28
2-10-81
2 5000 20 NTIS

(L)

R1865

ORNL/CSD/TM-137

UCC-ND

**NUCLEAR
DIVISION**

**UNION
CARBIDE**

**Data Format
Translation Routines**

R. D. Burris

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

OPERATED BY
UNION CARBIDE CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

DISCLAIMER

CRNL/CSD/TM-137

DATA FORMAT TRANSLATION ROUTINES

R. D. Burris

Sponsors: C. E. Hammons
G. R. Haste

Date Published: February 1981

COMPUTER SCIENCES DIVISION
at
Oak Ridge National Laboratory
Post Office Box X
Oak Ridge, Tennessee 37830

Union Carbide Corporation, Nuclear Division
operating the
Oak Ridge Gaseous Diffusion Plant . Paducah Gaseous Diffusion Plant
Oak Ridge Y-12 Plant . Oak Ridge National Laboratory
under Contract No. W-7405-eng-26
for the
Department of Energy

167870-101 OF THIS DOCUMENT IS UNLIMITED *xf*

TABLE OF CONTENTS

ACKNOWLEDGMENTS	v
ABSTRACT	1
1. INTRODUCTION	3
2. TRANSLATION PACKAGE OVERVIEW	5
2.1 USER CLASSES	5
2.2 THE INPUT DATA	6
2.3 THE OUTPUT FILE	8
3. SUBROUTINE DESCRIPTIONS	11
3.1 INITIALIZATION ROUTINES	11
3.1.1 Communication Initialization	11
3.1.1.1 Experimental User	12
3.1.1.2 Communications User	13
3.1.2 Data Base Initialization	14
3.1.2.1 Experimental User	14
3.1.2.2 Communications User	14
3.1.3 General Initialization	15
3.2 TRANSLATION ROUTINES	16
3.2.1 General Information	16
3.2.2 Data Acquisition Programs	17
3.2.3 Communications Routines	18
3.3 TERMINATION ROUTINES	21
3.4 MISCELLANEOUS ROUTINES	22
3.5 BUILDING A TASK	22
4. SUMMARY	25
REFERENCES	27
APPENDIX A: DATA BASE HEADER STRUCTURE	29
APPENDIX B: SUPPORTED DATA TYPES	31
APPENDIX C: BUILDING AN OVERLAID TASK	33
APPENDIX D: EXAMPLES	35

ACKNOWLEDGMENTS

The assistance of David E. Greenwood in the definition of these translation functions is gratefully acknowledged. The efforts of Teresa Craig in the preparation of the manuscript are also acknowledged with appreciation.

Funds for the support of the development of this package were provided by the Data Handling group, Computer Support Section, Fusion Energy Division.

ABSTRACT

To enable the effective connection of several dissimilar computers into a network, modification of the data being passed from one computer to another may become necessary. This document describes a package of routines which permit the translation of data in PDP-8 format to PDP-11 or DECsystem-10 formats or from PDP-11 format to DECsystem-10 format. Additional routines are described which permit the effective use of the translation routines in the environment of the Fusion Energy Division (FED) network and the Elmo Bumpy Torus (EBT) data base.

1. INTRODUCTION

Over the last few years it has become common practice for several computers to be connected into a network so that functions unavailable on one may be performed on another. When the computers are of dissimilar types, such networking can be made more difficult in that the data from the two different computers may be completely different in their internal representation (bit pattern). When this is the case, the data must be translated to the format of the destination computer to be usable by that computer's software.

Elmo Bumpy Torus (EBT) is an experiment for studying the generation of electrical power through controlled thermonuclear fusion. Computers are used for the acquisition of data about the experiment and for the analysis of those data. Specifically, data are acquired on two PDP-8's, a Nuclear Data 6600 (ND6600), and two PDP 11/34's. Preliminary analysis is performed on the ND6600 and the 11/34's, and trend and parametric analyses are performed on a DECsystem-10.

The architectures of these computers support 12-bit, 16-bit, and 36-bit word lengths, different floating-point number definitions, and different character representations. During the design of the EBT data acquisition and analysis system,¹ it was decided that one of the PDP-11/34's would be used to do all computer-computer communications processing (and would consequently be designated the 11/34-C) and that all the other data acquisition computers would be connected to it. It was further decided to ensure that all data would be stored in the format of the computer in which it was originally stored and

that the required translations of data format would be performed on the 11/34-C.

