if necessary, to make the length of the line fit the record length. If more positions are required to write a line number than are specified, the line number is truncated on the left.

Arguments: CNTRL,RETURN,TYPE,COL1,COL2

Name	**Function**
CNTRL	The same control word that was specified when EDINIT(P) was called.
RETURN	Return code. The possible values are:
	0 Processing completed normally.
	200 Invalid column positions were specified. The call to this subroutine will be ignored.
	201 The column positions specified are inconsistent with the attributes of the data set being processed. This subroutine call is ignored.
	202 The data set being processed is edit format. This subroutine call is ignored.
TYPE	An integer indicating information concerning the location and format of the line numbers must be stored in this integer variable before EDNCOL(P) is called. The possible values are:
	1 Turns off processing of line numbers. Further calls to EDGET(P) ignore line numbers that exist in the data set and further calls to EDPUT(P) no longer write line numbers into the data set.
	2 Indicates that WYLBUR line numbers are stored in the default positions. The location of the line numbers depends on the record format of the data set: F format -- last 8 positions of the data U format -- first 8 positions of the data V format -- first 8 positions of the data The numbers located in these positions are used as the

line number of each record in the data set.

3	Indicates that the columns where WYLBUR line numbers are stored are being specified by the user in the COL1 and COL2 parameters.
4	Specifies that TSO (IBM) line numbers are stored in the default positions. The location of the line numbers depends on the record format of the data set: F format -- last 8 positions of the data U format -- first 8 positions of the data V format -- first 8 positions of the data The numbers located in these positions are used as the line number of each record in the data set.
5	Indicates that TSO (IBM) line numbers are stored in the column positions specified by the user in the COL1 and COL2 parameters.

COL1	If the value of TYPE is 3 or 5, the integer variable COL1 must contain the starting absolute column position for the location of the line number when EDNCOL(P) is called. The first column is numbered 1. If TYPE is 1, 2, or 4, the contents of COL1 is not used by EDNCOL(P).
COL2	If the value of TYPE is 3 or 5, the integer variable COL2 must contain the ending absolute column position for the location of the line number when EDNCOL(P) is called. If TYPE is 1, 2, or 4, the contents of COL2 is not used by EDNCOL(P).

Note:
If the value of TYPE is invalid, the subroutine will ABEND.

3.12 SUBROUTINE EDNGEN(P)

Purpose: This subroutine permits the user to specify how line numbers are to be generated for each line in the data set.

Arguments: CNTRL,RETURN,TYPE,LINENO,INCR

Name	**Function**
CNTRL	The same control word which was specified when EDINIT(P) was called.

RETURN — Return code. The possible values are:

0	Processing completed normally.
200	An invalid starting line number was specified. The call to this subroutine will be ignored.
201	An invalid increment was specified. The call to this subroutine will be ignored.
202	An invalid multiplication factor was specified. The call to this subroutine will be ignored.
203	Starting line number is less than or equal to the previous line number. The call to this routine will be ignored. To obtain the value of the previous line number, call EDSHOW(P).

TYPE — An integer indicating the type of processing to be performed must be stored in this integer variable before EDNGEN(P) is called. The possible values are:

1	Removes the use of the line number generation scheme as previously established by a call to EDNGEN(P). Further calls to EDPUT(P) must include the line number, unless EDNCOL(P) has been called to stop the storing of line numbers in a non-edit format data set. Further calls to EDGET(P) will return the line number or return -1 if no line number is available.
2	The remaining parameters specify the starting line number and increment to be used to generate line numbers. If a non-edit format data set is being read and the EDNCOL(P) subroutine has been called to indicate that line numbers in each record of the data set are to be used, the positions containing the line numbers of each record are removed and the numbering process is performed as specified by this call to EDNGEN(P).
3	The line numbers are multiplied by the factor supplied in the variable LINENO. For data sets being read, the line numbers are obtained from the data set; for data sets being written, the user is specifying the line numbers in the calls to EDPUT(P).

LINENO	If TYPE is 2, LINENO must contain a binary integer line number representing the starting line number when EDNGEN(P) is called. If TYPE is 3, LINENO must contain a positive binary integer line number representing the multiplication factor used to generate line numbers when EDNGEN(P) is called. A line number is generated by multiplying the current value of the line number by the factor. If TYPE is 1, the contents of LINENO is not used by EDNGEN(P).
INCR	If TYPE is 2, INCR must contain a positive binary integer line number representing the increment used to generate line numbers when EDNGEN(P) is called. A line number is generated by adding the increment to the current value of the line number. If TYPE is 1 or 3, the contents of INCR is not used by EDNGEN(P).