2. TRANSLATION PACKAGE OVERVIEW

The translation facility has several levels of structure, all provided via subroutines. At the lowest (most architecture-dependent) level is the actual conversion of one bit pattern to another. However, before such a translation can be invoked, the program must know the types of the input data (ASCII, integer, floating-point, etc.). Therefore, there must be a structural level for data characterization. Finally, an additional level was imposed to aid in the routing of the input file. This layer provides the correct handling of the network command string² and the data base header information.³ In this section two classes of users will be described: the input data stream for each and the output file.

2.1 USER CLASSES

There are two types of user of the translation package--the experimenter (data originator) and the communications programmer (data router). The experimenter always knows the type of datum which must be converted. That is, if there are five numbers to be placed in a file and sent to the DECsystem-10, the experimenter will know during the writing of the program that three of those numbers will be integers and the remaining two will be floating-point. The communications program, on the other hand, is presented with a stream of bits from an outboard computer (i.e., any computer connected to the 11/34-C except the DECsystem-10) which contains information identifying the type of data contained in the stream. The program

forwards the stream to the translation package which determines the types of the data within the stream and provides the appropriate translations.

2.2 THE INPUT DATA

Two categories of input data are acceptable to the translation package--data whose type is known to the calling program (data from experimenters) and data whose type is not known to the calling (communications) program. In the former instance, subroutine arguments specify the data type, the number of units of that data, and the starting address of the stream. The number of units of a type of data is the number of ASCII characters, the number of integers, or the number of floating-point numbers. It is independent of the architecture of the originating computer. (This is in contrast to the number of words of a type of data. Often two characters are contained in a word and two or three words are required to make up one floating-point number.) Accordingly, a program could cause the translation of an entire floating-point array in one call by telling the package that the data were floating-point, the number of units in the array, and its address.

The communications program, however, is not required to inspect the data. Rather, such a program reads as much data as it wants (a function of its buffer size) and tells the translation package the size (in 16-bit words) and location of the buffer containing the current portion of the stream. The stream can include intermixed ASCII, floating-point, and integer units. However, there is a required structure to the data. The first 16-bit word presented to

the translation package must be an integer naming the first data type, the second 16-bit word must be an integer containing the number of units of those data, and then that many units of the data must appear. It is not necessary that all of the promised units be in memory simultaneously or that the buffer contain only complete units. The data stream is interpreted independently of the amount of data (buffer length) presented per call.

To demonstrate the processing of the package, consider a file consisting of a 200-unit floating-point array being received into a 99-word buffer. The following steps would be taken:

1. The communications program handles the command string.
2. It then fills the buffer with the first 99*16 bits of the data stream.
3. It calls a translation subroutine which interprets the first word as the type of the data to follow (floating-point) and the second word as the number of units of that data (200), then translates the bits into 48 floating-point variables and keeps the last 16 bits until the next call.
4. The communication program reads 99*16 more bits into the buffer.
5. It calls the same subroutine which takes the first 16 bits of the new chunk and completes the 49th floating-point unit, then makes 49 more floating-point variables of the remaining bits (with no bits left over this time).
6. The communications program again refills the buffer and calls the translation subroutine, which translates 49 more floating-point variables and saves the extra 16 bits.
7. The buffer is again refilled and the subroutine called, completing the previous unit and completing 49 more, for a total so far of 196 floating-point units complete. There remain only four units (eight words, 128 bits).
8. The communications program again tries to fill the buffer (since it does not know that the stream is nearly exhausted). An end-of-file is encountered after 128 bits, so the translation subroutine is called with a length of 8 (eight

16-bit words). The last four units are now completed and termination routines are called by the communications program.

Note that programs for outboard computers are provided with an easy interface to this system by the WDISK routine which records data in the correct structure for later translation at the 11/34-C.

2.3 THE OUTPUT FILE

Data being transferred through the Fusion Energy Division Network are in the form of files. The information determining the destination and source of the data is contained in an ASCII command string which precedes the file but is contiguous with it in its passage through the network. In the EBT system, outboard computer programs create their own command string (in the FED standard syntax), so that the communications program running on the 11/34-C must handle that command string properly. Programs running on EBT 11/34 computers have their command string created by this translation package. Whatever the source of the command string, if the file is destined for a computer not in the EBT network, it will be stored on disk in a format compatible with the FED network software and already translated to the format of the destination computer. If the file is destined for the 11/34-C, the network compatibility requirement is obviated--the communications program will use the command string to determine the disposition of the data (file name, UIC, etc.) and then discard the string.

Data files which are to be included in the data base must contain the data base header in the first 13 DECsystem-10 words of the file. Outboard computers build that header and include it in the stream

destined for the 11/34-C. PDP-11 files have the header built by the translation package, as we shall see.

10

Blank

3. SUBROUTINE DESCRIPTIONS

The subroutines comprising this package can be divided into four groups--initialization, processing, termination and miscellaneous routines. Each group will be discussed with attention to the use of the routines both by the experimenter and the communications programmer.

3.1 INITIALIZATION ROUTINES

Various functions provided by the translation package require initialization, including the communications functions, the data base functions, and the translation functions.

3.1.1 Communication Initialization

The Fusion Energy Division has a computer communications network linking its several computers.² There is a command string defined which describes the actions to be taken with the data file which follows that command string. The creation of that command string is handled by the translation routines for programs running on a PDP-11.

The command string is of the form

FED_DSK:fn.ext[P,PN]=EBC_[U,IC]fn.ext/BI

where the left side of the equal sign denotes the destination computer device, the name and extension under which the file is to be stored, and the project-programmer number (PPN) (disk area) in which it is to be stored. The right side of the equal sign denotes the source node, the originating user identification code (UIC) (equivalent to PPN, and optional), and the original file name (also optional). The characters

"/BI" inform the network programs that the file to follow is to be stored exactly as transmitted.

One of the command string syntax rules is that the side relevant to either the source or the destination computer must be in the syntax of that computer, so the sample syntax above is appropriate only for a file originating at the PDP-11 (under RSX-11M) and destined for the DECsystem-10.

3.1.1.1 Experimental User

The user of the translation routines may send files to a private area on the DECsystem-10 or to the EBT data base. In the former case, the subroutine INIT10 specifies the storage parameters for the file. In the latter case, the translation package defines those parameters.

If the user is sending the file to a private area, the INIT10 routine must be called with arguments describing the destination area. The subroutine calling sequence is:

```
CALL INIT10(IERR,LDEV,DEV,LFN,FN,LPPN,PPN)
where
```

IERR is returned =1 for success and otherwise for failure
LDEV and DEV are optional and represent the number of
characters in the device specifier and the device
specifier itself

LFN and FN are mandatory and are respectively the length
of the file name character string and the string
itself, in the form "NAME.EXT."

LPPN and PPN are also mandatory and are respectively the
length of the string and the PPN string itself.
The PPN may include any level of SFD
specification supported on the 10.

For example,

```
CALL INIT10(IEF5,6,'DSKBO:',9,'FILENA.ME',20,'[200,21455,SFDA,SF2]')
```

Note that the period in the filename is mandatory if the extension is given. Note also that the length of each string is the number of characters which are to be included in the resultant string, not the number of characters in the variable itself. The colon in the device specifier and the square brackets ("["","]") in the PPN specifier are optional and will be inserted if omitted by the caller.

The translation package will type out to the user the command string which will precede the data file into the network. A data base command string and the command string created as a result of the above example are shown below:

```
FED_DSKB:A0037.CEX[210,2740]=EBC_DLO:A0037.CEX/BI
```

```
FED_DSKB0:FILENA.ME[200,21455,SFDA,SF2]=EBC_FILENA.ME/BI
```

3.1.1.2 Communications User

If the file being received at the 11/34-C is to be kept on the 11/34-C, the communication program must call the subroutine INIT11 (with the same arguments as for INIT10). Files destined for the DECsystem-10 require no call to either INIT10 or INIT11 because the outboard computer has built the command string.