Note:
If the value of TYPE is invalid, the subroutine will ABEND.

3.13 SUBROUTINE EDSET(P)

Purpose: This subroutine may be used to set information about the data set (e.g., LRECL). Information set by this subroutine takes precedence over information on the DD statement or in the data set label.

Arguments: CNTRL,RETURN,ITEM,IPARM,CPARM,LENGTH

Name	Function
CNTRL	The same control word which was specified when EDINIT(P) was called.
RETURN	Return code. The possible values are: 0 Processing completed normally. 200 An invalid value for IPARM was specified. The call to this subroutine will be ignored. 201 Invalid character information was specified in CPARM. The call to this subroutine will be ignored.

ITEM

An integer indicating the item to be set must be stored in this integer variable before EDSET(P) is called. The possible values are:

1 The logical record length (LRECL), between 0 and 32760, of the data set is specified in the variable IPARM. This must be set before EDOPEN(P) is called.

2 The block size (BLKSIZE), between 0 and 32760, of the data set is specified in the variable IPARM. This must be set before EDOPEN(P) is called.

3 A flag may be set for each line in an edit format data set indicating whether the line has been changed. This is used together with the variable IPARM to indicate whether or not the flag is to be used. The acceptable values of IPARM are:

. 1 means that each line written is to be flagged as having been changed (this is the default).

. 2 means that any lines written are not to be flagged as having been changed.

This mechanism can be used to selectively flag lines that are being changed. The EDSHOW(P) subroutine may be used to display whether or not changed lines are being flagged, and if a line is read, whether or not it was flagged as having been changed.

4 Specifies the technique used for writing line numbers in each line of a non-edit format data set. The technique chosen is determined by the value of IPARM. The acceptable values of IPARM are:

. 1 means to move the text in the columns to the right and insert the line number (this is the default).

. 2 means to overlay whatever is in the columns with the line number.

. 3 means to merge the line numbers into the columns -- if the columns contain blanks, move the line number into the columns; if the columns do not contain all blanks, do not store the line number in the

line.

5 Indication of whether or not the subroutine is to intercept data management ABEND situations in EDOPEN(P), EDGET(P), EDPUT(P) and EDCLOS(P). The value of IPARM determines what is done. The acceptable values of IPARM are:

. 1 means ABEND situations will be intercepted. These situations will cause an IBM message to be written to the JES2 system log and no dump is produced (this is the default).

. 2 means ABEND situations will not be intercepted. These situations will cause termination of the job and a dump to be produced if a dump DD statement has been included in the JCL.

6 Indicates that the format of the data set is being specified. The format is determined by the value of IPARM. The acceptable values of IPARM are:

. 1 means EDIT format. For data sets being read, EDGET(P) will attempt to read the data set as edit format regardless of its characteristics. For data sets being written, any DCB parameters previously specified are not used and the data set is written in EDIT format.

. 2 means non-EDIT format. If no record format is specified, then VBS is used. If no logical record length is specified, then 504 is used. For blocked data sets and no block size is specified, the block size chosen will be equal to the greatest integer which is less than or equal to 11476 and which is compatible with the record format and logical record length.

7 Specifies whether or not character strings returned from a call to EDGET are to be padded on the right with blanks to the maximum length of the string (i.e., the value of LENMAX). The value of IPARM determines what is done. The acceptable values of IPARM are:

. 1 means padding is performed (this is the default).

. 2 means padding is not performed.

101 The record format (RECFM) of the data set (e.g., F, FB, VBS, etc.) is specified in the variable CPARM. This must be set before EDOPEN(P) is called.

IPARM — If the value of ITEM is less than 100, the integer variable IPARM must contain the information being set before EDSET(P) is called. If the value of ITEM is greater than 100, then the contents of IPARM are not used by EDSET(P).

CPARM — If the value of ITEM is greater than 100, CPARM must contain the information being set before EDSET(P) is called. If the value of ITEM is less than 101, then the contents of CPARM are not used by EDSET(P).

For subroutine EDSETP, this argument must be a PL/I character string variable.

LENGTH — The maximum number of characters that can be placed in location CPARM must be stored into this integer variable before EDSET is called.

For subroutine EDSETP, this argument should be omitted.

Notes:

If the value of ITEM is invalid, or an attempt is made to set a value for a data set that is opened and the item must be set before the data set is opened, the subroutine will ABEND.