3.1.2 Data Base Initialization

3.1 .1 Experimental User

Files which are destined for EBT's data base must contain the data base header in the first 13 DEC:rvstem-10 words (see Appendix A). Whether or not a file is sent to the data base directly by the translation package, it must contain a header if it is ever to be placed in the data base. Only if the user will never put the file into the data base is the header unnecessary.

The calling sequence for the INITHD routine is shown below:

```
CALL INITHD(IERR,DIAGID,DIAKEY)
where IERR   is returned =1 for success and anything else
              for failure
      DIAGID contains the diagnostic descriptor key
              (three characters)
      DIAKEY contains the key to be used by the data base to
              unpack the data in the file (three characters)
```

For example,

```
CALL INITHD(IERR,'CEX','002')
```

3.1.2.2 Communication User

The communications programmer does not need to call INITHD, because the outboard computer program has already built the data base header.

3.1.3 General Initialization (All users)

The general initialization routine is named XINIT, and its chief function is the creation of a disk file to contain the translated data. That file will be built in the UIC of the user and will be named TMP.XXX unless the user has called INITHD, in which case the diagnostic identifier will replace "XXX."

If INITHD has not been called prior to calling XINIT, no more processing is necessary. If INITHD has been called, XINIT acquires the current time, date, and machine condition sequence number for later inclusion into the data base header. It reserves space in the beginning of the file for that header.

NOTE: The disk operations performed within the package use LOGICAL UNIT (LUN) 1. Further note that, if the user calls INITHD or INIT10, those calls must precede the call to XINIT.

The calling sequence for this routine is as follows:

CALL XINIT(IERR,ISOURC,IDEST,COMSTR)

where IERR is returned =1 for success and anything else for failure

ISOURC is a code designating the source computer

IDEST is a code designating the destination computer

COMSTR is an optional argument containing a network command string

The codes for computers are:

1	PDP-8
2	PDP-11
3	DECsystem-10
4	Nuclear Data 6600
5	PDP-8 with FORTRAN IV

A sample call would be

```
CALL XINIT(IERR,2,3)
```

to cause translation from PDP-11 format to DECsystem-10 format.

If the fourth argument is specified, the contents of the array (the command string) must obey certain very specific rules.

1. The destination node is mandatory and is followed by the underscore or backarrow character.
2. The destination file specification must be in the language of the destination computer. In the case of the DECsystem-10, the PPN field follows the extension, whereas in RSX-11M the UIC field precedes the file name.
3. The equal sign is mandatory.
4. The command string must be terminated by a line feed.
5. If the data are to be transmitted precisely as is (or if translation routines on the 11/34-C are to be called), the characters "/BI" must be included just before the line feed.

3.2 TRANSLATION ROUTINES

3.2.1 General Information

The translation routines must be able to determine the type of data upon which they are to operate and how much of that data are present, do the translations and store the results in a file. As we shall see, there are two types of translation routines: those for data acquisition programs and those for communications programs. In both cases, however, the output file will be stored in the format of the destination computer, whether it be the DECsystem-10 or an 11.

3.2.2 Data Acquisition Programs

The only translation subroutine used by data acquisition programs is X1110, which does translation from the PDP-11 to DECsystem-10 format. The calling sequence is:

```
CALL X1110(IERR,ITYPE,LENGTH,ARRAY)
where
IERR   is returned =1 for success
        =-1 for errors
ITYPE  indicates the type of data to be translated in this call.
        The possible values are shown in Appendix B.
LENGTH designates the number of units of data
        contained in the stream to be translated.
        If, for example, the array is DCUBLE PRECISION ARRAY(5),
        LENGTH=5. If the array is BYTE ARRAY(21), LENGTH=21
ARRAY  is the name of the array containing the data to be translated.
```

Sample calls are shown below:

```
CALL X1110(IERR,'IS',5,INT)
CALL X1110(IERR,'DP',35,DPR)
CALL X1110(IERR,'AE',16,'send this string')
```

where the first call translates 5 PDP-11 16-bit integers, the second translates 35 PDP-11 double-precision floating point numbers, and the last translates the indicated string.

A concealed aspect of these translations relates to which PDP-11/34 is running a data acquisition program. Because of the emphasis placed upon fail-safe operation in this system, all programs written for one 11/34 must be able to run without modification on the other 11/34. Consequently, the translation subroutines must be able to translate (program running on 11/34-C) or to store a type-count-data stream identical in form to the stream from a PDP-8 (program running on 11/34-M) which can subsequently be translated on the 11/34-C. The

translation routines have been made capable of discovering upon which 11/34 they are running, so a single subroutine can translate or not, as needed.

The package determines the host machine by determining the presence or absence of the LINKM common region via an RSX-11M directive (operating system function). If that region is present, the computer is the 11/34-C.

If the program is not running on the 11/34-C, all translation routines call a special subroutine rather than perform the indicated translation. That routine, which may also be called by the user, is called WDISK (as is the similar routine on the PDP-8) and records type, count, and data on disk without inspection.

There is another non-translating subroutine, NOTRAN, which does not record type or count. This routine has the same sort of arguments as the other translators, but for it the length argument specifies the number of 16-bit words in the array rather than the number of units of data. The ITYPE argument is ignored.

3.2.3 Communications Routines

The general function of the communications program is to accept a stream of bits (in type-count-data format) from an outboard computer and to provide an output file which has been translated to the format of the destination computer. To make the communications program as clean as possible, special translation routines were written which accept the uninterpreted stream of bits, decode it, and perform appropriate translations. The routines are named U811, U810, and U1110

for translations from 8 to 11 format, 8 to DECsystem-10 format, and 11 to DECsystem-10 format, respectively.

To access these routines, the user makes calls just as described above, i.e.,

CALL sbr(IERR, ITYPE, LEN, IARRAY)

when sbr is replaced by U811, U810, or U1110. Now, however, ITYPE may be as described in Appendix B, with results identical to the results produced by X1110; or it may be -1 to trigger the unpacking of the stream received by the communications program. If the latter is the case, LEN is taken to be the number of 16-bit words in IARRAY.

When ITYPE=-1 in U1110, for instance, the subroutine requires that the initial contents of IARRAY be such that:

1. The first word (16 bits) contains the type of the first group of data in the stream.
2. The second 16-bit word contains the number of units of the indicated type in the first group.
3. The requisite number of units follow, although they need not all be present in the buffer simultaneously.

Using these values, translation begins. Sufficient bits to contain the first unit of the specified type are moved to a work area and translated from there into the output buffer. This pattern repeats until the group is completed or until there are not sufficient bits remaining in the buffer to do another unit. At this time any remaining bits in an incomplete unit are placed in the temporary area and the routine returns with IERR=0 (indicating that the translation routines have not yet completed a group). The communication program then refills IARRAY and calls the subroutine again.

This pattern repeats until the first group is translated. The translation package then acquires the next 16 bits (if they are present in IARRAY) to determine the type of the next group, the next 16 bits (if present) to determine the number of units, etc. If the end of a group and the end of IARRAY are simultaneously encountered, IERR will be returned =1 (success); otherwise it will be returned =0 (still working on a group). The overall pattern repeats until the communication program encounters the end-of-file of the input stream. At the end-of-file the number of words containing the remaining data in the buffer are passed as LEN.

Note that the user should not meddle with LEN overmuch. Values passed in LEN must be less than or equal to the amount of data contained in IARRAY or the translation package will attempt to translate words past the end of valid data. This can cause problems if those words should be interpreted as type or length.

The U1110 routine is available to experimental users on the 11/34 as well as to communications programs. One rather interesting use is the passing of an entire COMMON block at one blast. To do so, the COMMON block must be structured into groups of similar data. Each group must be preceded by two INTEGER*2 entries containing the type and number of units, respectively. The U1110 routine would be called with ITYPE=-1, LEN = the length of the COMMON block, and the name of the first item of the block as the IARRAY argument.