3.14 SUBROUTINE EDSHOW(P)

Purpose: This subroutine may be used to show information (e.g., LRECL of data set, RECFM of data set).

Arguments: CNTRL,RETURN,ITEM,IANS,CANS,LENMAX,LENACT

Name	Function
CNTRL	The same control word which was specified when EDINIT(P) was called.
RETURN	Return code. The possible values are:

0 Processing completed normally.

100 No information is available (e.g., before EDOPEN(P) has been called). If ITEM was less than 100, -1 was stored in IANS; if ITEM was greater than 100, blanks were placed in location CANS.

101 Before being stored in location CANS, character information was truncated on the right to a length equal to the number of characters specified by the value of LENMAX (or the length of the PL/I character string variable when EDSHOWP is called). The actual length of the record, after truncation, is returned in LENACT.

102 Data set has not been opened. The information returned is the default value or the value set by the EDSET(P) routine.

ITEM An integer indicating the item to be shown must be stored in this integer variable before EDSHOW(P) is called. The possible values are:

1 The logical record length (LRECL) of the data set.

2 The block size (BLKSIZE) of the data set.

3 The maximum number of characters allowed in each logical record of a data set. This differs from the LRECL in the case of edit format data sets and data sets with RECFM of V.

4 The actual length of text returned by EDGET(P), EDNCNV(P), or EDSHOW(P) (before truncation, if applicable).

5 Value of the last valid line number referenced.

6 Indicates whether or not a flag is being set for lines changed. The value returned in IANS indicates the action being taken:

. 1 means when lines are written they are flagged as having been changed.

. 2 means when lines are written they are not flagged as having been changed.

7 Indicates whether or not the last line read has the flag set

or not, indicating that a change has been made in the line. The value returned in IANS indicates the action being taken:

. 1 means the record read has been changed.

. 2 means the record read has not been changed.

8 Indicates the method used for writing line numbers into a line. The value returned in IANS indicates the action being taken:

. 1 means that text is moved to the right when inserting the line numbers.

. 2 means that whatever is in the columns is overlayed with the line number.

. 3 means that the line numbers are being merged with the columns -- if the columns contain blanks, the line number is moved into the columns; if the columns do not contain all blanks, the line number is not stored in the line.

9 Indicates whether or not ABEND situations are intercepted. The value returned in IANS indicates the action being taken:

. 1 means ABEND situations are intercepted.

. 2 means ABEND situations are not intercepted.

10 The format of the data set. The value returned in IANS indicates whether or not the data set is in edit or non-edit format:

. 1 means the data set is edit format.

. 2 means the data set is non-edit format.

11 The record format (RECFM) of the data set in the form it appears in the DCBRECFM field of a data control block. The three high order bytes of the integer are zero. See the *IBM MVS/DFP Macro Instructions for Non-VSAM Data Sets* manual for more information on the DCBRECFM field. The record format of the data set is

available in character representation by using 101 as the value for ITEM.

12 Indicates whether or not a string returned from a call to EDGET is padded on the right with blanks. The value returned in IANS indicates the action being taken:

. 1 means padding is performed.

. 2 means padding is not performed.

101 The record format (RECFM) of the data set. The characters returned are those that would be coded in the RECFM subparameter of the DCB parameter of a DD statement (e.g., FB for fixed blocked format).

102 The value of the IBM ABEND code for an intercepted ABEND. The value returned in CANS is of the form xxx-rc, where "xxx" is three characters representing the actual hexadecimal ABEND code and "-rc" is the optional return code provided with some ABEND codes. Examples are 80A and 213-04.

103 When reading a non-edit format data set, an invalid line number was encountered. The characters read that form the invalid line number are returned in CANS.

IANS An integer representing the information requested is stored in this integer variable by EDSHOW(P). Used when ITEM is less than 100. If ITEM is greater than 100, then the contents of IANS are not used by EDSHOW(P).

CANS The characters representing the information requested is stored in this area by EDSHOW(P). If necessary, the information is padded on the right with blanks or truncated on the right. Used when ITEM is greater than 100. If ITEM is less than 100, then the contents of CANS are not used by EDSHOW(P).

For subroutine EDSHOWP, this argument must be a PL/I character string variable. If CANS is a fixed-length character string variable, the information will be padded on the right with blanks. For a varying length character string, the current length is set to the actual length of the information.

LENMAX	The maximum number of characters that can be placed in location CANS must be stored into this integer variable before EDSHOW is called. For subroutine EDSHOWP, this argument should be omitted.
LENACT	Subroutine EDSHOW stores into this integer the actual number of characters stored in location CANS, after truncation, but before padding with blanks. For subroutine EDSHOWP, this argument should be omitted.

Note:
If the value of ITEM is invalid, the subroutine will ABEND.

3.15 SUBROUTINE EDGETL(P)

Purpose: This subroutine is a version of EDGET(P) that returns the address of the location where the characters of the line read are stored.

Since there is no support in COBOL and Fortran for handling pointers, this subroutine is not useable in these two languages.

Arguments: CNTRL,RETURN,LINENO,PNTR,LENACT

Name	**Function**
CNTRL	The same control word that was specified when the data set was opened by the EDOPEN(P) subroutine call.
RETURN	Return code. Values are identical to those of EDGET(P).
LINENO	A binary integer line number is stored by EDGETL(P) in this full word (4 byte) area. When reading an edit format data set, LINENO is normally set equal to the line number obtained from the record read. However, if the EDNGEN(P) subroutine has been called to generate line numbers, then LINENO will be set equal to the line number which was generated for the record read. When reading a non-edit format data set, LINENO is normally set equal to -1. There are two exceptions: (1) if the EDNCOL(P) subroutine has been called to specify that line numbers are in the data set, then LINENO will be set equal to the line number obtained from the record read: (2) if subroutine

EDNGEN(P) has been called to generate line numbers, then the LINENO will be set equal to the line number which was generated for the record read.

PNTR — A binary integer variable where EDGETL(P) stores the address of the location where the characters of the line read are stored. The value returned in PNTR is only valid until the next call to EDGETL(P), EDGET(P), EDCLOS(P), or EDTERM(P).

For subroutine EDGETLP, this argument must be declared as a POINTER variable. Upon successful return from EDGETLP, the pointer will contain the address of a fixed length character string whose length is returned in LENACT. Since a fixed length string is returned, the PL/I programmer must either know the length of the records to be returned, or declare a BASED character string with a length longer than any possible record and use the SUBSTR function to refer to the actual data.

LENACT — Subroutine EDGETL(P) stores into this integer variable the actual number of characters in the line read.

Note:
If the data set is not open, the subroutine will ABEND.

3.16 SUBROUTINE EDNCNV(P)

Purpose: This subroutine may be used to convert the representation of a line number (1) between binary integer and WYLBUR character format, or (2) between binary integer and TSO (IBM) character format.

Arguments: CNTRL,RETURN,OPTION,LINENO,CHARS,LENMAX,LENACT

Name	Function
CNTRL	The same control word that was specified when EDINIT(P) was called.
RETURN	Return code. The possible return values are: 0 Processing completed normally. 100 The converted line number was truncated on the left in order to fit in the area established by CHARS, to the length equal to the number of characters specified by the value of LENMAX (or the length of the PL/I character string if EDNCNVP is called). The actual number of characters in the line number,

after truncation, is returned in LENACT.

200 An attempt was made to convert an invalid line number.

OPTION An integer representing the type of conversion to be done must be stored in this integer variable before EDNCNV(P) is called. The values are:

1 Conversion from binary integer to WYLBUR format. A decimal point is always included in the converted line number. The result of the conversion is 9 characters, with the decimal point in the sixth position.

2 Conversion from binary integer to WYLBUR format. If the converted line number is an integer, the decimal point is not included in the converted line number. The result of the conversion is the least number of characters necessary to represent the line number.

3 Conversion from binary integer to TSO (IBM) format. The result of the conversion is eight digits, with leading zeros.

101 Conversion from WYLBUR format to binary integer format.

102 Conversion from TSO (IBM) format to binary integer format.

LINENO Contains the binary integer line number, either to be converted or after conversion, depending on the value of OPTION.

CHARS Contains the WYLBUR or TSO (IBM) line number in character string format, either to be converted or after conversion, depending on the value of OPTION.

For subroutine EDNCNVP, if the conversion is from binary integer to character string (i.e., WYLBUR or TSO (IBM)) format, this argument must be a character string variable. If CHARS is a fixed-length character string variable, the line number will be padded on the right with blanks. For a varying length character string, the current length is set to the actual length of the line number.

LENMAX	The maximum number of characters which can be placed in location CHARS must be stored in this integer variable before EDNCNV is called. For subroutine EDNCNVP, this argument should be omitted.
LENACT	The actual number of characters in a line number, after the line number was truncated, but before the line number was padded with blanks, is returned in this integer variable by subroutine EDNCNV. The number of characters is only calculated when conversion is from binary integer to character format (i.e., WYLBUR or TSO (IBM)). This variable will be ignored if conversion is from character to binary integer format. For subroutine EDNCNVP, this argument should be omitted. For subroutine EDNCNVP, this argument should be omitted.