Samples of the above uses follow:

```
CALL U1110(IERR,-1,50,IARRAY)
      to translate 50 16-bit words of unknown format
```

C
C
C

```

COMMON/BLAST/I1,L1,ARRAY(100),I2,L2,DPA(100)
C SET TYPES
  I1=12
  I2=15
C SET NUMBERS OF UNITS
  L1=100
  L2=100
C SET NUMBER OF WORDS IN BLAST
  LEN=1+1+100*2    +1+1+100*4
C INITIALIZATION CALLS OMITTED FOR CLARITY
  CALL U1110(IERR,-1,LEN,I1)
      to send the entire COMMON block

```

3.3 TERMINATION ROUTINES

When all data have been processed, the processing must be terminated, either normally (via XFIN) or in response to error conditions (via XABORT). In the case of normal termination XFIN performs following actions:

1. Completes the output file, including writing the last buffer and closing the file. If the file is to remain on the 11/34-C, then no more processing is required.
2. If no data base processing is to be done, go to 6.
3. Acquires the termination time and a name for the file (a diagnostic sequence number);
4. Re-opens the file for modification;
5. Creates the data base header and writes it over the first 13 DECsystem-10 words (845 bits) of the first record (which were recorded as zero when the first record was written), then closes the file.
6. Creates a network command string if there is none yet. If the file is destined for a user's area on the DECsystem-10, the arguments of the INIT10 routine are utilized in the creation of the string. If data base processing is used, the diagnostic sequence number and the data base standard PPN and device are used.
7. Invokes the NET command with the command string, after which the file is automatically processed by the communication software.

If some errors have been made or if the user has reason to do so, the file can be deleted before transmission by a call to the XABORT routine. The sole function of that routine is to delete the file built in the user's area.

The calling sequences for these routines are

```
CALL XFIN(IERR)
CALL XABORT(IERR)
where IERR is returned =1 for success and otherwise for failure.
```

3.4 MISCELLANEOUS ROUTINES

There are modules which the user does not reference directly--BASE, XLTTY, and XLDATA. BASE is the root segment of the translation package and contains some generally used routines. XLTTY contains the error messages and the teletype handler, and it can overlay the initialization and translation routines. XLDATA contains all data modified during translation processing.

3.5 BUILDING A TASK

The translation package may be loaded either directly or with overlay structure. If the former is done, the size of the package is over 4100 (10) words. If overlays are used, the size drops to about 2300 (10) words. Direct loading may be done with the following command string:

```
TKB PGM,PGM=PGM,[7,37]TRANSLATE/LB
      where [7,37]TRANSLATE.OLB is the library
      containing all the translation modules.
```

If the user desires to load using overlays, a sample overlay descriptor file is contained in Appendix C.

4. SUMMARY

The EET data acquisition and analysis system has provided the experimenters using that device with a useful, fast, efficient, and reliable computer network consisting of several dissimilar computers. The usability, speed, efficiency, and reliability were all a consequence of the use of standard hardware and software modules throughout all systems. Since the data produced or required by the software modules must be in the format expected on that computer, the data had to be translated to permit the use of the standard software. This document has described the package providing the requisite translation facility.

REFERENCES

1. R. D. Burris, D. E. Greenwood, J. S. Stanton, and K. A. Geoffroy, EBT Data Acquisition and Analysis System, ORNL/TM-7464, Oak Ridge, Tennessee (October, 1980).
2. R. D. Burris, NET-An Inter-Computer File Transfer Command, ORNL/CSD/TM-56, Oak Ridge, Tennessee (May, 1978).
3. J. S. Stanton, Elmo Bumpy Torus Data Base, ORNL/CSD/TM-136, Oak Ridge, Tennessee (in preparation).
4. R. D. Burris, EBT Data Acquisition System Control Routines, ORNL/CSD/TM-131, Oak Ridge, Tennessee (in preparation).

APPENDIX A: DATA BASE HEADER STRUCTURE

The data base header as stored on the DECsystem-10 is described below:

BYTE NUMBER	FIELD NAME	DESCRIPTION
1-2	header format number	ASCII, 2 characters, 00 for this format
3-5	diagnostic i.d.	ASCII, 3 characters. Unique for each diagnostic
6-10	diagnostic sequence number	ASCII, 5 characters, left-most alphabetic, rest numerics. Unique for each run of a diagnostic.
11-13	diagnostic descriptor key	ASCII, 3 digits, leading zeroes
14-15	filler	ASCII, 2 characters, blanks
16-20	machine condition sequence number at start	ASCII, 5 characters, leftmost alphabetic rest numerics. Unique for each change in machine conditions
21-31	start date	ASCII, 11 characters, left justified, DD-MMM-YYYY or DD-MMM-YY blank padded
32	filler	ASCII, blank
33-40	start time	ASCII, 8 characters, leading zeroes present. HH:MM:SS
41-45	machine condition at end of acquisition	as above for start conditions
46-56	end date	as above for start date
57		filler
58-65	end time	as above for start time