Note:

If the value of OPTION is invalid, the subroutine will ABEND.

3.17 Examples

The following examples include programs for each language that can call these subroutines: COBOL, FORTRAN, PL/I and Assembler. Each program performs the same function. An edit format data set is copied to a non-edit data set. The ddname of the DD statement for the edit format input data set is stored in DDNAMI. There are at most 80 characters in each record of the data set. In the non-edit format data set written, the line numbers are stored in columns 73 through 80. The ddname of the DD statement for the output data set is stored in DDNAMO. In an effort to keep the examples as short and simple as possible, only a bare minimum of error processing has been included. Sample DD statements for these data sets are:

```
//GO.IN    DD  DSNAME=AAAAIII.INPUT,DISP=SHR
//GO.OUT   DD  DSNAME=AAAAIII.OUTPUT,UNIT=TMP,
//             DISP=(NEW,CATLG),DCB=(RECFM=FB,LRECL=80,
//             BLKSIZE=11440),
//             SPACE=(TRK,(10,2),RLSE)
```

3.17.1 COBOL

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ED.

DATA DIVISION.
WORKING-STORAGE SECTION.
01  CNTRLI              PIC X(4).
```

```
 01  CNTRLO              PIC X(4).
 01  DDNAMI              PIC X(8).
 01  DDNAMO              PIC X(8).
 01  LINE-VAR            PIC X(80).
 01  MSG-VAR             PIC X(132).
 01  RETURN-VALUE        PIC S9(8) COMPUTATIONAL.
 01  LINENO              PIC S9(8) COMPUTATIONAL.
 01  TYP                 PIC S9(8) COMPUTATIONAL.
 01  LNOUT               PIC S9(8) COMPUTATIONAL.
 01  LENGTH-VAR          PIC S9(8) COMPUTATIONAL.
 01  LENMAX              PIC S9(8) COMPUTATIONAL.
 01  LENACT              PIC S9(8) COMPUTATIONAL.
 01  COL1                PIC S9(8) COMPUTATIONAL.
 01  COL2                PIC S9(8) COMPUTATIONAL.
 PROCEDURE DIVISION.
 SET-UP-SECTION.
     MOVE 'IN      ' TO DDNAMI.
     MOVE 'OUT     ' TO DDNAMO.
*
*  SET UP DATA SETS
*
     CALL 'EDINIT' USING CNTRLI RETURN-VALUE.
     CALL 'EDINIT' USING CNTRLO RETURN-VALUE.
     MOVE 1 TO TYP.
     CALL 'EDOPEN' USING CNTRLI RETURN-VALUE DDNAMI TYP.
     IF RETURN-VALUE NOT EQUAL 0 GO TO ERROR-I.
     MOVE 2 TO TYP.
     CALL 'EDNCOL' USING CNTRLO RETURN-VALUE TYP COL1 COL2.
     IF RETURN-VALUE NOT EQUAL 0 GO TO ERROR-O.
     MOVE 101 TO TYP.
     CALL 'EDOPEN' USING CNTRLO RETURN-VALUE DDNAMO TYP.
     IF RETURN-VALUE NOT EQUAL 0 GO TO ERROR-O.
*
*  READ AND WRITE DATA SET
*
 READ-WRITE-SECTION.
     MOVE 80 TO LENMAX.
     CALL 'EDGET' USING CNTRLI RETURN-VALUE LINENO LINE-VAR
        LENMAX LENACT.
     IF RETURN-VALUE NOT EQUAL 0 GO TO ERROR-I.
     MOVE 72 TO LENGTH-VAR.
     CALL 'EDPUT' USING CNTRLO RETURN-VALUE LINENO LNOUT
        LINE-VAR LENGTH-VAR.
     IF RETURN-VALUE NOT EQUAL 0 GO TO ERROR-O.
     GO TO READ-WRITE-SECTION.
 *
 * OBTAIN TEXT OF ERROR MESSAGE
```

```
      *
       ERROR-I.
           MOVE 132 TO LENMAX.
           CALL 'EDMSG' USING CNTRLI RETURN-VALUE MSG-VAR
              LENMAX LENACT.
           GO TO PRINT-IT.
       ERROR-O.
           MOVE 132 TO LENMAX.
           CALL 'EDMSG' USING CNTRLO RETURN-VALUE MSG-VAR
              LENMAX LENACT.
       PRINT-IT.
           DISPLAY MSG-VAR.
      *
      *  CLOSE DATA SETS
      *
       CLOSE-SECTION.
           CALL 'EDCLOS' USING CNTRLI RETURN-VALUE.
           CALL 'EDCLOS' USING CNTRLO RETURN-VALUE.
           CALL 'EDTERM' USING CNTRLI RETURN-VALUE.
           CALL 'EDTERM' USING CNTRLO RETURN-VALUE.
       EXITING.
           STOP RUN.
```

3.17.2 FORTRAN

```
C
C  DECLARATIONS
C
      INTEGER*4 CNTRLI,CNTRLO,RETURN
      INTEGER*4 LINENO,LNOUT,LINE(20),LENMAX,LENACT,MSG(33)
      REAL*8 DDNAMI,DDNAMO
      DATA DDNAMI/'IN      '/,DDNAMO/'OUT     '/
C
C  SET UP DATA SETS
C
      CALL EDINIT(CNTRLI,RETURN)
      CALL EDINIT(CNTRLO,RETURN)
      CALL EDOPEN(CNTRLI,RETURN,DDNAMI,1)
      IF (RETURN.NE.0) GO TO 200
      CALL EDNCOL(CNTRLO,RETURN,2,0,0)
      IF (RETURN.NE.0) GO TO 300
      CALL EDOPEN(CNTRLO,RETURN,DDNAMO,101)
      IF (RETURN.NE.0) GO TO 300
C
C  READ AND WRITE DATA SET
C
  100 CALL EDGET(CNTRLI,RETURN,LINENO,LINE,80,LENACT)
```

```
      IF (RETURN.NE.0) GO TO 200
      CALL EDPUT(CNTRLO,RETURN,LINENO,LNOUT,LINE,72)
      IF (RETURN.NE.0) GO TO 300
      GO TO 100
C
C  OBTAIN TEXT OF ERROR MESSAGE
C
  200 CALL EDMSG(CNTRLI,RETURN,MSG,132,LENACT)
      GO TO 400
  300 CALL EDMSG(CNTRLO,RETURN,MSG,132,LENACT)
  400 WRITE(6,500) MSG
  500 FORMAT(1X,33A4)
C
C  CLOSE DATA SETS
C
  600 CALL EDCLOS(CNTRLI,RETURN)
      CALL EDCLOS(CNTRLO,RETURN)
      CALL EDTERM(CNTRLI,RETURN)
      CALL EDTERM(CNTRLO,RETURN)
  700 STOP
      END
```

3.17.3 PL/I

```
 ED: PROCEDURE OPTIONS(MAIN);

    %INCLUDE EDDCLS;
    DCL (CNTRLI,CNTRLO) FIXED BINARY(31);
    DCL DDNAMI CHARACTER(8) INIT('IN');
    DCL DDNAMO CHARACTER(8) INIT('OUT');
    DCL LINE CHARACTER(80) VARYING;
    DCL MSG CHARACTER(132) VARYING;
    DCL (RETURN,LINENO,LNOUT) FIXED BINARY(31);

    /*  SET UP DATA SETS  */

    CALL EDINITP(CNTRLI,RETURN);
    CALL EDINITP(CNTRLO,RETURN);
    CALL EDOPENP(CNTRLI,RETURN,DDNAMI,1);
    IF RETURN ^= 0 THEN GO TO ERROR_I;
    CALL EDNCOLP(CNTRLO,RETURN,2,0,0);
    IF RETURN ^= 0 THEN GO TO ERROR_O;
    CALL EDOPENP(CNTRLO,RETURN,DDNAMO,101);
    IF RETURN ^= 0 THEN GO TO ERROR_O;

    /*  READ AND WRITE DATA SET  */
```

```
    DO WHILE ('1'B);
       CALL EDGETP(CNTRLI,RETURN,LINENO,LINE);
       IF RETURN ^= 0 THEN GO TO ERROR_I;
       CALL EDPUTP(CNTRLO,RETURN,LINENO,LNOUT,LINE);
       IF RETURN ^= 0 THEN GO TO ERROR_O;
       END;

    /*  OBTAIN TEXT OF ERROR MESSAGE  */

    ERROR_I:
     CALL EDMSGP(CNTRLI,RETURN,MSG);
     PUT EDIT(MSG) (A) SKIP;
     GO TO CLOSE;
    ERROR_O:
     CALL EDMSGP(CNTRLO,RETURN,MSG);
     PUT EDIT(MSG) (A) SKIP;