APPENDIX B: SUPPORTED DATA TYPES

The supported data types are listed below (Type can be specified either by code or by literal.):

LITERAL	CODE	DATA TYPE	DESTINATION FORMAT
ID	-1	Internally defined	DECsystem-10 or PDP-11 format
I8	1	PDP-8 12-bit integer	DECsystem-10 36-bit integer or PDP-11 16-bit integer
I4	2	PDP-11 32-bit integer with 16-bit range (Integer*4, FORTRAN IV)	DECsystem-10 36-bit integer
I5	5	PDP-11 16-bit integer (default, FORTRAN IV)	DECsystem-10 36-bit integer
I+	7	PDP-11 32-bit integer with 32-bit range (FORTRAN IV +)	DECsystem-10 36-bit integer
F8	11	PDP-8 36-bit floating-point	DECsystem-10 36-bit floating point or PDP-11 32-bit floating point
FE	12	PDP-11 32-bit floating-point	DECsystem-10 36-bit floating point
DP	15	PDP-11 64-bit double-precision floating-point	DECsystem-10 72-bit double-precision floating point
A8	21	PDP-8 6-bit ASCII	DECsystem-10 7-bit ASCII
AE	22	PDP-11 8-bit ASCII	DECsystem-10 7-bit ASCII
H8	29	PDP-8 6-bit ASCII but in a special header format	DECsystem-10 7-bit ASCII
L4	44	PDP-11 LOGICAL*4 (default)	DECsystem-10 LOGICAL
L2	45	PDP-11 LOGICAL*2	DECsystem-10 LOGICAL
L1	46	PDP-11 LOGICAL*1	DECsystem-10 LOGICAL

32

Blank

APPENDIX C: BUILDING AN OVERLAID TASK

The following demonstrates the building of a task using the translation package overlaid to conserve memory.

```
; The first two lines must be entered exactly this way
      .NAME   XLD000,GBL
      .PSECT  $$FSR1,RW,D,GBL,REL,OVRLY
; Below, PGM should be replaced by the user's root segment
;       and OVRLY by the user's overlaid structure
      .ROOT   PGM-Z-OVRLY
;
; The next lines force impure data to be loaded next
;       to the root and define the factor XLATE which will be used
;       shortly
; Enter the next two lines exactly.
Z:     .FCTR   XLD000-$$FSR1-*(7,37)TRANSLATE/LB:XLDATA
@[7,37]XL1110.ODL
; Note that the above is for X1110 use.  Similar files
;       exist for U1110, U810, and U811
;       Also note that XL1110.ODL is listed at the end of this appendix
;
; Now define the rest of the overlay structure
; Somewhere in the remainder of the file the factor XLATE
;       must be defined so that the remainder of the translation
;       routines can be included.  For example, OVRLY will be
;       defined next.
OVRLY: .FCTR   SAVER-XLATE
; Where saver is the user's subroutine which accesses
;       the translation package.
      .END
```

The overlay descriptor file for use with X1110 is shown below. The user does not have to type in this file if the above style of .ODL file is used.

```
;
;
XLATE: .FCTR   *XM0000-*(XF0001,XF0002)
XM0000: .FCTR   [7,37]TRANSLATE/LB:BASE
;
XF0001: .FCTR   (XM0001,XM0002)
XM0001: .FCTR   [7,37]TRANSLATE/LB:XLTTY
XM0002: .FCTR   [7,37]TRANSLATE/LB:X1110
;
```


XF0002:	.FCTR	XM0003-(XM0004, XM0005)
XM0003:	.FCTR	[7,37]TRANSLATE/LB:HEAD
XM0004:	.FCTR	[7,37]TRANSLATE/LB:XINIT
XM0005:	.FCTR	[7,37]TRANSLATE/LB:XFIN

APPENDIX D: EXAMPLES

This appendix will provide somewhat more complete examples of the uses of the routines described in the body of this report.

First we will show the calling sequence for a program running on the 11/34-C which will send two integers and a floating-point array into the data base area on the DECsystem-10. The diagnostic identifier will be CEX (charge exchange) and the descriptor key will be 003.

```

DIMENSION ARRAY(17)
CALL INITHD(IERR,'CEX','003')
CALL XINIT(IERR,2,3)
CALL X1110(IERR,'IS',1,I)
CALL X1110(IERR,'IS',1,J)
CALL X1110(IERR,'FE',17,ARRAY)
CALL XFIN(IERR)
STOP
END

```

No more would be necessary, although it would be nice if the data had meaning.

Now we complicate matters slightly by sending the data to the user's area on the DECsystem-10 rather than to the data base area. The user, however, still wants to construct the data base header.

```

DIMENSION ARRAY(23)
CALL INITHD(IERR,'CEX','003')
CALL INIT10(IERR,,,8,'TEST.FIL',13,'210,21455,SFD')
C TO PUT THE FILE INTO THE SFD [210,21455,SFD] UNDER THE NAME
C TEST.FIL
CALL XINIT(IERR,2,3)
CALL X1110(IERR,'IS',1,I)
CALL X1110(IERR,'IS',1,J)
CALL X1110(IERR,'FE',23,ARRAY)
CALL XFIN(IERR)
STOP
END

```

Now we will demonstrate the same type of activity but, since the user knows that the data will never be added to the data base, no header is built. Also, several arrays will be sent.

```

DOUBLE PRECISION DP(20)
DIMENSION ARRAY(200,5)
CALL INIT10(IERR,,,8,'TEST.FIL',9,'210,21455')
CALL XINIT(IERR,2,3)
CALL X1110(IERR,'DP',20,DP)
CALL X1110(IERR,'FE',1000,ARRAY)
C NOTE THAT, ON THE DECSYSTEM 10 AND THE PDP-11, THE LEFTMOST
C   SUBSCRIPT VARIES FIRST.
CALL X1110(IERR,'AE',17,'Send this string.')
CALL XFIN(IERR)
STOP
END

```

Now the user wants to create the command string. The following demonstrates a way to do it.

```

BYTE STRING(80)
DIMENSION FAKE(20)
EQUIVALENCE (STRING(1),FAKE(1))
DATA FAKE/'FED ','DSKA',' :DAT','A.PI','L[23','4,21',
1      '23]=' , 'EBC_', '[100',' ,40]','SEND','R.DA','1/BI',7*0/
C GENERATES THE STRING
C FED_DSKA:DATA.FIL[234,2123]=EBC_[100,40]SENDR.DAT/BI
C DEFINE THE LINE FEED
  BYTE(53)="12
C
  CALL XINIT(IERR,2,3,BYTE)
C
C AND NOW CALL TRANSLATION STUFF

```

Some examples using unpacking versions follow. First we assume that we are sending a file which is being transmitted from the 11/34-M to the DECsystem-10. We must call XINIT to keep the command string unmodified, then call U1110 to unpack and translate the data.

```

DIMENSION TARRAY(200)
C

```

```

C FIRST GET THE COMMAND STRING. SUBROUTINE FILLER, WRITTEN BY
C   THE USER, WILL FILL TARRAY
C   IT RETURNS IERR=-1 IF NO END-OF-FILE IS ENCOUNTERED
C   OR +NUMBER TO INDICATE END-OF-FILE AND HOW MANY
C   WORDS WERE PLACED INTO TARRAY
C   CALL FILLER(TARRAY,IERR)
C   IF(IERR.NE.-1)STOP

C       CALL XINIT(IERR,1,3,TARRAY)
C WHICH PUTS THE COMMAND STRING IN TARRAY INTO THE OUTPUT FILE
C
C GET THE DATA AND TRANSLATE IT
100  CALL FILLER(TARRAY,IERR)
      IF(IERR.NE.-1)GO TO 200
      CALL U1110(IERR,-1,200,TARRAY)
      GO TO 100

C
C HERE WHEN END-OF-FILE
200  CONTINUE
      ILEN=IERR
      CALL U1110(IERR,-1,ILEN,TARRAY)
      CALL XFIN(IERR)
      STOP
END

```