    /*  CLOSE DATA SETS  */

   CLOSE:
    CALL EDCLOSP(CNTRLI,RETURN);
    CALL EDCLOSP(CNTRLO,RETURN);
    CALL EDTERMP(CNTRLI,RETURN);
    CALL EDTERMP(CNTRLO,RETURN);
   EXIT:
    END ED;
```

3.17.4 Assembler

```
ED       CSECT
         SAVE  (14,12),,*
         BALR  12,0
         USING *,12
         ST    13,SAVEAREA+4
         LA    11,SAVEAREA
         ST    11,8(0,13)
         LR    13,11
         OPEN  (PUTDCB,(OUTPUT))
*
*  SET UP DATA SETS
*
         CALL  EDINIT,(CNTRLI,RETURN)
         CALL  EDINIT,(CNTRLO,RETURN)
         CALL  EDOPEN,(CNTRLI,RETURN,DDNAMI,ONE)
         CLC   RETURN,=F'0'
         BNE   ERRORI
         CALL  EDNCOL,(CNTRLO,RETURN,TWO,ZERO,ZERO)
```

```
         CLC   RETURN,=F'0'
         BNE   ERRORO
         CALL  EDOPEN,(CNTRLO,RETURN,DDNAMO,ONEOHONE)
         CLC   RETURN,=F'0'
         BNE   ERRORO
*
*  READ AND WRITE DATA SET
*
LOOP     CALL  EDGET,(CNTRLI,RETURN,LINENO,LINE,LENMAX,LENACT)
         CLC   RETURN,=F'0'
         BNE   ERRORI
         CALL  EDPUT,(CNTRLO,RETURN,LINENO,LNOUT,LINE,LENGTH)
         CLC   RETURN,=F'0'
         BNE   ERRORO
         B     LOOP
*
*  OBTAIN TEXT OF ERROR MESSAGE
*
ERRORI   CALL  EDMSG,(CNTRLI,RETURN,LINE,LENMAX2,LENACT)
         B     PRINT
ERRORO   CALL  EDMSG,(CNTRLO,RETURN,LINE,LENMAX2,LENACT)
PRINT    PUT   PUTDCB,LINE
*
*  CLOSE DATA SETS
*
CLOSE    CALL  EDCLOS,(CNTRLI,RETURN)
         CALL  EDCLOS,(CNTRLO,RETURN)
         CALL  EDTERM,(CNTRLI,RETURN)
         CALL  EDTERM,(CNTRLO,RETURN)
EXIT     CLOSE (PUTDCB)
         L     13,SAVEAREA+4
         RETURN (14,12),RC=0
*
*  SET UP INITIAL VALUES AND WORK AREAS
*
CNTRLI   DS    F
CNTRLO   DS    F
DDNAMI   DC    CL8'IN'
DDNAMO   DC    CL8'OUT'
LINE     DC    CL132' '
LENGTH   DC    F'72'
LENMAX   DC    F'80'
LENMAX2  DC    F'132'
LENACT   DS    F
RETURN   DS    F
LINENO   DS    F
LNOUT    DS    F
```

```
ZERO     DC    F'0'
ONE      DC    F'1'
TWO      DC    F'2'
ONEOHONE DC    F'101'
PUTDCB   DCB
DDNAME=SYSOUT,DSORG=PS,MACRF=(PM),RECFM=FB,LRECL=80,     X
               BLKSIZE=80
SAVEAREA DC    18A(0)
         END
```

3.18 Reference Summary

Page	Subroutine	Arguments	Function
23	EDINIT	CNTRL,RETURN	Initialization
23	EDOPEN	CNTRL,RETURN,DDNAME, TYPE	Open data set
25	EDGET	CNTRL,RETURN,LINENO, LINE,LENMAX,LENACT	Read next record
28	EDPUT	CNTRL,RETURN,LINENO, LNOUT,LINE,LENGTH	Write next record
30	EDMSG	CNTRL,RETURN,MSG, LENMAX,LENACT	Get message text
31	EDCLOS	CNTRL,RETURN	Close data set
32	EDTERM	CNTRL,RETURN	Termination
37	EDNCOL	CNTRL,RETURN,TYPE,COL1,COL2	Column positions for line numbers
39	EDNGEN	CNTRL,RETURN,TYPE,LINENO, INCR	Line number generation
41	EDSET	CNTRL,RETURN,ITEM, IPARM,CPARM,LENGTH	Set information
44	EDSHOW	CNTRL,RETURN,ITEM,IANS, CANS,LENMAX,LENACT	Show information
48	EDGETL	CNTRL,RETURN,LINENO, PNTR,LENACT	Get address of next record

Page	Subroutine	Arguments	Function
49	EDNCNV	CNTRL,RETURN,OPTION,LINENO, CHARS,LENMAX,LENACT	Line number conversion
23	EDINITP	CNTRL,RETURN	Initialization
23	EDOPENP	CNTRL,RETURN,DDNAME, TYPE	Open data set
25	EDGETP	CNTRL,RETURN,LINENO, LINE	Read next record
28	EDPUTP	CNTRL,RETURN,LINENO, LNOUT, LINE	Write next record
30	EDMSGP	CNTRL,RETURN,MSG	Get message text
31	EDCLOSP	CNTRL,RETURN	Close data set
32	EDTERMP	CNTRL,RETURN	Termination
37	EDNCOLP	CNTRL,RETURN,TYPE,COL1,COL2	Column positions for line numbers
39	EDNGENP	CNTRL,RETURN,TYPE,LINENO, INCR	Line number generation
41	EDSETP	CNTRL,RETURN,ITEM,IPARM, CPARM	Set information
44	EDSHOWP	CNTRL,RETURN,ITEM,IANS,CANS	Show information
48	EDGETLP	CNTRL,RETURN,LINENO, PNTR, LENACT	Get address of next record
49	EDNCNVP	CNTRL,RETURN,OPTION,LINENO, CHARS	Line number conversion

3.19 Common ABEND Codes

IBM ABEND codes are documented as hexadecimal integers. Each of the ABEND codes under consideration have an IEC message that must be examined to obtain a complete description of the error that occurred. The IEC messages appear in the JES2 system log, located at the beginning of the output of each job. The ABEND codes are described in the IBM document

OS/390 MVS System Codes and the IEC messages are described in the IBM document *OS/390 MVS System Messages* (5 volumes).

The following list contains the most commonly encountered IBM ABEND codes in hexadecimal, their associated IEC message numbers, and a short description of the most common cause of the error.

The following ABEND codes may be encountered when a data set is opened by the EDOPEN(P) subroutine:

HEX	IEC Msg	Common Meaning
013	IEC141I	Conflicting or unsupported DCB parameters; member name specified on DD statement not found
213	IEC143I	Data set on disk not found
413	IEC145I	I/O error reading label on tape
613	IEC147I	I/O error in tape label processing or tape positioning
813	IEC149I	Data set name on DD statement does not match data set name in tape label
A13	IEC151I	File sequence number in LABEL on DD statement not on volume
C13	IEC153I	Output data sets cannot be concatenated

The following ABEND codes may be encountered when a data set is closed by the EDCLOS(P) subroutine:

HEX	IEC Msg	Common Meaning
214	IEC210I	I/O error in tape positioning
614	IEC214I	I/O error in writing file mark
714	IEC215I	I/O error in tape label processing
B14	IEC217I	Error in processing the directory of a PDS

The following ABEND codes may be encountered when a data set is referenced by any of the subroutines (EDOPEN(P), EDGET(P), EDPUT(P), EDCLOS(P)):

HEX	IEC Msg	Common Meaning

137	IEC022I	I/O error in end of volume processing of tape
237	IEC023I	Verification error in tape label processing
637	IEC026I	I/O error in tape label processing
737	IEC027I	Data set on disk not found for multi-volume or concatenated data set
B37	IEC030I	Not enough space available on volume or not enough space requested for disk data set
D37	IEC031I	Data set needs more space but no secondary quantity specified in SPACE on DD statement
E37	IEC032I	Not enough space available on volume or not enough space requested for disk data set

4 INDEX

Document Evaluation

Is the Manual:

	YES	NO
Clear?	☐	☐
Well organized?	☐	☐
Complete?	☐	☐
Accurate?	☐	☐
Suitable for the beginner?	☐	☐
Suitable for the advanced user?	☐	☐

Comments:

_____
_____
_____
_____
_____
_____
_____

Please give page references where appropriate. If you wish a reply, include your name and mailing address.

Send to: Application Services Branch
Division of Computer System Services, CIT
National Institutes of Health
Building 12A, Room 4011
Bethesda, MD 20892-5607

FAX to: (301) 496-6905

ICD or Agency:
Date Submitted:
Name (Optional):
E-Mail Address: