UTILITY MANUAL

TABLE OF CONTENTS

Revised 6/15/83

CENTURION

CPU-6

DEALER SUPPORT MANUAL

SYSTEM UTILITY PROGRAMS

Includes 12/08/83 Revisions

CENTURION COMPUTER CORPORATION
1780 Jay Ell Drive
Richardson, Texas 75081

Printed in the USA

```
***************
*             *
*    COPUT    *
*             *
***************
```

PURPOSE

COPUT is the utility used to copy disk files. The utility requires 8K bytes of memory to run. The main module (COPUT) copies from any file , except 'D' type, to any other file, except 'D' type, one track at a time. As the file is copied a record count (or sector count for type 'C' and 'L' files) is kept, and at the end of the copy, it is displayed on SYSLOG along with other file statistics.

PROCEDURE

The following is a list of system assignments necessary to run the utility:

SYSLOG	Log Output (device independent)
SYS000	Input File (disk only)
SYS001	Output File (disk only)

If an error is encountered during the copy, an error message will be displayed followed by the message *** COPY ABORTED ***. The utility will then set the completion code to 100 and terminate.

ERROR MESSAGES

FILE TYPE MIS-MATCH
: The file type of the output file was not the same as the file type of the input file.

EOM ON OUTPUT
: An EOM was reached on the output file before an EOF was detected on the input file, i.e. the output file was shorter than the input file.

CONVERSION UTILITIES

These utilities are designed to smooth the transition from a CPU5 to a CPU6. They are used to convert disk files from CPU5 formatted disks to CPU6 formatted disks, or from CPU6 to CPU5. Because of the complexity of converting the many file types, it is necessary to have several utilities.

These utilities are:

COPSC	CPU5 to CPU6, keeping the same file type (used for "A","B", or "E" type files only)
COPCS	CPU6 to CPU5, keeping the same file type (used for "A","B", or "E" type files only)
COPCSC	CPU5 to CPU6, random/spanned "C" type files on CPU5 to "C" type CPU6 files
COPISC	CPU5 to CPU6, VSI indexed "C" type files on CPU5 to "I" type CPU6 files
COP4SC	CPU5 to CPU6, "C" type four-byte indexed files on CPU5 to "C" type CPU6 files
COP6SC	CPU5 to CPU6, "C" type six-byte indexed files on CPU5 to "C" type CPU6 files
PCVT	CPU5 to CPU6, creates a jobstream, which, when executed, converts an entire CPU5 formatted disk to a CPU6 formatted disk; or an entire private library from a CPU5 disk to a library of the same name on a CPU6 disk
ALOC5	allows a CPU6 system to access a CPU5 formatted disk, for the purpose of performing many standard JCL functions

```
***************
*             *
*    COPSC    *
*             *
***************
```

PURPOSE

COPSC is the utility used to copy and convert "A", "B", and "E" type files from a CPU5 formatted disk to a CPU6 formatted disk. As the file is converted, a record count is kept and displayed on SYSLOG along with other file statistics.

PROCEDURE

The following is a list of system assignments necessary to run the utility:

```
SYSLOG    Log Output (device independent)
SYS000    CPU5 disk as Input
SYS001    CPU6 disk file as output
SYSRDR    Control input
```

SYSRDR input can be in either of the following forms:

```
INPUT = filename
        or
INPUT = lname.file
```

An example of usage of COPSC is the following:

```
.USE DISK2 for SYS0
.USE AFILE on 1 for SYS1
.RUN XCOPSC
INPUT=AFILE
.END
```

If an error is encountered in the copy/conversion, the input file is not damaged and an error message will be displayed. This will be followed by the message * * * COPY ABORTED * * *. The utility will then set the completion code to 100, and terminate.

ERROR MESSAGES

FILE TYPE MIS-MATCH
The file type of the output file was not the same as the file type of the input file.

EOM ON OUTPUT
The output disk or the output library is full.

```
***************
*             *
*    COPCS    *
*             *
***************
```

PURPOSE

COPCS is the utility used to copy and convert "A", "B", and "E" type files from a CPU6 formatted disk to a CPU5 formatted disk. As the file is converted, a record count is kept and displayed on SYSLOG along with other file statistics.

PROCEDURE

The following is a list of system assignments necessary to run the utility:

```
SYSLOG    Log Output (device independent)
SYS000    CPU6 disk file as Input
SYS001    CPU5 disk as output
SYSRDR    Control input
```

SYSRDR input can be in either of the following forms:

```
OUTPUT = filename
       or
OUTPUT = lname.file
```

An example of usage of COPCS is the following:

```
.USE AFILE on 1 for SYS0
.USE DISK2 for SYS1
.RUN XCOPCS
OUTPUT=AFILE
.END
```

If an error is encountered in the copy/conversion, the input file is not damaged and an error message will be displayed. This will be followed by the message * * * COPY ABORTED * * *. The utility will then set the completion code to 100, and terminate.

ERROR MESSAGES

FILE TYPE MIS-MATCH
The file type of the output file was not the same as the file type of the input file.

EOM ON OUTPUT
An EOM was reached on the output file before an EOF was detected on the input file, i.e. the output file was shorter than the input file.

```
***************
*             *
*    COPCSC   *
*             *
***************
```

PURPOSE

COPCSC is the utility used to copy and convert "C" type random/spanned files from a CPU5 disk to a CPU6 disk. As the file is converted, a record count is kept and displayed on SYSLOG along with other file statistics.

PROCEDURE

The following is a list of system assignments necessary to run the utility:

```
SYSLOG    Log Output (device independent)
SYS000    CPU5 disk as Input
SYS001    CPU6 disk file as output
SYSRDR    Control input
```

SYSRDR input can be in either of the following forms:

```
INPUT = file, RECSIZ=n, /*
      or
INPUT = file, RECSIZ=n, SPANNED, /*
```

An example of usage of COPCSC is the following:

```
.USE DISK2 for SYS0
.USE CFILE on 1 for SYS1
.RUN XCOPCSC
INPUT=CFILE, RECSIZ=101, /*
.END
```

If an error is encountered in the copy/conversion, the input file is not damaged and an error message will be displayed. This will be followed by the message * * * COPY ABORTED * * *. The utility will then set the completion code to 100, and terminate.

ERROR MESSAGES

FILE TYPE MIS-MATCH
The file type of the output file was not the same as the file type of the input file.

EOM ON OUTPUT
The output disk or the output library is full.

```
***************
*             *
*    COPISC   *
*             *
***************
```

PURPOSE

COPISC is the utility used to copy and convert "C" type, VSI indexed files on a CPU5 disk to an "I" type file on a CPU6 disk. As the file is converted, a record count is kept and displayed on SYSLOG along with other file statistics.

PROCEDURE

The following is a list of system assignments necessary to run the utility:

```
SYSLOG    Log Output (device independent)
SYS000    CPU5 disk as Input
SYS001    CPU6 disk file as output
SYSRDR    Control input
```

SYSRDR input should be in the following form:

```
INPUT = file, /*
```

An example of usage of COPISC is the following:

```
.USE DISK2 for SYS0
.USE IFILE on 1 for SYS1
.RUN XCOPISC
INPUT=IFILE, /*
.END
```

If an error is encountered in the copy/conversion, the input file is not damaged and an error message will be displayed. This will be followed by the message * * * COPY ABORTED * * *. The utility will then set the completion code to 100, and terminate.

ERROR MESSAGES

FILE TYPE MIS-MATCH
The file type of the output file was not the same as the file type of the input file.

EOM ON OUTPUT
The output disk or the output library is full.

```
***************
*             *
*    COP4SC   *
*             *
***************
```

PURPOSE

COP4SC is the utility used to copy and convert "C" type, four-byte indexed files on a CPU5 disk to a "C" type file on a CPU6 disk. As the file is converted, a record count is kept and displayed on SYSLOG along with other file statistics.

PROCEDURE

The following is a list of system assignments necessary to run the utility:

```
SYSLOG    Log Output (device independent)
SYS000    CPU5 disk as Input
SYS001    CPU6 disk file as output
SYSRDR    Control input
```

SYSRDR input should be in the following form:

```
INPUT = file, RECSIZ=n, /*
```

An example of usage of COP4SC is the following:

```
.USE DISK2 for SYS0
.USE CFILE on 1 for SYS1
.RUN XCOP4SC
INPUT=CFILE, RECSIZ=101, /*
.END
```

If an error is encountered in the copy/conversion, the input file is not damaged and an error message will be displayed. This will be followed by the message * * * COPY ABORTED * * *. The utility will then set the completion code to 100, and terminate.

WARNING!! CPU6 output must have been previously initialized with the XRINT utility!!

ERROR MESSAGES

FILE TYPE MIS-MATCH
The file type of the output file was not the same as the file type of the input file.

EOM ON OUTPUT
The output disk or the output library is full.

```
***************
*             *
*   COP6SC    *
*             *
***************
```

PURPOSE

COP6SC is the utility used to copy and convert "C" type, six-byte indexed files on a CPU5 disk to a "C" type file on a CPU6 disk. As the file is converted, a record count is kept and displayed on SYSLOG along with other file statistics.

PROCEDURE

The following is a list of system assignments necessary to run the utility:

```
SYSLOG    Log Output (device independent)
SYS000    CPU5 disk as Input
SYS001    CPU6 disk file as output
SYSRDR    Control input
```

SYSRDR input should be in the following form:

```
INPUT = file, RECSIZ=n, /*
```

An example of usage of COP6SC is the following:

```
.USE DISK2 for SYS0
.USE CFILE on 1 for SYS1
.RUN XCOP6SC
INPUT=CFILE, RECSIZ=101, /*
.END
```

If an error is encountered in the copy/conversion, the input file is not damaged and an error message will be displayed. This will be followed by the message * * * COPY ABORTED * * *. The utility will then set the completion code to 100, and terminate.

WARNING!! CPU6 output must have been previously initialized with the X?RINT utility!!

ERROR MESSAGES

FILE TYPE MIS-MATCH
The file type of the output file was not the same as the file type of the input file.

EOM ON OUTPUT
The output disk or the output library is full.

```
***************
*             *
*    PCVT     *
*             *
***************
```

PURPOSE

PCVT is the utility which will create a JCL jobstream to convert an entire CPU5 formatted disk to a CPU6 formatted disk. PCVT can also be used to convert private libraries in CPU5 format to a library of the same name in CPU6 format. The jobstream created by PCVT will convert all "A", "B", and "E" type files. It will not convert "C" or "D" type files, but will generate a .NEW statement for each in the first pass of the utility. Each "C" type file must then be converted using one of the other conversion utilities, and the "D" type files may then be converted using the PCVT utility, on a separate pass, to "L" type files.

PROCEDURE

The following is a list of system assignments necessary to run the utility:

SYSLOG	Log Output (device independent)
SYS002	'A' type disk file to receive JCL jobstream
SYSIPT	CRT input and output of operator messages

PCVT will solicit the following information through SYSIPT:

"ENTER LIBRARY NAME (OR NEWLINE)"/

Operator enters the private library name to be converted, or NEWLINE if the entire pack is to be converted. If a name is entered, the output library must have previously been created before the resultant jobstream is executed.

"ENTER UTILITY LIBRARY NAME (OR NEWLINE)"/

Operator enters the library name which contains the conversion utilities programs (normally CVT), or NEWLINE if the conversion programs are not in a library.

"COPY FROM="/

Operator enters the disk number for the CPU5 formatted input disk.

"COPY TO="/

Operator enters the disk number for the CPU6 formatted output disk.

"ENTER DEVICE/FILE FOR SYSLOG"/

Operator enters the device or filename to be used as SYSLOG when the resultant jobstream is executed (such as CRT#, PRT#, PRTQ for spooler, or an 'A' type disk file).

""IS THIS A DISK FILE??"/

Operator enters "Y" or "N" if the previous name entered is a disk file name.

"ON:"/

If the previous question was answered "Y", the operator now enters the disk number where the 'A' type disk file to be used for SYSLOG is located. This message will not appear if the previous question was answered "N".

"CLEAR OBJECT PACK? (Y OR N)"/

Operator enters "Y" or "N" (yes or no) if the output (object) pack or library is or is not to be cleared initially.

"SELECTIVE COPY? (Y OR N)"/

Operator answers yes ("Y") if individual file selection for conversion is desired, or no ("N") if the entire library/disk is to be copied.

"filename"
"COPY?"/

If "Y" was answered for the previous question, PCVT will display each file found on the input disk, and will solicit whether or not it should be converted. Operator may answer either "Y" or "+" for yes, or "N" or "-" for no.

Each conversion in the resultant JCL jobstream is followed by an .END statement. If an error condition should occur, the operator merely answers "Y" to the "CANCEL?" question, and the jobstream will continue processing the next conversion.

```
***************
*             *
*    ALOC5    *
*             *
***************
```

PURPOSE

ALOC5 is the utility used to access and allocate CPU5 formatted disks while executing on a CPU6. It performs many standard JCL functions on a CPU5 disk. Other than conversion utilities, it is the only program on a CPU6 which may manipulate CPU5 disks. It will operate either as an interactive program accepting input from an operator, or as a jobstream program reading input control from the jobstream file.

PROCEDURE

The following is a list of system assignments necessary to run the utility:

```
SYSLOG    Log Output (device independent)
SYS000    CPU5 disk, if mounted on DISK0
SYS001      "   "          "     on DISK1
.
.
.
SYSnnn    CPU5 disk, if mounted on DISKnnn
SYSRDR    Control Input
```

If UPSI is equal to zero, ALOC5 will not echo input, solicit commands, and will terminate upon error, setting CC to 100. If UPSI is non-zero, ALOC5 will operate interactively, and not terminate upon error.

The only unique command in the menu list is the *EXP command, which expands CPU5 files. All other commands are standard JCL CPU5 commands, with the command flag set to '*' rather than '.'.

If ALOC5 detects an error, it will write to SYSLOG an appropriate message signifying the error, and, if UPSI is zero, it will terminate, setting the completion code to 100.

The control commands which may be input through SYSRDR are listed in the following menu:

```
/=====================================================================
/
/         ALOC5 WILL ACCESS, ALLOCATE, AND EXPAND CPU 5 FORMATTED
/       DISK FILES UNDER A CPU 6 OPSYS. THE COMMANDS ARE:
/
/       1.  *DIR D [(MMMMMM)] ['T'] [TEMP] [PERM] [DATE]
/
/       2.  *NEW FILNAM [ON] D ['T'] SIZE [TEMP] [PERM]
/
/       3.  *EXP FILNAM [ON] D [BY] SIZE
/
/       4.  *NAM FILNAM1 [ON] D [TO] FILNAM2
/
/       5.  *DEL FILNAM [ON] D
/
/       6.  *FSPEC [LIBNAM.]FILNAM ON D [P=SIZE],[P=TYPE]
/
/       7.  *CLR D VOL=VVVVVV [DSIZE=N]
/
/       8.  *END, END, OR /*    (END OF EXECUTION)
/
/=====================================================================
ALOC5 - 6.02
```

CAUTIONS

The only caution associated with ALOC5 is, that, if ALOC5 is attempted on a CPU6 formatted disk, the program will terminate in an error condition.

```
***************
*             *
*    DSORT    *
*             *
***************
```

PURPOSE

DSORT is the utility used to sort and merge the contents of disk files. Parameter values for the sort/merge utility may be entered from a jobstream (via SYSRDR) or from a device/file (via SYSIPT).

Note: If sort parameters are to be input from SYSRDR, UPSI should be set to 0; if sort parameters are to be input from SYSIPT, UPSI should be set to 1.

STANDARD PROCEDURE

DSORT is designed primarily as a stand-alone utility that sorts/merges the contents of one or more disk files. It is possible, however, to use the individual load modules from the DSORT utility within user-written programs. For a discussion of the standard usage of the DSORT utility, see the information contained in this section (STANDARD PROCEDURE). For a discussion of the alternate uses of the DSORT utility, see the ALTERNATE PROCEDURE section.

The following device/file assignments must be made before execution of the sort/merge utility.

SYSRDR/SYSIPT	Sort parameter input
SYS000	Sort file output (omit if output file is equal to input file)
SYS001	Sort file INPUT1
.	.
.	.
.	.
.	.
SYS00M	Sort file INPUTM
SYS00M+1	Sort work FILE1
.	.
.	.
.	.
.	.

Type 'B' work files must be used if:

(1) Output type is VSI and (RECSIZE + ARGLEN) LE 397

(2) Output type is 'A', 'B' or 'C' and (RECSIZE) LE 397

Type 'C' work files must be used in all other instances. When type 'C' files are used, the work files must be at least N sectors; N= (RECSIZE -1)/400+2.

The parameter list may be on as many lines as needed. It is key word oriented (i.e. parameters may be in any order). Each parameter is composed of a key word followed by an equal size and the parameter operand value. Parameters may be separated by either commas or blanks. The end of the parameter list is signified by '/*' or 'END'.

Parameters for the sort/merge program are as follows:

1. PRINT=

The PRINT= parameter controls the output of information concerning the sort/merge to the device/file assigned to SYSLOG. Possible values for the PRINT= parameters are:

PRINT=NO	Error messages (default)
PRINT=YES	Parameter statements and error messages
PRINT=ALL	Parameter statements, sort/merge statistics and error messages

Messages

(1) Sort/merge statistics printed after each pass -

END OF SORT PASS or END OR MERGE PASS n

(2) Messages printed at the end of the sort/merge program -

RECORDS READ	n
RECORDS WRITTEN	n
SORT PASSES	n
RECORDS PER PASS	n
MERGE PASSES	n

(3) Message printed for each input file if default input/output modules are used -

RECORDS READ FROM 'FILENAME' n

2. INPUT=

The INPUT= parameter indicates the number of input files.

3. FILTYP=[FILETYPE=]

The FILTYP= parameter specifies the type of disk file used for input and output. Possible values are:

FILTYP=A	ASCII
FILTYP=B	Binary
FILTYP=C	Random Access
FILTYP=SPAN	Random Spanned
FILTYP=I	Variable Spanned Indexed (VSI)

4. RECSIZ=[RECSIZE=]

RECSIZ= indicates the maximum record size of the logical records in the input/output files. If the output file type is 'I', then the value of the RECSIZ= parameter must be the sum of the prime data record length plus the argument length. This is valid for both input and output RECSIZ.

5. WORK=

The WORK= parameter specifies the number of work files available to the sort/merge program. The integer value must be a multiple of 2 (two) in a range of from 4 (four) to 14 (fourteen).

Note: Work files are not needed for the "merge only" option; the WORK= parameter need not be specified in this instance.

6. OUTPUT=

The OUTPUT= parameter specifies that (1) the sorted records should be written to a new file or (2) the input file should be overwritten.

If more than one input file is used and OUTPUT=INPUT is specified, the first input file (i.e. the one assigned to SYS001) will be used as the output file; only one output file may be used, however. If the input file type is VSI or FUNCTION=MERGE, OUTPUT=NEW must be used.

Output files may be specified in one of the following ways:

OUTPUT=NEW	New output file
OUTPUT=INPUT	Same as input file
OUTPUT=INP	Same as input file

7. KEY=

The KEY= parameter specifies the logical locations and types of the sort keys in the input records as well as the order in which they are to be sorted. A maximum of 10 keys may be specified in this parameter.

There are four fields for each key specified. These fields include:

(1) Key location in the record

Note: If the output file is VSI, the argument used as a sort key. This is specified by setting the key location to 0 (zero).

(2) Key length (in bytes)

(3) Key type

Possible key type values and the type of sort compare used include:

ZD Zoned Decimal
(Numeric integer compare of fixed length field. Contains ASCII digits 0 - 9, + or -.)

PD Packed Decimal

BI Binary Integer
(Numeric binary compare with sign. Length 1 implies 8 bits, length 2 implies 6 bits, etc.)

AV ASCII Variable Length
(Variable length string compare - CPL string. String is terminated by a binary zero (0). Lower case and upper case characters are considered equivalent.)

AF ASCII Fixed Length
(Fixed length string compare. Lower case characters considered equivalent to upper case characters.)

CH Character
(Fixed character string compare. Lower case characters are considered greater than upper case characters.)

(4) Order

The order parameter is optional and will default to ascending if omitted. Possible values for the order parameter are:

AS Ascending sequence
DE Descending sequence

8. INPFIL [INPFILE]/OUTFIL[OUTFILE]=

The INPFIL/OUTFIL= parameter is used when the input file structure is different from the output file structure (i.e. the record size and/or file type may be diferent for input and output files). If the output file is VSI, the input file(s) must be VSI.

The format for this parameter is:

```
INPFIL[INPFILE]=(RECSIZ[RECSIZE]=N,FILTYP[FILETYPE]=T)
OUTFIL[OUTFILE]=(RECSIZ[RECSIZE]=N,FILTYP[FILETYPE]=T)
```

9. INPMOD/OUTMOD=

The INPMOD/OUTMOD= parameter specifies a user-written input or output module.

Note: The INPMOD= parameter is used for sort only; it may not be used with FUNCTION=MERGE.

The input module reads a record from the input file into the record area. Register A points to this record area, while Register B contains the address of the common area.

Note: The format of the common area is outlined in the ALTERNATE PROCEDURE section.

If reading is successful, Register A must be set to zero (0) on return; if end-of-file is found, Register A should be set on non-zero. Additionally, condition codes must be set to the value of Register A on return.

The output module writes a record to the output file. Register A contains the address of the record to be written upon entry, while Register B contains the address the common area. An end-of-merge is indicated if Register A is zero (0) upon entry. In this instance, an end-of-file mark should be written on the output file.

The value in Register Z must be saved and restored in each input/output module. The format of this parameter is:

```
INPMOD=(M,INAME)
OUTMOD=(N,INAME)
```

If the output module is not specified, the input module will be used for output as well. If the disk number M/N is not specified, the run disk will be used.

10. MEMORY=

This MEMORY= parameter specifies the number of kilobytes of memory used for the sort work area. One kilobyte is equal to 1024 bytes (i.e. K=1024); the default is 12K.

The format for the MEMORY= parameter is:

```
MEMORY=N[K]
```

11. FUNCTION=

The FUNCTION= parameter specifies whether the input files are to be (1) sorted and merged or (2) merged only. If the "sort/merge" option is specified, the input files must already be in sorted sequence. If the "merge only" option is specified, VSI files may not be used.

The format for the FUNCTION= parameter is:

```
FUNCTION=SORT   **
FUNCTION=MERGE
```

(** Indicates the default)

12. SELECT=

The SELECT= parameter specifies the keys for selecting records to be included in the output file. These keys need not be the same as the sort keys, however.

A maximum of six (6) keys may be specified in this parameter. In order for a record to be selected, it must satisfy all conditions specified in the SELECT= parameter. Additionally, there are five (5) fields for each key specified. These field include:

(1) Select key location in record

If the output file is VSI, the argument may be used as a select key. This is specified by setting the key location to zero (0).

(2) Select key length (in bytes, maximum 10)

(3) Select key type

Possible select key types are the same as those detailed in the KEY= parameter.

(4) Select key relation

Possible values are:

EQ	Equal to
GE	Greater than or equal to
GT	Greater than
LE	Less than or equal to
LT	Less than
NE	Not equal to

(5) Comparison value

The comparison value is specified in decimal rather than binary for the binary select key.

Examples of usage of the DSORT utility include:

```
PRINT=ALL, INPUT=1, FILTYP=A, RECSIZ=80
OUTPUT=NEW  MEMORY=8K
FUNCTION=MERGE
KEY=(12,6,AV,3,3,BI)
SELECT=(22,4,BI,EQ,45,29,3,AF,GT,ABC)/*
```

```
PRINT=NO, INPFILE=(RECSIZ=132,FILTYPE=A)
INPUT=2 OUTFIL=(RECSIZ=80,FILTYP=C)
WORK=6 OUTPUT=INP, INPMOD=(2,INAME)
OUTMOD=(,ONAME), MEMORY=6K,FUNCTION=SORT
KEY=(5,4,CH,DE,11,2,AF,13,4,BI,AS)
END
```

ALTERNATE PROCEDURE

Alternate usage of DSORT includes (1) the sort/merge utility module, (2) the sort pass module or (3) the merge pass module. These modules may be loaded and called from another program with parameters being passed through a common area. For a discussion of the Common Area Format, see the attached chart.

Sort/Merge Utility - "DSORT"

The address of the common area may be passed in the "X" Register. If X=0, the parameter list will be read from SYSRDR or SYSIPT (see STANDARD PROCEDURE, page). Load and transfer control to DSORT (no return).

Sort/Pass - '@SORT'

The address of the common area must be passed in the "Z" Register. Linkage to the "sort pass" is via subroutine call:

```
JSR address
```

The IO module(s) must be loaded and the entry point(s) set in the common area before loading '@SORT'. One of the following default IO modules may be used:

```
Input file type = 'I' load '@SM101'
Otherwise       =     load '@SM100'
```

Merge/Pass - '@MERGE'

The address of the common area must be passed in the "Z" Register. Linkage to the "merge pass" is via subroutine call with one parameter:

```
JSR address
DW  0
```

The IO module(s) must be loaded and the entry point(s) set in the common area before loading '@MERGE'. One of the following default modules may be used:

```
Input file type = 'I' load '@SM101'
Otherwise       =     load '@SM100'
```

Common Area Format

1. Parameter input flag (1 byte)
   - 0 = do not process parameters
   - 1 = process parameters

2. Parameter list memory address (2 bytes)
   - 0 = no parameter list in memory

3. Input file type (1 byte)
   - 0 = span
   - 1 = B
   - 2 = A
   - 3 = C
   - 6 = I

4. Input record size (2 bytes)

5. Output file type (1 byte)
   - 0 = span
   - 1 = B
   - 2 = A
   - 3 = C
   - 6 = I

6. Output record size (2 bytes)

7. Number of input files (2 bytes)

8. Number of work files (2 bytes)
   - 0 = merge only option

9. Output option (1 byte)
   - 0 = new
   - 1 = input/inp

10. Print option (1 byte)
    - 0 = no
    - 1 = yes
    - 2 = all

11. Input module memory address (2 bytes)

12. Output module memory address (2 bytes)

13. Input module disk number (1 byte)
    - -1 = run disk

14. Input module file name (prmlen X 2 + 1 byte)

15. Output module disk number (1 byte)
    -1 = run disk
    -2 = unused

16. Output module file name (prmlen X 2 + 1 byte)

17. Sort key table (10 entries of 6 bytes each)
    Key location (2 bytes
        0 = VSI argument
    Key length (2 bytes)
    Key type (1 byte)
        0 = ZD
        1 = PD
        2 = BI
        3 = AV
        4 = AF
        5 = CH
    Order (1 byte)
        0 = AS
        1 = DE

18. Sort key table terminator (2 bytes)
    Must be set to -1

19. Select key table (6 entries of 16 bytes each)
    Key location (2 bytes)
        0 = VSI argument
    Key length (2 bytes)
    Key type (1 byte)
        0 = ZD
        1 = PD
        2 = BI
        3 = AV
        4 = AF
        5 = CH
    Relation (1 byte)
        2 = LT
        3 = LE
        4 = EQ
        5 = GE
        6 = GT
        7 = NE

    Comparison value (10 bytes)

20. Memory (1 byte)

Unless otherwise noted, unused entries must be set to -1

ERROR MESSAGES

Abort conditions for the DSORT utility include the following:

AB50 - Parameter Error
AB51 - IO Error on Work File Output
AB52 - EOM on Work File Output
AB53 - IO Error on Work File Input
AB54 - Insufficient Memory Available
AB60 - IO Error on Sort/Merge Input
AB61 - IO Error on Sort/Merge Output
AB62 - EOM on Sort/Merge Output

```
**********
*        *
*  FILMV *
*        *
**********
```

PURPOSE

The utility creates a jobstream to move selected files from one disk to another. The source files may all be discrete, or may all be part of a single library. A single jobstream may not be used to move both discrete files and subfiles. The new files created by the jobstream may be discrete files or may be placed in an output library.

PROCEDURE

1. The utility is run by the jobstream FMOV. Enter:

   → [S.]FMOV file d

   where

file	is the name of the file in which the move jobstream is to be constructed, and
d	is the number of the disk on which that file is located. The file must already exist; it will not be created by the utility.

2. CRT then displays:

   JOB NAME=

   Enter the name to be placed in the .JOB command; this will normally be the same as the file name.

3. CRT then displays:

   ENTER PARAMETER NUMBER TO BE USED FOR SIZE:

   Enter the number of a general parameter (2-9) to be used to contain the size of the existing file in an .FSPEC command. Parameter eight is most commonly used for this purpose.

4. CRT then displays:

   ENTER PARAMETER NUMBER TO BE USED FOR TYPE:

   Enter the number of a general parameter to be used to contain the type of the existing file. Parameter seven is most commonly used for this purpose.

5. CRT then displays:

   ENTER INPUT LIBRARY NAME (OR NEWLINE)

   Enter one of the following:

name	the name of the library in which the source files are located, if they are to be copied out of a library.
NEWLINE	if discrete source files are being copied.

6. CRT then displays:

   ENTER OUTPUT LIBRARY NAME (OR NEWLINE)

   Enter one of the following:

name	the name of the library into which the files will be moved, if they are to be copied into a library.
NEWLINE	if discrete target files are to be output.

7. CRT then displays:

   ENTER UTILITY LIBRARY NAME (OR NEWLINE)

   Enter one of the following:

name	the name of the systems utility library (usually S) in which the file XCOPUT resides.
NEWLINE	if XCOPUT exists as a discrete file.

8. CRT then displays:

   FILES=

   Enter one of the following:

names	one or more names of files to be copied by the jobstream, separated by spaces or commas. As many files as will fit in a line (132 characters) may be specified. Repeat this step.

END or + if you have finished entering file names. The program terminates.

The jobstream produced by FILMV has the following statements for each file name.

```
.USE [s.]file ON #0 FOR SYSO SHAR
.FSPEC [t.]file ON #1 n=SIZE
.SKIP 2 IF 0 NE #n
.FSPEC [s.]file ON #0 n=SIZE n=TYPE
.NEW [t.]file ON #1 '#n' #n
.USE [t.]file ON #1 FOR SYS1
.RUN [S.]XCOPUT
.END
```

where

file is the name of the file to be copied, as entered during FILMV,

s is the name of the library from which the file is to be copied (if appropriate),

t is the name of the library into which the file is to be copied (if appropriate), and,

n represents a general parameter (2-9).

When the files are to be moved, the jobstream is called up by entering:

```
file d1 d2
```

where

file is the name of the jobstream created by FILMV,

d1 is the number of the disk on which the files are currently located, and

d2 is the number of the disk onto which the files are to be copied.

CAUTIONS

1. The jobstream created by FILMV does not verify that the source file actually exists before it attempts to copy it. Should the name of the file have been entered improperly, or not be found for any other reason, an error message (JX13) will interrupt the jobstream.

2. The jobstream, as created by FILMV, does not include the various messages and formatting commands recommended for a standard Centurion move jobstream. This must be added by the programmer.

```
***************
*             *
*     OPT     *
*             *
***************
```

PURPOSE

The utility program XOPT was written to provide a simple means of job parameter input and validation for MENU jobstreams.

The program will accept operator input and validate the general parameters 0 (key number), 1 (disk number), and 2 (print-device).

The parameter(s) requested by the program are determined by the value set in UPSI when the program is run:

UPSI	Parameter requested for validation
0	Parameter 0 (key number)
1	Parameter 1 (disk number)
2	Parameter 2 (print option)
3	Parameters 0 and 1
4	Parameters 0, 1, and 2

When the program is run with UPSI set to one of the values specified, the program will perform the specified operations as described below, reset UPSI to zero, and close. If the program is run with UPSI preset to any other value, the program will close without action.

The program does not display any messages other than those described below, and all output is done on line 23 of the CRT screen.

PROCEDURE

Key Number Entry/Validation

The program will erase line 23 of the CRT screen and display the message:

* ENTER KEY NUMBER FOR FILE SET

The valid responses are:

1. A one- to three-character string (validate new key number), or

2. A NEWLINE, plus bar, or minus bar (input current parameter zero and validate as a key number)

If the program can validate the input as a legal key number, the character group will be placed in parameter zero. If the program cannot validate the input as a legal key number, the program will output a BELL character and repeat the prompt message.

An input character group will be considered a legal key number if it is from one to three characters in length, and contains only the ASCII characters A-Z or digits 0-9.

Disk Number Entry/Validation

The program will erase line 23 of the CRT screen and display the message:

* ENTER DISK NUMBER FOR FILE SET

The valid responses are:

1. A number from zero through ten (validate new disk number), or

2. A NEWLINE, plus bar, or minus bar (input current parameter one and validate as a disk number)

If the program can validate the input as a legal disk number, the character(s) will be placed in parameter one. If the program cannot validate the input as a legal disk number, the program will output a BELL character and repeat the prompt message.

Print Option Entry/Validation

The program will erase line 23 of the CRT screen and display the message:

* ENTER PRINT-DEVICE OPTION

The valid responses are:

1. A NEWLINE, plus bar, or minus bar (input current parameter two and validate as a print option), or

2. 0 Representing PRT0
   1 PRT1

2	PRT2
3	PRT3
4	PRT4
5	PRT5
6	PRT6
7	PRT7
8	PRT8
9	PRT9
10	PRT10
11	PRT11
12	PRT12
13	PRT13
14	PRT14
15	PRT15
16	PRT16
17	PRT17
18	PRT18
19	PRT19
Q	PRTQ
R	PRTR
CRT	CRT#V (Video terminal)
TYPE	CRT#V (Hard-copy terminal)
LST	@LST#I (Partition list file)
NO	Dummy output assignment

If the program can validate the input as a legal print option, the character(s) will be placed in parameter two. If the program cannot validate the input as a legal print option, the program will output a BELL character and repeat the prompt message.

```
***************
*             *
*    RINT/    *
*    ?RINT    *
*             *
***************
```

PURPOSE

The utility RINT is used to initialize a 'C' type random-access file so that it may be accessed by the GETKEY and NEWKEY subroutines (4-byte arguments). ?RINT formats a file for use with ?GKEY and ?NKEY subroutines (6-byte arguments).

RINT and ?RINT read parameters from either SYSRDR (UPSI=0) or SYSIPT (UPSI<>0).

1. RECSIZ=
   This keyword must be used to specify the size of the prime data records within the file. This information is used to determine the number of records that the file can hold, and thus the proper size for the index area.

2. PADDING=
   This optional keyword allows a percentage of increase in the size of the file's index area. This increased size allows more efficient access when the file nears a full condition.

3. /*
   This mandatory keyword is used to terminate input of control parameters, and must appear on a line by itself.

Each individual index record contains an argument as well as the necessary information for locating the associated prime data record. During RINT or ?RINT, these arguments are set to -1, indicating that no data record is associated with the index record.

CAUTIONS

1. The maximum number of arguments that may be used with a file initialized by RINT or ?RINT is 65,535.

ERROR MESSAGES

MISSING VALUE
PARAMETER ERROR
The RECSIZ parameter is negative, zero, or nonexistant. This parameter must be specified.

VALUE TOO LARGE
PARAMETER ERROR
The RECSIZ parameter is larger than 395 (a full sector).

DISC SPACE TOO SMALL
PARAMETER ERROR
Too large a padding was specified, resulting in an index area larger than the file as a whole.

PARAMETER ERROR
A keyword was misspelled, or the keyword /* was not on a separate line.

```
***************
*             *
*    SMCB     *
*             *
***************
```

Each channel of the asynchronous multiplexer has associated with it an 8-bit register known as a control byte which defines the format of the data transferred on that channel. A utility program called SMCB exists which can be used to set these control bytes. It resides in a file named XSMCB and is invoked by performing three JCL statements:

1. A .SETA statement which sets UPSI to the desired control value (see below),

2. A .USE statement which assigns SYS0 to the desired channel, and

3. A .RUN statement which names XSMCB as the file to be executed.

The desired control byte value to which UPSI is set is determined by the format which is to be used on the channel. The format is classified into four characteristics: data length, parity, number of stop bits transmitted, and baud rate. Several selections are available for each characteristic and each selection has a value associated with it. The control byte value is obtained by choosing one selection within each characteristic and adding up the values associated with the selections chosen.

The length characteristic defines the number of "data" bits per character, exclusive of start, parity, and stop bits. Four selections are available:

No. of Bits	Control Byte Value
5	0
6	2
7	4
8	6

The parity characteristic defines whether a parity bit exists and if so, how it is computed. Three selections are available:

Parity	Control Byte Value
odd	0
even	1
none	16

The number of stop bits transmitted characteristic defines the number of stop bits per transmitted character. It has no effect on received data. Two selections are available:

No. of Stop Bits	Control Byte Value
1	0
2	8

ASCII CRT's and teletypes require one stop bit. Baudot devices require two.

The baud rate charactistic defines the rate at which data is transferred into and out of the channel. Eight selections are available:

Baud Rate (Bits per Second)	Control Byte Value
programmable generator	0
75	32
300	64
1,200	96
2,400	128
4,800	160
9,600	192
19,200	224

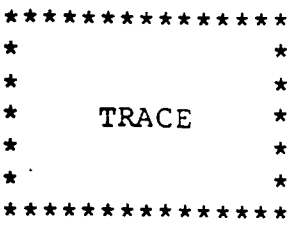
The 4-port interface must be modified if it is to operate at 19,200 baud.

The standard channel format is 7 bits, an even parity bit, and one stop bit, at 9,600 baud. This gives a control byte value of 4+1+0+192=197. If it were desired to set channel CRT2 to this format, it could be done with the following JCL statements:

```
.SETA UPSI=197
.USE CRT0 FOR SYS0
.RUN [S.]XSMCB
```

The following is a table of control byte values for common 8-bit formats:

Parity:	None	Even	Odd
Baud Rate:			
300	86	69	68
➤ 600	-	5	-
1200	118	101	100
2400	150	133	132
4800	182	165	164
9600	214	197	196



CPL TRACE

The CPL Trace Procedure, found in the 'P' library, displays program paragraph names, integer values and character strings during program execution.

TRACE USAGE

The insertion of a ;TRACEON anywhere in a CPL program after the ENTRY statement will cause any paragraph name encountered in the execution of the object program to be displayed on SYSLOG. The display terminates when a ;TRACEOFF is encountered in a source program. Multiple sets of ;TRACEON and ;TRACEOFF statements may be used. There must be at least one ;TRACEOFF statement or the program will not assemble.

If a ;TRACEON is placed immediately after the ENTRY statement and a ;TRACEOFF is placed immediately before the END statement, all paragraph names encountered in the logic of program execution will be displayed.

Assume the following logical flow:

```
;TRACEON
A10:
  . . .
  . . .
A11:
  Go to A15
  . . .
A12:
  . . .
;TRACEOFF
  . . .
A15:
  . . .
  . . .
  Go to A12
  . . .
```

Labels A10, A11, and A12 will display; A15 will not display.

Other items may be inserted into the logic which will cause data items to be displayed on SYSLOG while Trace is on.

;MARK4 (32-BIT-INTEGER-NAME) will cause the value of the integer to be displayed in the format of N-15,

;MARK6 (48-BIT-INTEGER-NAME) will cause the value of the integer to be displayed in the format of D-15,

;MARKS (STRING-NAME) will cause the value of the string to be displayed in the format of X10,C122.

TRACE IMPLEMENTATION

A special procedure is used to assemble a program containing a Trace:

P.TRACPL filename d lst [XREF][LIB]

The parameter LIB is entered if a source library is used (CPLC procedure is normally used). The jobstream will request up to two library names.

TRACPL will insert additional instructions into a copy of the source program (therefore, the Trace and Mark instructions do not have to be removed after testing is completed and the program is normally compiled) and then compiles the copy into an executive program.

After assembly is complete, the program may be executed normally. To display Trace information on the printer, assign SYSLOG to the printer before execution of the program.

USAGE NOTES

1. a. The additional coding will be inserted immediately after the ENTRY statement, before the END statement, after each paragraph name, ;MARK4, ;MARK6, and ;MARKS encountered between sets of ;TRACEON and ;TRACEOFF statements and before the END statement (see partial program listing).

   b. The Mark instructions will give a single value each time they are encountered in program logic. If placed immediately after the ENTRY and ;TRACEON statements and before any program label occurs, only the initial values of the specivied integers and strings will be displayed on SYSLOG. A Mark instruction placed inside a loop will display the

values of specified integers and strings with each pass through the loop.

c. Compilation of the program must be under EXPANSION A.

d. Called routines must not be traced if the routine is modified by assembly language linkage. The linkage will usually store data names at a paragraph name plus a displacement. Since TRACPL will display only the first 8 characters of a paragraph name.

2. While not generally known, paragraph names may be one of any length; TRACPL will display only the first 8 characters of a paragraph name.

3. TRACPL will cause additional memory to be used during testing:

16 Bytes per Traced Label
5 Bytes per Mark
87 Bytes Overhead per Program

Sample of Trace Usage and Code

```
entry
OPEN OUTPUT @LOG
;
a10:
;
;traceon
;
a20:
DIRECT
  JSR/@TRACE
  DW *+4
  JMP *+11
  DC 'A20      '
  DB 0
CPL
;
;mark4 (zero)
DIRECT
  JSR/ @MARK4
  DW ZERO
CPL
;
;mark6 (zero)
DIRECT
  JSR/ @MARK6
  DW ZERO
CPL
;
;marks (sval)
DIRECT
  JSR/ @MARKS
  DW SVAL
CPL
;
;traceoff
;
a30:
;
a40:
;
DIRECT
@MARK4 LDA= @FN15
  JMP @MARK@
@MARK6 LDA= @FD15
  JMP @MARK@
@MARKS LDA= @FC122
  JMP @MARK@
@TRACE LDA= @F90
```

Sample Trace Usage and Code (cont'd)

```
@MARK@ STA *+11
  LDA- X+
  STA *+9
  JSR- Z
  DB @WF
  DW @LOG
  DW 0
  DW 0
  DW 0
  RSR
CPL
FORMAT @FN15:N-15
FORMAT @FD15:D-15
FORMAT @FC122:X10,C122
FILE @LOG: SYSLOG, CLASS=1
END
```

```
***************
*             *
*    TRACT    *
*             *
***************
```

PURPOSE

Creation of outboard list files for sorted access of VSI files is done by the program XTRACT. This program may be used to create either type 'B' or 'C' list files. XTRACT will construct the list record in the format specified by the programmer, and may include data extracted from the master-file prime data record.

The format of the list record is controlled by parameter statements entered via SYSRDR. The following examples indicate the basic form of the parameters.

EXAMPLE 1

```
.USE XX#0A ON #1 FOR SYS0
.USE XX#0L ON #1 FOR SYS1
.RUN S.XTRACT
FILTYP=B,KEY,DATA=(36,4),ARG,DATA=(5,31)
/*
.SKIP TO ERR IF 0 NE #C
```

EXAMPLE 2

```
.USE YY#0A ON #1 FOR SYS0
.USE YY#0L ON #1 FOR SYS1
.RUN S.XTRACT
PRINT=YES FILTYP=C KEY ARG /*
.SKIP TO ERR IF 0 NE #C
```

EXAMPLE 3

```
.USE ZZ#0A ON #1 FOR SYS0
.USE ZZ#0L ON #1 FOR SYS1
.RUN S.XTRACT
PRINT=ALL,KEY,ARG,DATA=(5,21,265,4),/*
.SKIP TO ERR IF 0 NE #C
```

The control parameters are entered as a series of keywords separated by either a space or a comma. Input of control parameters is terminated by the keyword /*.

*** NOTE ***

The use of control parameters to establish the list record format is mandatory. There is no default record format in the absence of parameter input.

The parameter keywords are:

1. PRINT=
The PRINT= parameter controls the output of diagnostic messages to the device/file assigned to SYSIPT. The effect of the print parameter is:

PRINT=NO	Print error messages only
PRINT=YES	Print parameter control statements
PRINT=ALL	Print parameter control statements and the first two fields of each list record created. The two fields printed will be in the format N-10, N-15 (if the argument is 4-byte integer), in the format N-10, D-20 (if the argument is a 6-byte integer), or in the format N-10, X2, C67 (if the argument is a character string).

If none of the PRINT= keywords are used, the extraction program will assume the PRINT=NO condition.

2. FILTYP=
The FILTYP= parameter specifies the type of disk file to be used for output from the extraction program. The options available are:

FILTYP=B	Output to binary file.
FILTYP=C	Output to contiguous file with record zero containing the number of records in the first four bytes.

The extraction program will assume a binary output file if the FILTYP= keyword is omitted.

3. ARG
The ARG keyword is used to specify the position of the master-record argument in the list record.

4. KEY
The KEY keyword is used to specify the position of the master-record relative-key in the list record.

5. DATA=
The DATA= keyword is used to specify the location and length of data in the prime data record which is to be placed in the list record. The construction of the DATA= keyword is:

DATA=(xx,yy,xx,yy)

where xx is the beginning byte of the field in the

prime-data record and yy is the field-length in bytes.

The data fields must be enclosed in parentheses.

6. /*
The slash-asterisk keyword is used to terminate input of control parameters.

The examples shown above will be interpreted as follows:

EXAMPLE 1

(Assume that file XX#0A uses 4-byte integer arguments)

No PRINT= key word is used, so the default PRINT=NO will be observed.

The output file-type has been specified as a binary file.

The format of the list record will be:

```
RECORD LSTREC (43)
INTEGER RELKEY ; Master-file key
INTEGER FLD01  ; Prime data bytes 36 thru 39
INTEGER FLD02  ; 4-byte argument of prime record
STRING FLD03 (30) ; Prime data bytes 5 thru 35
ENDREC
```

EXAMPLE 2

(Assume that file YY#0A uses 6-byte integer arguments)

The PRINT=YES keyword will cause the line of parameter input to be echoed.

The output file-type has been specified as a contiguous file.

The format of the list records will be:

```
RECORD LSTHDR (10)
INTEGER POINT  ; Pointer to last record in list file
ENDREC
;
RECORD LSTREC (10)
INTEGER RELKEY ; Relative key of prime record
INTEGER ?FLD01 ; 6-byte argument of prime record
ENDREC
```

EXAMPLE 3

(Assume that file ZZ#0A uses 10-byte character-string arguments)

The PRINT=ALL keyword will cause the line of parameter input and the key and character-argument of each record to be echoed.

The output file-type has not been specified, so a binary file is assumed.

The format of the list record will be:

```
RECORD LSTREC (40)
INTEGER RELKEY ; Relative key of prime record
STRING FLD01 (10) ; 10-byte argument of prime record
STRING FLD02 (20) ; Prime data bytes 5 thru 25
INTEGER FLD03  ; Prime data bytes 265 thru 268
ENDREC
```

ERROR MESSAGES

The following errors are considered fatal. The error message will be output to SYSLOG and the program will terminate immediately with the indicated completion code.

PARAMETER ERROR (CC = 100)

A parameter error may be caused by:

1. Imbedded blank(s) in parameter keyword.
2. "Data=" keyword contains improper field definition.
3. "Data=" field is not within prime record.
4. List record contains more than 50 fields.
5. List record is more than 393 bytes long.

ERROR - OUTPUT FILE TOO SMALL (CC = 111)

Self-explanatory.

The following error is considered non-fatal. The error message will output to SYSLOG, but the program will continue. If no fatal error occurs, when the program finishes, it will terminate with a completion code of 1 (one).

ILLEGAL KEY OMITTED: xx yy

The relative key (xx) of the record in the input file for argument yy is not within the bounds of the file. Data from the record cannot be extracted.

```
***************
*             *
*    VCOPY    *
*             *
***************
```

PURPOSE

VCOPY is a general-purpose program designed to copy any VSI-master file into another file which has been initialized by VRINT. VCOPY is capable of copying any file, regardless of record size or key length, thus eliminating the need for custom copy programs in an application.

PROCEDURE

The following jobstream illustrates the set-up for a VSI-file copy:

```
.NEW file-2 ON d2 'I' size
RECSIZ=nn,KEYLEN=nn,PADDNG=nn
/*
.SETA UPSI=0
.USE file-1 ON d1 FOR SYS0
.USE file-2 ON d2 FOR SYS1
.RUN S.XVCOPY
.SKIP TO ERR IF 0 NE #C
```

where

file-1 is the name of the file to be copied,

d1 is the number of the disk on which the source file is located,

t is the size of the new file in tracks, equal to the size of the source file,

file-2 is the name of the target file, and

d2 is the number of the disk on which the target file is located.

CAUTIONS

1. The target file must be both empty and initialized by VRINT. If the target file contains data, the copy may not produce the desired results. If the target file was not initialized by VRINT, the copy will be terminated.

2. The target file must have been initialized with the same record size and key length as the source file. If the record size or the key length is different, the copy will be terminated. The padding in the two files may vary, as long as a conflict will not be caused by the difference in area available for prime-data records.

3. If UPSI is set to a non-zero value, the relative key and argument of each record will be output to SYSLOG before the record is copied to the target file. If UPSI is set to zero, the copy will be made without record counts being displayed.

ERROR MESSAGES

The following errors are considered fatal. The error message will be output to SYSLOG and the program will terminate immediately with the indicated completion code.

ERROR - FILE NOT INITIALIZED (CC = 100)

Either the input or output file has not been initialized by XVRINT.

ERROR - RECORD SIZES DO NOT MATCH (CC = 100)

The target file was initialized with a record size different from the source file.

ERROR - KEY LENGTHS DO NOT MATCH (CC = 100)

The target file was initialized with a keylength different from the source file.

ERROR - OUTPUT FILE TOO SMALL (CC = 111)

Self-explanatory.

The following errors are considered non-fatal. The error message will output to SYSLOG, but the program will continue. If no fatal error occurs, when the program finishes, it will terminate with a

completion code of 1 (one).

ILLEGAL KEY OMITTED: xx yy

The relative key (xx) of the record in the input file for argument yy is not within the bounds of the file. The record cannot be copied.

DUPLICATE ARGUMENT OMITTED: xx yy

A record with argument yy already exists in the output file. The input record whose relative key is xx will not be copied.

****
*   *
* WTAG/ *
* ?WTAG *
*   *
****

PURPOSE

WTAG and ?WTAG are used to create list files for type 'C' initialized files. These binary list files, containing the argument and relative key for each prime data record, may then be sorted in order of the arguments. WTAG creates files with four-byte arguments; ?WTAG creates files with six-byte arguments.

PROCEDURE

The indexed file to be read is assigned to SYS0, and the 'B' type list file is assigned to SYS1. When [S.]XWTAG or [S.]X?WTAG is run, the index area of the 'C' type file is read, and all arguments other than -1 are copied, with their relative keys, into the list file. An argument of -1 indicates that no prime data record is asscciated with the argument, and these arguments are not copied.

CAUTIONS

1. While negative arguments (other than -1) may be entered into a file, and will be copied by WTAG or ?WTAG, the record associated with the argument cannot be accessed by the GETKEY or ?GKEY subroutines.

ERROR MESSAGES

********** I/O ERROR **********

The operating system was unable to read the indexed file or write to the list file during the normal operation of the program.

WXIBM

IBM FORMAT DISKETTE UTILITY

CPU 6

February 29, 1980

Warrex Computer Corporation

Richardson, Texas

Printed in USA

****
* WXIBM *
****

Purpose

This utility provides an interface between IBM format data and Centurion formatted data. The utility is used to copy data from Centurion 'A' type file to IBM format files or vice versa. It is also used for directory maintenance and diskette formatting in IBM format.

Note: This utility will not work with the Finch/Floppy Controller.

The system logical units must be assigned in the following manner:

SYSRDR	command input (device independent)
SYSIPT	alternate for SYSRDR (see below)
SYSLOG	all log output (device independent)
SYSØØØ	disk assignments
SYSØØ1-SYSnnn	optional 'A' type file for copying to or from IBM format

The command input unit is determined by the value of UPSI. If UPSI is zero (Ø), SYSRDR is used; if UPSI is one (1), SYSIPT is used. The disk unit or units to contain the IBM format diskette(s) must be assigned to the same SYS number as the disk unit number. For example:

```
.USE DISK4 FOR SYS4
.USE DISK7 FOR SYS7
```

Any logical units (SYSnnn) not used for disk assignment may be used to assign 'A' type files for copying to or from the IBM format diskette(s).

Once the utility is loaded (.RUN XWXIBM), a sign-on message is displayed, and the command input device is read (with parameter substitution) for control statements as follows:

1. COPY

   There are two forms of the COPY statement: one to copy Centurion to IBM format:

   ```
   COPY SYSn to 'file' [ON] d [UC]
   ```

   and one to copy from IBM to Centurion format:

   ```
   COPY 'IBM file' [ON] d to SYSn [UC]
   ```

   The utility searches the IBM format diskette for the specified file. The name of the Centurion format file assigned to the logical unit in the copy statement (SYSn) is displayed along with the logical diskette.

drive name. The beginning and ending sector addresses of the IBM format file are also displayed. Upon completion of the copy sequence, the utility displays the number of data records transferred.

When copying from Centurion to IBM format (first example), the data is converted from ASCII to EBCDIC and written to the IBM file, one record per sector. As the IBM sector is 128 bytes in length, records shorter than 128 bytes are padded with trailing zeros. Records longer than 128 bytes will be truncated, and an error message will be displayed. The occurance of a truncated record does not terminate the copy. The "UC" option, if stated, will force an upper-case conversion, of alpha characters, only during the copy process.

Likewise, when copying from IBM format to Centurion format (second example), the IBM format file is read and trailing zeros are removed. Data is then converted to ASCII and written to the Centurion file, one Centurion record per IBM sector.

2. FORMAT

Enter:

```
FORMAT d VOL='volume'
```

where:

d — is the number of the diskette to be formatted

volume — is the 10-character volume name enclosed in single quotes.

The diskette is formatted in IBM format. The sectors are 128 bytes in length (plus 1 control byte), numbered 1 through 26. There are 77 tracks per diskette, numbered 0 through 76. After the diskette is formatted, the utility clears the directory.

3. CLR

Enter:

```
CLR d VOL='volume'
```

where:

d — is the number of the diskette to be cleared

volume — is the 10-character volume name to be assigned to the diskette

The directory of the diskette is set up with one active file, 'DATA', with beginning of extent (BOE) of 01001 and end of extent (EOE) of 73026, and inactive files 'DATA09' through 'DATA26' numbered sequentially. Each of these files has a BOE of 74001 and EOE of 73026.

4. VOL

Enter:

VOL d 'volume'

where:

d is the disk number

volume is the 10-character volume name enclosed in single quotes.

This command is used to assign a volume name to an IBM format diskette.

5. DIR

Enter:

DIR d

where:

d is the number of the diskette whose directory is to be displayed

The directory listing consists of the entry number, volume name, data set label, beginning of extent, end of extent, end of data, record length and active status.

6. EXTENT

Enter:

EXTENT n [ON] d [LABEL='label'][BOE=nnnnn][EOE=nnnnn] [EOD=nnnnn] [RECLEN=nnnnn] [ACTIVE=Y/N]

where:

n is the entry number of an entry in the directory listing

d is the disk number

label is a 9-character data set name enclosed in quotes

nnnnn is a 5-digit number with leading zeros

This control statement enables the user to perform file allocation functions for an IBM format diskette by changing entries in the directory of that diskette.

The directory display (see DIR) for each diskette contains 19 numbered entries (1-19). These numbers are used to identify the entry to be changed.

The keywords may be used in any order and not all keywords need appear in the statement. The keyword LABEL is used to change the name associated with a data set (file) and must be enclosed in quotes. The keyword BOE alters the beginning of extent field in the entry. The value following this keyword must be a 5-digit disk address. The first two digits of the disk address are the track number (00-76), the third digit is a zero, and the last two digits are the sector number (01-26). The keywords EOE and EOD are used to alter the end of extent and end of data fields, respectively. They are used in the same manner as the BOE keyword. The keyword RECLEN is used to alter the record length field. This keyword must be followed by a 5-digit record length with leading zeros. The keyword ACTIVE must be followed by either a 'Y' or 'N' to signify whether or not the file is active.

The EXTENT statement reads in the proper directory sector as designated by the entry number. Changes re made; if no errors are found, the directory sector will be written back to the diskette.

7. END

This command terminates the utility.

Cautions

SYS000 must be unassigned when using this utility as it is the program logical unit for the IBM format floppy.

Due to the allocation process and the fact that the utility cannot determine if the same IBM format file is in use by another partition, the user must insure exclusive use of the IBM file. If this condition is not met, data loss and directory errors are likely to occur.

The SYS number specified in the control statement must refer to an unassigned logical unit, otherwise an ABORT-10 will occur.

Error Messages

INPUT RECORD TRUNCATED

This error occurs when a record larger than 128 bytes is encountered while copying from Centurion format to IBM format.

CONTROL STATEMENT ERROR

This error occurs when the control statement cannot be processed as entered.

ILLEGAL PHYSICAL UNIT

This error occurs when the disk number entered in a control statement is invalid.

EOM ON OUTPUT

This error occurs when attempting to write to an IBM format file past the end of the file.

DATA SET NOT FOUND

The IBM format file entered in the control statement was not located in the directory.

DUPE DATA SET NAME

This error occurs when attempting to assign a pre-existing name to an entry in the directory via the extent command.

```
**********
*        *
*  @REORG *
*        *
**********
```

PURPOSE

@REORG is a stand-alone disk dump-reorganization utility used for backing up disks. It can only be accessed by IPL loader at system boot-up time. This utility should only be used by experienced personnel, for there are no safeguards to stop one from reorging over a system or default disk.

PROCEDURE

1. Depress SELECT button

2. CRT may display: (if not go to step 3)

   D=

   ENTER - system disk #

   Example: H# (hard disk #)
   F# (floppy disk #)

   CRT then displays:

   LOS 7.XX

3. NAME =

   ENTER - @REORG

4. DISK =

   ENTER - disk number which contains the @REORG utility file.

   NOTE: The disk referencing for @REORG is:
   Ø-5 on-line Hawk/Falcon/ Pertec disk (6)
   6-23 on-line CMD disk (18)
   24-25 on-line Floppy disk (2)

5. CODE =

   ENTER - disk code

   Newline if set to default

6. CRT displays:

   @REORG 7.XX

7. CRT may display: (if not go to step 8)

   HOW MANY CMD PLATTERS?

   ENTER - number of CMD platters on this drive.

This message will only display if a CMD drive is detected to be on-line. The number of platters must be entered to prevent @REORG from accessing non-existant platters (on-line test) which causes the drive to go into a "fault" condition. The only way to reset this condition is to re-power the drive.

8. CRT will display system status using @REORG's configuration.

```
#          VOL
0      CCCCCCCCCC  NN/NN/NN
```

   And the prompt

   Enter source disk

   Enter

   Disk number to be reorganized

9. CRT displays

   Enter target disk

   Enter

   Disk number to receive the reorganized files

10. CRT displays

    Do you want to check volumes dates? (Y/N)

    Enter

    Y, if you want to assure that the date of the source disk is later than the date of the target disk.
    N, if you do not want to make this check.

11. CRT displays

    Do you want to check volume labels? (Y/N)

    Enter

    Y, if you want to assure that the volume name of the target disk is the same as the source disk.
    N, if you do not want to make this check.

12. CRT displays

    Do you want any special options? (Y/N)

    Enter

    Y, if you want any special options. Go to step 13.
    N, if you do not want any special options. Go to step 16.

13. CRT displays

    Enter options:

    Enter one or more of the following commands, separting multiple entries with commas

    CONDENSE=N
    SELECT=Y
    FSIOPT=Y
    DSIZE=NS OR NT, WHERE N IS NUMBER OF SECTORS OR TRACKS
    USIZE=NS OR NT, WHERE N IS NUMBER OF SECTORS OR TRACKS
    FSIZE=NS OR NT, WHERE N IS NUMBER OF SECTORS OR TRACKS
    SCODE=NNNNN, WHERE NNNNN IS SECURITY CODE OF SOURCE DISK
    TCODE=NNNNN, WHERE NNNNN IS SECURITY CODE OF TARGET DISK

14. CRT may display: If not go to step 15.

    Option statement error

    Return to step 13.

15. CRT displays the system status including the source (S) and target (T) disks and the prompt

    Ready? (Y/N)

    Enter

    Y, if your options are correct and you are ready to execute the REORG. Go to step 16.
    N, if you are not ready to execute the REORG. Return to step 8.

16. CRT may display: If not, go to step 17.

    Output disk newer than input disk. Return to step 8.

    or

    Volume names not equal. Return to step 8.

17. CRT may display: If not, go to step 18.

    Output disk protected. NOGUARD? (Y/N)

    Enter

    Y, if your options are correct and you want to proceed with the REORG. Go to step 18.
    N, if you are not ready to execute the REORG. Return to step 8.

18. If the SELECT=Y option has been chosen, CRT will display files one at a time, with the prompt

    ?

    Enter

    Y, if you want to copy the file
    N, if you do want want to copy the file

19. CRT displays the files as they are copied and the prompt

    Do you want to guard the target disk? (Y/N)

    Enter

    Y, if you want to guard the target disk
    N, if you do want to noguard the target disk

20. CRT displays

    Do you want to noguard the source disk?

    Enter

    Y, if you want to noguard the source disk
    N, if you want to guard to source disk

21. CRT displays

    Ready to load operating system? (Y/N)

    Enter

    Y, if you have completed your REORGS. Go to step 22.
    N, if you want to REORG other disks. Return to step 8.

22. CRT displays usual prompts to load operating system

```
**********
*        *
*  SELREST  *
*        *
**********
```

PURPOSE

The S.SELREST program is used to selectively restore files that were created with ".BACKUP". The utility requires a file from the source disk (@SINDO) and will attempt to ".RESTORE" that file first. This utility has the ability to accept file names from either an 'A' type file or the CRT.

PREPARATION/REQUIREMENTS

1. The disk to be .RESTORED from must have been created using .BACKUP.

2. The "S" library must be on disk

3. The target disk must have enough room to hold the ".RESTORED" file. See Note 2.

PROCEDURE

1. Enter:

   S.SELREST

2. The CRT will display:

   ENTER SOURCE DISK

   Enter the number of the source disk.

3. The CRT will display:

   ENTER THE TARGET DISK

   Enter the number of the target disk.

4. The CRT will display:

   SELECTIVE RESTORE REQUIRES THE INDEX FILE (@SINDO) TO BE RESTORED FIRST.

   INSERT DISK CONTAINING THE INDEX FILE (NORMALLY THE LAST DISK).

   NEWLINE TO CONTINUE.

   Insert the correct disk, press newline.

5. The index file (@SINDO) will be restored

   a. If wrong disk was inserted, the CRT will display:

      INDEX FILE "@SINDO" IS NOT ON THE DISK

      NEWLINE TO CONTINUE

      when the condition causing the error has been corrected, press newline.

   b. If the index file cannot be restored, an error message will be displayed, followed by ".use" CRT#V SYSDR". At this pint, the data on this backup is not able to be selectively restored.

6. The CRT will display:

   DO YOU WANT TO ENTER FILE NAMES FROM THE CRT? (Y/N OR END TO EXIT)

   Enter one of the following:

      "END" if you want to stop without restoring any data files.

      "Y" if you want to enter file names from the CRT. Go to step 7.

      "N" if you want the file names to come from an 'A' type file. Go to step 9.

7. The CRT screen will be cleared and the program will sign on the screen, followed by the prompt:

   ENTER FILE NAMES

8. Enter the file name to be restored, in any sequence, separated by spaces or commas or one file per line. The last entry must be "/*". Go to step 12.

9. The CRT will display:

   ENTER INPUT FILE NAME

   Enter the 'A' type file name, that contains the files to be restored, in the format described in step 8.

10. The CRT will display:

    ENTER DISK NUMBER FOR "FILE NAME"

    Enter the disk number where the file resides.

11. The CRT will display:

    "/ERROR! FILE "FILE NAME" IS NOT ON DISK.

    Correct the error condition and return to step 9.

12. At this point, selectively restore has created a JCL in "@SCR#I" that will restore the requested file names.

13. The data files will be restored one at a time from the source disk to the target disk as entered in steps "2" and "3".

During execution of restore for each individual file, you may see the following:

```
WARNING !!!!
FILE "FILENAME" HAS NOT BEEN RESTORED FORM "DISK"
NEWLINE TO CONTINUE
```

See Note 1 below

Press Newline after noting the file name.

NOTES:

1) If the file did not exist on the target disk, the temporary file will be deleted. However, if the file existed, and was renamed when the backup was attempted, the point of failure, the incomplete file was deleted and the original file was re-instated.

2) If a file exists on the target disk, there must be enough room to maintain two copies of the file during the restore.

XMIT/RECV

Transmit/Receive Communications Utility

CPU-6

May, 1982

```
***************
*             *
*  XMIT/RECV  *
*             *
***************
```

XMIT/RECV UTILITIES
(Software Available for Purchase)

PURPOSE

The Centurion XMIT/RECV utilities allow you to transfer data directly between Centurion VI computers over phone lines via modems.

The XMIT/RECV utilities provide you with the following capabilities:

1. File-to-file data transmission;
2. File-to-device data transmission;
3. Error detection and recovery;
4. Optional operator control for the RECV utility;
5. Application program interfaces in APLIB for communications.

DETAILED OVERVIEW

The file transmission utilities XMIT and RECV allow the user to transmit any type file from one Centurion 6000 system which is equipped with an asynchronous telecommunications interface to either a receiving file of the same type or an output device on another Centurion 6000. This is accomplished via dialup or leased telephone lines. The data transfer baud rates used can be up to 1200 baud asynchronous.

The transmitting system must have the program XXMIT on disk with at least 8K of partition memory available. XMIT is designed to run in a interactive or blind partition and is called up by the XMIT and XMITB jobstreams respectively.

The receiving system must have the program XRECV on disk; it also requires 8K of partition memory. RECV is also designed to run in an interactive or blind partition and is called up by the RCVB jobstream. It can control up to four (4) asynchronous ports and can receive up to four (4) files simultaneously.

XMIT UTILITY

The XMIT utility allows you to send data from one Centurion 6000 series system to another over telephone lines. The receiving system must contain the XRECV program which allows that system to receive the information. Both the XMIT and the RECV utilities are designed to operate in blind or interactive partitions.

XMIT INITIALIZATION

General Statements

1. All messages will be displayed at SYSLST and SYSLOG.
2. The RECV utility must have been started on the receiving system before you initialize XMIT.
3. All invalid input is followed by the related error message and the ILLEGAL PARAMETER message.

Procedures

1. The information you enter to bring up the XMIT utility can vary according to the information you need to transfer. The following are examples of the information you can enter to bring up the XMIT utility:

   NOTE: Information in brackets indicates optional data.

   [.START Pn] XMIT input d1 cline ouput ON d2 [,CODE = OCCC]

   [.START Pn] XMIT input d1 cline device [,CODE = OCCC]

   Where:

   START = JCL command to initialize the XMIT utility (used only when running XMIT in blind mode);
   Pn = partition number of the partition in which XMIT is to operate (used only when running XMIT in blind mode);
   XMIT = JCL that calls the XMIT utility;
   input = file to be transmitted;
   d1 = disk number where the file is located;
   cline = device name for the asynchronous communications port;
   ouput = destination filename or device (must exist on destination system);
   d2 = destination disk;
   ,CODE = OCCC = optional security code for the receiving asynchronous port.

   Note : 'ON d2' is not used when outputting to a device.

2. After you initialize XMIT by entering the correct information detailed above, XMIT automatically issues an XMIT start request to the receiving system. This request is asking the RECV utility if it will accept data transfer. If RECV is operating in a blind partition, the RECV utility automatically replies yes or no depending on whether the information entered in step 1 of this section is valid (e.g. you entered an existing file, the proper disk number, the file is the proper type, etc.). If RECV is operating in an interactive environment, the operator can make one of the following entries through the RECV system:

   YES = Causes the XMIT program to begin transmitting sector-by-sector the specified input file to the specified output file or device;

   NO = Causes the XMIT program to abort. The system then displays this message:

```
mm/dd/yy hhmmss SEC 0000 - OPERATOR REJECTED
XMIT ABORT
hhmmss END XMIT CC = 4
```

   After initialization and startup, there are only three (3) ways of exiting XMIT:

   1. After successful information transfer, the system automatically terminates XMIT;
   2. RECV stops transmission through either voluntary operator action or automatic error termination;
   3. XMIT automatically terminates itself when an error is detected.

   The XMIT system automatically displays an abort message when exiting XMIT indicating why the transmission was stopped and detailing the completion code. Examples of the messages that can display are given below:

   Termination from error condition:

```
mm/dd/yy hhmmss SEC 00000 - NO RESPONSE
XMIT ABORT
hhmmss END XMIT CC = 6
```

   Termination at end of successful transmission:

```
mm/dd/yy hhmmss SEC ## - EOF
## SECTORS TRANSMITTED, ## RECEIVED
hhmmss / END XMIT CC =0
```

Refer to the XMIT message section for message descriptions.

RECV UTILITY

The RECV utility allows you to directly communicate data over dialup or leased telephone lines from one Centurion 6000 system to another. Information transmitted must be sent through the XMIT utility programs which must be on the sending Centurion system. A system of checks is continuously in operation during data transmission ensure data integrity. If any problems occur during data transmission, the system displays the appropriate error message to notify you of the problem.

RECV INITIALIZATION

General Statements

1. All valid input is acknowledged by COMMAND ACCEPTED.

2. If the system detects an error condition, it will attempt to continue operation 14 times. If, after the fourteenth attempt, the error condition still exists, the system will display the appropriate message and automatically terminate XMIT/RECV. To continue operation, you must restart both XMIT and RECV.

Procedures

1. The JCL you need to enter to bring up the RECV utility varies depending on the specific information transfer transaction involved. The following factors can vary:

    1. Whether RECV is operating in a blind or interactive mode;
    2. Number of communications lines;
    3. Line names;
    4. Device assigned as SYSLOG;
    5. JCL name for the RECV utility.

    Because of these variable factors, the method you use to bring up RECV is determined at installation. Commands used to bring up RECV should be written in the box outlined on the next page for easy reference. Below is an example of what you might enter:

EXAMPLE: (Assume LINE0 is the SYSGENED communications line.)

```
.STARTPn  RECV jobstream
.USE LINE0 FOR SYS4
.USE DUMMY FOR SYS5
.USE DUMMY FOR SYS6
.USE DUMMY FOR SYS7
.USE PRTx FOR SYSLOG
.MULTP
.RUN XRECV ON #5
.END
S.CON
.START Pn   (Optional if running in blind)
```

Where:

n = Partition number (If you are running in blind)
x = Printer number;
XRECV = Executable receive program.

2. CRT displays:

   ENTER COMMAND

   At this time, the asynchronous port(s) named device must be assigned to PORTn. To assign the port(s), enter the following:

   USE device FOR n

   Where:

   device = asynchronous port name;
   n = communications port 1 through 4.

   Go to step 3.

3. Once PORTn has been assigned, it must be started to allow data reception. To start the port, enter the following:

   START n

   Where:

   n = port number 1 through 4.

   Go to step 4.

4. CRT displays:

   HHMMSS - PORTn - STARTED

   The assigned PORTn is initialized; it can now receive the XMIT start request. During the XMIT start request, XMIT is asking the RECV utility if it will accept data transfer.

If RECV detects an error in the XMIT start request, it will display the appropriate message and automatically terminate the program. It will also send the error status information to XMIT. The following messages can display if an error is detected during the XMIT start request:

```
hhmmss - PORTn - PROTOCOL ERROR - START
hhmmss - PORTn - filename ON d - DOES NOT EXIST - REJECTED
hhmmss - PORTn - device - DOES NOT EXIST - REJECTED
hhmmss - PORTn - INCORRECT FILE TYPE
hhmmss - PORTn - SECURITY CODE VIOLATION
```

Refer to the RECV message section (IV-69) for specific descriptions of all messages.

If there are no problems with the XMIT start request and the system is operating in a blind partition, the RECV system automatically accepts information transfer. If there are no problems with the XMIT start request and the system is operating in an interactive mode, the system displays one of the following two messages depending on whether the start request is to a filename or a device:

```
REQUEST TO filename ON d - ACCEPT?

REQUEST TO device - ACCEPT?
```

Where:

```
filename = file to receive data;
       d = disk on which the file is located;
  device = device to receive 'A' type file data only.
```

The operator can then make one of the following entries:

```
 NO = Causes the XMIT program to abort.
      The system then displays the
      following message at the sending
      system and the receiving system
      respectively:

mm/dd/yy hhmmss SEC 0000 - OPERATOR REJECTED
XMIT ABORT
hhmmss END XMIT CC = 4

hhmmss - PORTn - STOPPED
hhmmss - PORTn - ##### SECTORS TRANSMITTED, ##### SECTORS RECEIVED
```

YES = Causes the XMIT program to begin transmitting sector-by-sector the specified input file to the specified output file or device;

If the data is successfully transmitted to RECV, the RECV utility displays the following message on the terminal:

```
hhmmss - PORTn - END OF DATA RECEIVED
hhmmss - PORTn - STOPPED
hhmmss - PORTn - ##### SECTORS TRANSMITTED, ##### SECTORS RECEIVED
```

COMMANDS

The operator can utilize the following commands to manipulate the system utility operation:

1. STOP;
2. USE;
3. START;
4. CANCEL (port);
5. STATUS.
6. CANCEL (RECV).

1. STOP

Purpose:

Allows you to stop an activity or communications line (port).

Procedure:

Enter the following:

STOP n

Where:

n = line (port) number

NOTE: If the port is active, the system will not terminate the port while it is performing a task. Instead, it will wait for the task to be completed before it stops the line. While it is waiting for the task to be completed, it displays the following message:

hhmmss - PORTn - ACTIVE

When the task has been completed, the system then displays the following message:

hhmmss - PORTn - STOPPED
hhmmss - PORTn - ##### SECTORS TRANSMITTED, ##### SECTORS RECEIVED

2. USE

Purpose:

Allows you to assign a port.

Procedure:

Enter the following:

USE device FOR n

Where:

device = asynchronous port name;
n = communications port 1 through 4.

3. START

Purpose:

Allows you to start a port.

Procedure:

Enter the following:

START n

Where:

n = port number 1 through 4.

4. CANCEL (port)

Purpose:

Allows you to cancel (unassign) a port.

Procedure:

Enter the following:
CANCEL n

Where:

n = the specific port to be canceled

NOTE: A port must be stopped with a STOP command before it can be cancelled with a CANCEL command.

5. STATUS

Purpose:

Allows you to determine the status of all ports.

Procedure:

Enter the following:

STATUS n (used for one port)

STATUS ALL (used for all ports)

After you enter the valid STATUS command, the system displays the status information in the following format:

	DEVICE	STA	OUTPUT/SECTORS
hhmmss - PORTn	device	R	filename ON d
hhmmss - PORTn	device	R	odevice
hhmmss - PORTn	device	S	
hhmmss - PORTn	device	C	

Where:

device = device name of asynchronous port or blank if cancelled;
odevice = output device name (printer, CRT);
filename = receiving filename;
d = disk on which receiving file is located

The status codes that can be listed in the STA column are as follows:

A = port transferring data from line buffer to RECV buffer;
R = port is reading or awaiting line buffer data;
RT = port is attempting transmission retries (up to M);
S = port is stopped, but still assigned;
C = port is stopped and is unassigned;
E = error in status was detected (unknown).

6. CANCEL (RECV)

Purpose:

Allows you to cancel RECV to return control to console when RECV is operating in an interactive environment.

Procedure:

Enter the following:

CANCEL RECV

The system stops all active ports and sends the following message to the stopped ports:

hhmmss - PORTn - STOPPED
hhmmss - PORTn - ##### SECTORS TRANSMITTED, ##### SECTORS RECEIVED

The system also sends a stop transmission status message to XMIT.

RECV TERMINATION ON ERROR

After initialization and during normal data transfer, the program checks for errors. If an error is detected, the RECV program will issue a reverse interrupt (RVI) back to XMIT. When the RECV program issues the RVI, it also does the following:

1. Indicates the specific error;
2. Displays the error message;
3. Terminates related port data reception

OPERATOR MESSAGES

XMIT/RECV messages are divided into two separate operator message sections. This allows you to more easily refer to system messages. The messages are listed in alphabetical order and are followed by specific cause/corrective action information.

If the system detects an error condition, it will automatically try 14 times to continue operation. If after the fourteenth attempt, the error condition still exists, the system displays the appropriate message and automatically terminates XMIT/RECV. To continue operation, you must restart both XMIT and RECV.

XMIT MESSAGES

The following XMIT messages are listed in alphabetical order for easy reference.

1. BUFFER FULL

   The data sent to the receiving system did not match the data length specified. The completion code is set to 21.

2. CHECK SUM ERROR

   The check sum for the data sent to the receiving system does not match the check sum for the data transmitted. This means that data was lost during transmission. The completion code is set to 20.

3. EOF

   The end-of-file has been detected and has been sent to RECV. This indicates that data file transmission is complete. The completion code is set to zero.

4. FILE ASSIGNED

   The specified output file or device on the receiving computer is assigned. The completion code is set to 10.

5. FILE TYPES NOT MATCH

   The input file on the transmitting computer is of a different type than the output file or device on the receiving computer. Only "A" type files may be transmitted to a receiving device. The completion code set to 12.

6. ILLEGAL PARAMETER

   Input from the operator was either invalid or unrecognizable. You must make another entry.

7. LINE BUSY

The communications port is currently active and cannot be accessed at this time. The completion code is set to 22.

8. LINE INACTIVE

The communications port is inactive. The completion code is set to 23.

9. LINE NOT OPENED

The communications port is not open at the time a READ or WRITE was issued by the program. XMIT aborts with the completion code set to 8.

10. NO MEMORY

There is not enough partition memory available to generate another buffer. The partition is at or close to the 32K maximum. The completion code is set to 18.

11. NO RESPONSE

The receiving computer does not respond to the request. This may be due to a hardware or line problem; it could also be caused because RECV is not operational on the receiving computer. The completion code is set to 6.

12. OPERATOR REJECTED

The operator at the receiving computer either rejected the transmission request or stopped reception before the transmission was complete. The completion code is set to 4.

13. OUTPUT FILE NOT EXIST

The file or device to which the data is to be transmitted does not exist at the receiving end. The completion code is set to 10.

14. OUTPUT TIMEOUT

Output cannot be performed due to a hardware problem. The completion code is set to 7.

15. READ FILE ERROR

An error was detected on reading a file sector of the file to be transmitted. The completion code is set to 9.

16. RECEIVE TIMEOUT

RECV did not complete data reception already in progress. The completion code is set to 25.

17. SECURITY CODE VIOLATION

The specified security code sent to the receiving system is incorrect. All other messages indicate an internal RECV problem. The completion code is set to 13.

18. TRANSMISSION STOPPED - RECEIVE ERROR

XMIT was stopped due to a hardware or system error detected by RECV. The completion code is set to 24.

19. XMIT PAUSED

A HOLD command was received from the receiving system. Transmission cannot resume until the receive system receives a RESUME command.

20. XMIT RESUMED

The output is resumed by RECV from the PAUSE mode.

RECV MESSAGES

The following RECV messages are listed in alphabetical order for easy reference.

1. COMMAND ACCEPTED

   Operator input has been found to be valid and the system will receive specific command function.

2. ENTER COMMAND

   The RECV program is ready to accept a command from the operator.

3. hhmmss - PORTn - ACTIVE

   A STOP has been issued to the port, by a command or ERROR TERMINATION; ACTIVE indicates there is a data transfer in progress. The STOP will take effect when ACTIVE status is dropped.

4. hhmmss - PORTn - ALREADY ASSIGNED

   The communications link (device) is currently assigned. This assignment must be cancelled before the device can be re-assigned.

5. hhmmss - PORTn - ALREADY STARTED
   hhmmss - PORTn - ALREADY STOPPED
   hhmmss - PORTn - ALREADY CANCELLED

   The communications port specified in the START, STOP, or CANCEL portn command is already started, stopped, or cancelled.

6. hhmmss - PORTn - BUFFER OVERFLOW

   The length of data received does not match the length of data expected.

7. hhmmss - PORTn - CHECK SUM ERROR

   A CHECK SUM error calculated on the current data received does not match the check sum sent.

8. hhmmss - PORTn - DEVICE DOES NOT EXIST

   The device specified in the USE command does not exist. You must select another device.

9. hhmmss - PORTn - DEVICE OUTPUT ERROR

   An error was detected during output to a specified device (e.g. a CRT or printer).

10. hhmmss - PORTn - DISK ERROR

    A disk error has been detected while the program attempted to write the current sector received.

11. hhmmss - PORTn - filename ON d - DOES NOT EXIST - REJECTED
    hhmmss - PORTn - device - DOES NOT EXIST - REJECTED

    The RECV program rejected a transmission request on portn because one of the following conditions existed:

    1. The device or file is assigned;
    2. The device or file does not exist;
    3. The operator rejected the request (this condition valid only in an interactive mode).

12. hhmmss - PORTn - END OF DATA RECEIVE

    RECV received an END OF DATA command from XMIT. This command indicates the end of file was detected.

13. hhmmss - PORTn - INCORRECT COMMUNICATIONS TYPE

    The device assigned to portn is of the wrong type (e.g. it is not a communications device).

14. hhmmss - PORTn - INCORRECT FILE TYPE

    The requested file type does not match the source file type sent by XMIT.

15. hhmmss - PORTn - NO MEMORY AVAILABLE

There is not enough partition memory available to generate another buffer. The partition is at or close to the 32K maximum.

16. hhmmss - PORTn - NOT ASSIGNED

The communications port was not assigned to a proper device when the START portn command was entered. You must assign the port with the USE command before you can use it.

17. hhmmss - PORTn - NOT STOPPED

The operator attempted to cancel or reassign a port without first stopping the port. You must stop a port with the STOP command before you can cancel a port with the CANCEL command.

18. hhmmss - PORTn - OUTPUT RECORD ERROR

This message displays because one of the following conditions exists:

1. An incorrect file type (a file type other than A) has been sent to a device;
2. No END OF FILE or END OF SECTOR has been detected in a 400 byte sector sent by XMIT.

19. hhmmss - PORTn - PROTOCOL - ERROR - RECEIVE

Improperly formatted data was received from XMIT.

20. hhmmss - PORTn - PROTOCOL - ERROR - START

An improperly formatted command was received by the system while it waited for the XMIT start request from the sending system.

21. hhmmss - PORTn - RECEIVE TIMEOUT

A timeout error has occurred on portn. Once reception of file data has begun, the RECV program expects to receive records continuously. If 30 seconds elapse with no data being received, RECV assumes it is a timeout error. It automatically attempts recovery - allowing 14 retries with a maximum wait of 30 seconds for each If all retries fail, the invalid sector, if one was being processed, is discarded and ignored. RECV then sends the RVI (reverse interrupt) status to XMIT and terminates data transmission.

22. hhmmss - PORTn - REQUEST TO device

There is a reception request on portn with data destined for the output device. The operator can answer a yes response to accept transmission or he can enter a no response to reject the request.

23. hhmmss - PORTn - REQUEST TO FILE filename ON d
ACCEPT?

There is a reception request on portn with data destined for the filename on the specified output device. The operator can enter a YES response to accept data transmission or a no response to reject the request.

24. hhmmss - PORTn - nn SECTORS TRANSMITTED, nn SECTORS RECEIVED

The SYSLOG message generated when reception is completed on portn. If the nn record/sector counts are equal, all file data was successfully transmitted. If all data is not sucessfully transmitted, the difference between the number of transmitted sectors and the number of sectors received is the number of records/sectors lost during transmission. This message will also display on the operator's console if RECV is being used in an interactive and not a blind partition.

25. hhmmss - PORTn - SECURITY VIOLATION

A transmission request on portn contains an incorrect security code. The RECV program automatically rejects the request for data transmission. In addition, RECV will automatically stop portn to ensure security integrity. Data transmission is automatically stopped.

26. hhmmss - PORTn - filename ON d - START
hhmmss - PORTn - odevice - START

This SYSLOG message displays when reception is started on portn to filename on disk d at hhmmss time or to output device "odevice."

27. INVALID COMMAND

The operator entered an invalid command. You must enter another command.

28. MAX PORTS OPENED

The maximum number of communication ports possible are now being used.

29. PORTn STARTED
PORTn STOPPED
PORTn CANCELLED

The reception portn has successfully been started, stopped, or cancelled. These messages are also sent to SYSLOG with a preceeding time back.

```
***************
*             *
*    XNEW     *
*             *
***************
```

PURPOSE

XNEW is used to calculate optimum file size/FSI. The calculation is based on an estimated file size that is passed to the program through a job parameter.

PROCEDURE

The following jobstream illustrates the set-up for XNEW:

```
.SETA UPSI=JP=
.SETA JP#=nnn [s t]
.RUN S.XNEW
```

Where:

JP# is the job parameter where the estimated file size is being passed,

nnns s, is the size of the file in sectors,

or

nnnt t, is the size of the file in tracks,

or

nnn a number without either an s or t will be processed as tracks.

ON EXIT:

XNEW will return the calculated file size where the estimated file size was passed and the FSI will be in the next sequential job parameter.

CAUTIONS

The CC will be zero if the calculation was done and CC=100 if an error was detected, also the job parameters remain unchanged.

BISYNC

DATA COMMUNICATIONS UTILITY
(Software Available for Purchase)

CPU-6

March, 1983

```
**************
*            *
*   BISYNC   *
*            *
**************
```

BISYNC UTILITIES
(Sofware Available for Purchase)

PURPOSE

The Centurion Bisync utilities allow direct transfer of data between a Centurion CPU-6 and any other system supporting 3780, 2780 or 2770 protocol.

The Bisync utilities provide you with the following capabilities:

- o File transmission in both directions;
- o Ability to print data from host system on remote printer;
- o Ability to transfer data from CPU-6 to host, from host to CPU-6 or from CPU-6 to CPU-6;
- o Ability to run independently of other jobs on the system;
- o Use of generic driver to eliminate the need for a specialized driver, permitting direct control of interface card from a partition level program;
- o Source code written in FORTH for ease of implementation and to enhance program portability;
- o May be run inter-actively from CRT or from a Command File.

Requirements:

Memory - 32K partition
Hardware - CPU-6
Synchronous modem (customer supplied)
OPSYS - 6.4 or later

Limitations:

Only one BISYNC interface card may be used.

DETAILED OVERVIEW

The BISYNC utilities allow the user to transmit data to and receive data from other systems using a synchronous modem. Three different protocols are supported. They are:

3780
2780
2770

BISYNC utilities have the ability to run inter-actively (controlled from a CRT) or read from a command file which is conceptually similar to JCL. Any of the valid commands may be entered from the CRT in the interactive mode or read from a command file. Additionally, the command file may contain a job file command which will cause the execution of the current command file to be suspended and process the new command file until "end of file". At that time, the original command file will executed, beginning with the command following JOBFILE. This is a process known as nesting; JOBFILE commands may be nested up to 8 levels.

Standard BISYNC File assignments:

	SYS	0	Screen file
*	SYS	1-9	Command files
*	SYS	10	Transmit data file
*	SYS	11	Receive data file
	SYS	12-13	Reserved
	SYS	14	Generic Driver (@GN)
	SYS	15	Output device

* Items assigned during execution.

DETAILED COMMAND DESCRIPTIONS

Command syntax - LOADFILE D# <filename>

LOADFILE resets the Bisync interface and downloads the interface with the object file specified. During reset, the Bisync interface runs a memory diagnostic. If a memory fault is detected an abort is generated.

CAUTION

A LOADFILE command will function at any time. It will unconditionally reset and reload the BISYNC interface.

Command syntax - INITLINE

INITLINE transfers the operational parameters to the program operating in the Bisync interface. See Line Initialization Options.

NOTE: INITLINE will not function if the line has been CONNECTed without an intervening DISCONNECT. As the operational paramaters will likely not change once a connection is made, it is recommended that INITLINE be done immediately after LOADFILE.

Line Initialization Options - (MIN MAX DEFAULT)

Settable parameters

PHYTRM - Physical Terminator (1 2 1)

This is the required terminator for a record in the transmission buffer. It is added after each record following the logical terminator (if any). A value of 1 selects IRS; a value of 2 selects IUS; default is IRS.

LOGTRM - Logical Terminator (0 255 0)

This value is appended at the end of each record if it is non-zero. If the value is 0, nothing is appended. The byte appended is the value specified in SETI.

BLKFAC - Blocking Factor (1 255 100)

The maximum number of records in a transmission buffer.

CMPRES - Compression (0 1 1)

A value of 0 disables space compression. 1 enables compression.

SYNNUM - SYN Limit (3 10 4)

Determines the number of SYN characters preceeding each message. SYN establishes character synchronization at the receiver.

CODE - Code Set (1 2 1)

Establishes the code being used for transmission. A value of 1 flags EBCDIC and data conversion is done by the Bisync Interface. A value of 2 flags ASCII and no conversion is done.

IDLE - Idle Timeout (0 8 8)

Controls how long the Bisync Interface will wait before disconnecting an idle line.

Value	Wait in seconds
0	30
1	60
2	90
3	120
4	150
5	180
6	210
7	240
8	300

BIDLMT - Bit Limit (3 255 20)

This value determines the number of ENQs (bids) that will be sent before a line access failure will be flagged. A value of 255 will permit and unlimited number.

RPTLMT - Repeat limit (3, 255 4)

Controls the number of ENQs (repeat) that can be sent without a response. If the limit expires, transmission fails.

TXLMT - Retransmission limit (4 255 4)

The maximum number of times a message will be retransmitted in response to a NAK. A timeout will abort transmission.

WTLMT - Wait limit (10 255 255)

The maximum number of wait (WACK) messages allowed before transmission aborts. A 255 value is unlimited.

DLYLMT - Delay limit (10 255 255)

Specifies the maximum number of times a delay message (TTD) will be sent before transmission is aborted.

RCVLMT - Receive limit (3 255 3)

This value controls the number of times a "receive timeout can occur. A timeout occurs if no response to a message is received within 3 seconds.

TRMTYP - Terminal type (1 2 1)

A value of 1 indicates primary. A value of 2 is secondary. A primary terminal sends line bids every 1 second. A secondary terminal sends bids every 3 seconds.

BUFSIZ - Buffer size (128 512 256)

The maximum number of characters in a transmission block. Value varies based on unit being emulated.

RECSIZ - Record size (1 512 80)

A logical record's maximum size.

Command Syntax - <#> SETI <PARAM NAME>

SETI sets the initialization parameter <PARAM NAME> to the value <#>. The legal <PARAM NAME>s are given in the Initialization Options section.

SETI should be done prior to INITLINE. After INITLINE has been executed, a spurious SETI may cause erratic performance due to a disagreement between the parameters used by the Bisync interface and the driving program's image of them.

Command Syntax - CONNECT

CONNECT raises the DTR to the data set. CONNECT will wait for an acknowlegement from the operator that the connection has been established (a ctrl-O).

NOTE: After a CONNECT command has been issued, an INITLINE command cannot be issued without an intervening DISCONNECT. See DISCONNECT.

Command Syntax - DISCONNECT

DISCONNECT is issued at the end of a session. It should only be done after the appropriate sign-off procedures with the host. The line must be DISCONNECTed before an INITLINE function.

Command Syntax - \

The line -

\/*SIGNON RMT9

if encountered in a command stream, will cause -

```
/*SIGNON RMT9
```

to be transmitted. The '\' must be followed by a space. '\' will transmit the line that follows it (excluding the initial space) to the host.

A TIMEOUT message may appear indicating the host did not accept the transmission probably because of a disconnect condition.

Command Syntax - TRANSMIT <D#> <FILENAME>

TRANSMIT causes the specified 'A' type file to be transmitted to the host. The entire file is transmitted without regard to content. Records shorter than the RECSIZ set by INITLINE (see Line Initialization Options) will be blank filled to the RECSIZ length.

Command Syntax - / (receive to screen)

'/' reads from the host and displays data to the CRT screen. '/' will make four attempts to read. If no data is received, nothing is done and the next line of the command file executes.

Command Syntax - RECEIVE <D#> <FILENAME>

RECEIVE places the next transmission from the host into the specified 'A' type file.

This command functions like '/'.

Command Syntax - EOF

EOF, when encountered in a command file, will transmit an empty record marked as the last block to the host. This permits signaling the end of a transmission.

Command Syntax - JOBFILE <D#> <FILENAME>

JOBFILE causes the execution of the current command file to be suspended. <FILENAME> will be used until it reaches end-of-file. Execution continues at the next line of the command file that executed JOBFILE.

JOBFILE may be used to nest to a depth of 8 levels.

Command Syntax - SHOW

SHOW causes the lines in a JOBFILE to display on the CRT as fetched for execution. The display occurs before execution is attempted. If the print toggle is enabled, the JOBFILE lines are also recorded in the file assigned to sys015.

Command Syntax - NOSHOW

NOSHOW turns off the display of lines from a JOBFILE.

Command Syntax - PAGE

PAGE clears the CRT screen.

Command Syntax - <x-column> <y-row> XYPOS

XYPOS positions to a specific place on the CRT. For example entry of -

```
1 12 XYPOS
```

will position the cursor to column 1, line 12.

Command Syntax - (and)

When a '(' followed by a space is encountered in a JOBFILE, it flags the beginning of a comment. A ')' terminates the comment. For example, if -

```
( JOBFILE 0 JES2ON )
```

is in a JOBFILE, the line will be treated as a comment and will not be executed.

Command Syntax - CR

A CR causes the cursor to position to column 1 of the next line.

Command Syntax - ." and "

." <message> " causes <message> to be displayed on the CRT. This is used to communicate with the operator.

NOTE: ." must be followed with a blank before <message>.

Command Syntax - BYE

When encountered, BYE terminates the execution of XFORTH. In an applications environment, BYE would normally be used after the DISCONNECT.

NOTE: The Z80 program downloaded to the Bisync Interface card continues to run. Therefore, it is not always necessary to accomplish a LOADFILE or INITLINE after .RUN XFORTH.

Bisync Interface - Software Installation

The Bisync implementation uses a special unit driver called @IODGN. To use, move the file @IODGN from the distribution media onto the system disk. P.OSCON the configuration file to include the special unit. Give it a name of @GN and a PUB size of 32. The file name is @IODGN. All other special unit prompts may be defaulted with a new line.

Move the libraries SAMPJCL, SAMPINIT and SAMPDATA to disk 0..

Move discrete files XZRBTE and XZRBTEDIAG to disk 0. XZRBTE is the Z80 object code downloaded to the bisync interface card.

Move discrete file BSCSTART to disk 0. BSCSTART must be edited for the sys015 assignment if your system does not have a PRTQ or has it under a different name. Sys015 can be assigned to any 'A' type file.

Move discrete files XFORTH and XAFORTH to disk 0. XAFORTH is an auto start version of XFORTH.

Move discrete file BSCSCREENS to disk 0.

Set your default disk to 0 and enter 'BSCSTART'. After the Fig-Forth banner, press a single Control-P. The Control-P is the printer toggle. It is optional. The first Control-P will duplicate all information appearing on the CRT to the file assigned to sys015. The next Control-P will toggle the print off. Entering -

```
EPRINT .
```

followed by a NEW LINE will display the current state of the toggle; a 1 indicating on and a 0 indicating off.

NOTE: Turning the printer toggle on, opens the file on sys015; off closes the file. If the output file is an 'A' type file, it will only have the contents of the last printer interval. If the sys015 is to the spooler, entries will be made in the spool file.

Enter -

```
1 LOAD
```

followed by NEWLINE.

The Bisync will then load up and ask you to make a connection to a specified telephone number. It will send the necessary JCL to sign on and to run a sample job.

Physical Configuration: - interface card

The BISYNC card may be placed in any open slot in the card cage.

Physical Description:

When viewed from the cable connector end of the interface card, the top connector is for the "DMA" cable. The second, somewhat larger connector is for the DB25S modem. The bottom connector is for DB25S modem diagnostic terminal.

Switch Settings:

There are three "DIP" switches located on the interface card at the following co-ordinates: C3, F10, K5. The switch settings are as follows:

C3 - all off
F10 - off, on, on, on, on, off, on, on
K5 - on, on, on, on, on, off, on, off

Jumper Locations:

There should be four jumpers within the jumper area (i.e. between "B5" and "B6"). Viewing the interface from the component side, with the backplance connector to the left, the left-most jumper should be to the left; all others should be to the right.

Physical Configuration - Bell 201C Modem

- Transmitter timing from data set
- No auto call unit
- Auto answer if DRT
- Ring indicates EIA voltage
- Half Duplex
- RTS Controls Carrier
- No New Sync

Switch Settings:

The 201C modem is equiped with an 8 position DIP switch which should be set as follows:

off, on, off, off, off, off, on, off

NOTE: The switch settings DO NOT correspond to the above listed 8 options.

GENERAL NOTES

The Bisync system, with the exception of the Z80 object file downloaded to the interface card, is written in FORTH. This was done both for ease of implementation and for considerations of program portability. Additionally, the pool of FORTH experienced programmers is much larger than that for CPL or CPU-6 Assembler. Finally, the source for the Bisync function is available for examination, modi- fication and/or enchancement at each installed location.

The best learning publication for FORTH is: Starting FORTH, Leo Brodie, Prentice-Hall, Inc., publishers, Englewood Cliffs, N.J. 07632.

KTEST

VSI FILE UTILITY

CPU-6

March, 1983

```
***************
*             *
*    KTEST    *
*             *
***************
```

PURPOSE

KTEST is the utility to display and make limited changes to VSI files (I type).

PROCEDURE

Assign the VSI file to SYS0; if needed, assign the printer to SYSLST. Then run P.XKTEST.

The main menu will be displayed (see below). Select the appropriate option from the main menu. Each selection is described in the following pages.

The KTEST utility requires that the user be familiar with the basic input/output functions of a VSI file. Further knowledge is required for the key area and free chain functions. Refer to the description of VSI files, located at the end of this document, for additional information.

NOTE: When asked "SPACE WHEN FINISHED WITH PAGE", a "Q" will exit the current function. Any other key will continue on.

Main Menu

```
 0  -  End Program
 1  -  GETK
 2  -  NEWK
 3  -  DELK
 4  -  GETR
 5  -  PUTR
 6  -  HLDR
 7  -  FRER
 8  -  display record zero
 9  -  change record zero
10  -  display/change key record
11  -  list key area
12  -  print graphic analysis of key area
*13 -  print numeric analysis of key area
14  -  display/modify free chain
```

*NOTE: This option does not exist, but performs option 14 instead.

Option 1 - GETK

Program displays:

OLD ARG IS and argument

Prompts:

ENTER ARGUMENT

If NEWLINE or "plus bar" is entered, the old argument is used; otherwise the new argument is used.

GETK is performed.

The key record for the argument entered is displayed:

KEY	PDP	CHN	ARG
relative key # in index area	prime date pointer	chain pointer	argument

If the status on the GETK was not zero, then the status is displayed (i.e. the key record is meaningless).

The program then prompts the operator before returning to the main menu.

Option 2 - NEWK

The program prompts for an argument in the same manner as GETK (option 1). If the argument entered is zero (null for string argument), then the program returns to the main menu.

If for some reason the record length is zero, the program prompts for a new relative key (see GETR).

The program then performs the NEWK and displays the new relative key number.

The program performs a GETK and displays the key record in the same manner as the GETK option (option 1). The status displayed is that of the NEWK.

Option 3 - DELK

Enter the argument in the same manner as the GETK option (option 1). DELK is performed and status is displayed. If a non-zero status occurs, the operator is prompted before returning to the main menu.

Option 4 - GETR

Program displays:

OLD KEY IS: key

Prompts:

ENTER RELATIVE KEY

Enter the new relative key.

If NEWLINE or "plus bar" is entered, the old key is used. If zero (0) or a negative key were entered, the program returns to the main menu.

Program displays:

OLD REC IS: old record length

Prompts:

NEW RECORD LENGTH: record size.

If NEWLINE or "plus bar" is entered, the old record length (i.e. record size) is used. If a new length is entered, it will be used for this "read" only. Subsequent I/O will use the old record length.

GETR is performed.

Status is displayed.

If a non-zero status occurs, the program prompts the operator before continuing.

Record is displayed in hex format.

Operator is prompted before continuing with another GETR.

To exit, enter zero (0) as the new relative key.

Option 5 - PUTR

The relative key and record size are entered in the same manner as the GETR option.

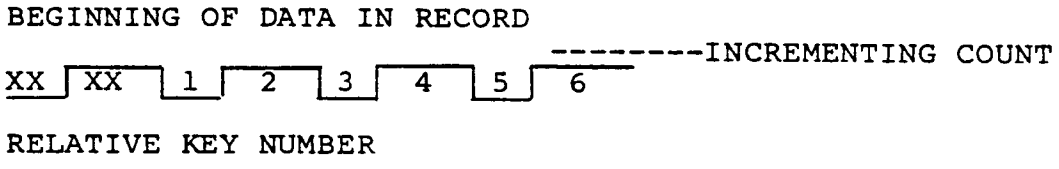
A record filled with a count pattern (see below) is then written out to the file.

Status is displayed.

Processing continues with another PUTR.

To exit, enter zero (0) in response to the relative key prompt.

Count pattern:



Option 6 - HLDR

The relative key is entered as in the GETR option.

HLDR is performed.

Status is displayed.

If a non-zero status occurs, the operator is prompted before continuing.

Process continues with another HLDR.

To exit, enter zero (0) to the relative key prompt

Option 7 - FRER

This function is identical to HLDR, except a FRER is performed instead of a HLDR.

Option 8 - Display Record Zero

Record zero is "read" and the contents are displayed as follows:

NOKEYS KEYLEN RECLEN CHAIN BEGIN

NOKEYS - Number of keys in the index area

KEYLEN - Key length (4,6,7-35)

RECLEN - Data record length

CHAIN - Relative key of first record in prime data free chain.

BEGIN - Beginning of prime data area

The operator is prompted before returning to the main menu.

Option 9 - Change Record Zero

Record zero is displayed (see Display Record Zero - Option 8) with field numbers.

Enter the field number of the item to be changed.

Enter zero (0) to exit.

Enter the new data to be written into the field.

Record zero is written out and re-displayed.

NOTE: Although fields 2 and 3 prompt for new data, the key and record length will remain unchanged.

Option 10 - Display/Change Key Record

Enter the relative key to display or change; (see GETR option).

NOTE: Zero will exit and return to the main menu.

The key record is read and displayed in the same manner as the GETK option.

Field numbers are placed above the PDP, CHH and ARG fields to allow changes.

Operator is prompted:

SPACE WHEN FINISHED WITH PAGE

Enter "Q" to return to the main menu.

Enter "P" to back up to the prior key in the chain; this is only valid after an "N".

Enter "N" to display the next key in the chain pointed to by CHN. Once the end of the chain is reached, the last key in the chain will continue to be displayed.

Enter "C" to change a field in the key record; see below for additional information on the key record.

Entry of any other key will re-dislay the current record, allowing another key record to be displayed/changed.

Program displays:

CHANGING THE KEY RECORD ('C'):

Enter the field number to change.

NOTE: A valid field number must be entered.

Enter the new data to be written to that field. The new key record is written out and re-displayed.

Option 11 - List Key Area

The operator is prompted for the relative key at which the listing and output is to begin.

The key (index) area is then listed as follows:

KEY PDP CHN ARG

When the screen fills up, the operator is prompted to continue on to the next page.

If printer output was not selected, the operator will be prompted as follows:

ENTER A 'Q' TO EXIT THE LIST.

Option 12 - Print Graphic Analysis of Key Area

Enter the relative key to begin the display. The actual display will begin with the next lower key number divisible by 20.

Enter "Y" or "N" to the "DO YOU WISH TO OUTPUT TO PRINTER ALSO" prompt. The key area is then displayed showing the keys used as a "1" and the keys not used as "-". The graph shows 20 keys per line, with the relative key number of the first key in a line on the left.

The operator is prompted to continue when a page fills up (if the printer option was not selected). Enter a "Q" to exit the display.

Option 14 - Display/Modify Free Chain

The program then reads the first 3 bytes of the record using a GETR and displays:

VALUE IS:	pointer in 1st 3 bytes	KEY:	relative record # just read

Program displays:

SPACE WHEN FINISHED WITH PAGE

Enter one of the following:

"Q" - to quit and return to the main menu;

"C" - to change the free chain pointer;

Note: Enter the data as prompted. The pointer is written back to the file and the program continues with another free chain record.

"N" - to go to the next record in the free chain;

Note: If the free chain pointer is zero (0), then the next relative record is read. The record is read, displayed and the "SPACE WHEN FINISHED" prompt re-displays.

"P" - to back up to the previous record in the free chain.

Note: This is only valid after "N".

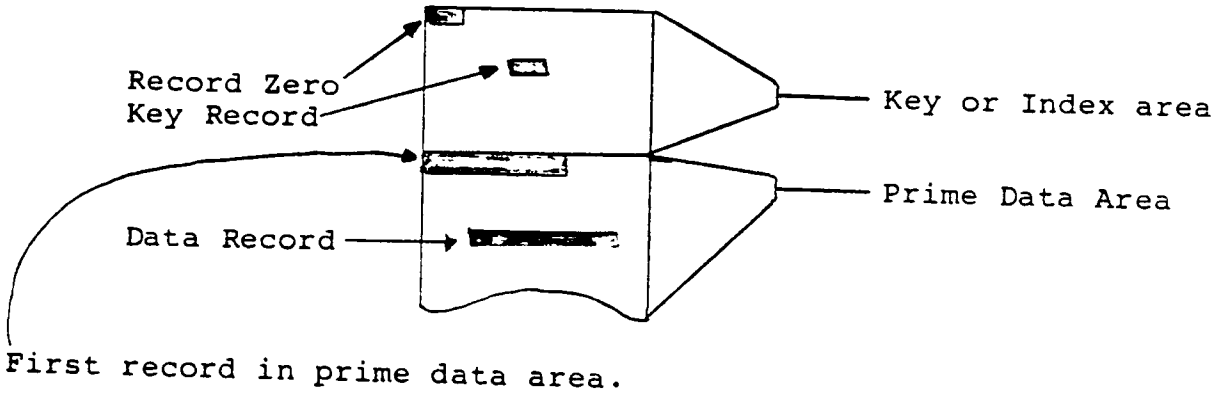
GENERAL INFORMATION - VSI FILES

Record 0	bytes
Number of keys (i.e. key records) in index area	3
Prime data record length	2
Key length	1
Relative key of first record in prime data free chain	3
Beginning of prime data area (relative record#)	3

Key Record	
Prime data record number	2
Chain to next key	2
Argument	4-35

Key Record-Extended	
Prime data record number	3
Chain to next key	3
Argument	4-35

VSI DIAGRAM:



GENERAL INFORMATION - GETK NEWK DELK

Given an argument, these routines put the argument through an algorithm that results in a relative key number.

That key record is read and arguments compared. In the case of GETK and DELK, the key chain pointer are followed until the correct argument is found or until the end of the chain (x '00' or x'FF'). In the case of NEWK, the chain is followed until a free key is found (prime data pointer is 0). The prime data pointer is then made available to the user through the 'KEY=' integer in the FILE statement. Also status is set depending on the situation (i.e. end-of-chain, if GETK or DELK and no matching argument, out-of-space if NEWK).

GENERAL INFORMATION - PRIME DATA FREE CHAIN

This chain is not to be confused with the key record chain. The key record chain keeps up with the situation where multiple arguments give rise to the same relative key from the algorithm. The free chain keeps up with free prime data records. At a file's creation, the first free data record is the first data record in the file. As records are added, the first free data record pointer increments. No chain is set up yet. The chain occurs when DELK is performed. Each time a DELK is performed, the data record is made the first free data record and a pointer is put in the first 3 bytes of the record.

This pointer points to the next free record. A pointer of zero indicates the end of the chain; free records are sequential from that point on. In other words, as free records are needed, the previous DELK'ed records are processed in a "last in, first out (LIFO)" method.

STREAMER TAPE/FINCH SYSTEMS

CPU-6

JUNE, 1983

```
**********
*  STREAMER  *
*  BOOT TAPE  *
*   LAYOUT   *
**********
```

Purpose

MAKEBOOT is a JCL jobstream used to create a Streamer Boot Tape. The following files must exist as discrete files on the default disk (#D):

0	@SB
1	MAKEBOOT
2	T@REST
3	T@FMT
4	@OJX53
5	?FINCH
6	?TMOS
7	?STST

NOTE: File @OJX53 will also be used; it is taken from @SYS on the system disk (#S).

Procedure

1. Enter the following to call the MAKEBOOT jobstream:

```
MAKEBOOT tapename d
```

tapename	-	device name of Streamer Tape (sysgen)
d	-	disk number where the above discrete file exist

Contents - Streamer Boot Tape

File 0 @SB Streamer Boot Program

1 MAKEBOOT JCL for creation of a Boot Tape in XCOPUT form

2 T@REST stand alone streamer tape to Finch Restore

3 T@FMT stand alone Finch Format
Formats and has option to write loader track

4 @OJX53 taken from @SYS

```
********************
*  STREAMER TAPE   *
* BACKUP/RECOVERY  *
*    COPY OPTIONS
******************
```

OVERVIEW

There is one method of backup/recovery on a Streamer Tape/Single Finch Drive configuration. This method includes:

1. @REST/T@REST stand alone package described in this section.

There are two utilities that allow single file copies to be copied to and from Streamer Tape. These include:

1. S.COPT is a utility used to copy a file to tape.

2. S.TCOP is a utility used to copy a file from tape.

STREAMER TAPE/SINGLE FINCH DRIVE BACKUP AND RECOVERY

Backup Procedure

1. The method of Backup is the .REORG to TAPE utility described in the CPU-6 System Utilities Program Manual.

Recovery

1. The stand alone @REST/T@REST package may also be used for full or partial recovery of data from the Finch Drive; for additional information on the @REST/T@REST utilities, see documentation in this section.

NOTE: The current .REORG from tape to disk is destructive. Using a JCL line similar to the following -

.REORG tape# to d

will do a CLR on the disk, destroying whatever data was there. For this reason, .REORG from tape to disk processing will abort in the following instances:

1. The disk is the systems disk;
2. The disk is the default disk;
3. The disk has a file assigned.

Caution

If selective files are desired from a .REORG(ed) tape onto a single Finch system - and it is not a recovery procedure - @REST/ T@REST should not be used. This procedure clears the disk (see.CLR), destroying all previous data. The S.TCOP utility should be used instead.

S.TCOP

If it is desired to copy a file from tape, proceed as follows:

1. Enter the following:

S.TCOP file# tapename filename d

where

file#	file number of tape
tapename	device of tape (SYSGEN)
filename	name of file to be copied
d	disk number of file

This will "read the file from tape at the file number selected". Caution should be used to make sure you have identified the proper file# on tape. Use the S.XTDSP utility to locate such file (See S.XTDSP following).

S.TPLST

This jobstream is used to read the files from a cartridge tape and produce a "snapshot" display or printout of the file# and corresponding file name.

1. Enter the following:

S.TPLST filenumber device name (Y/N)

where

filenumber	is the file number to position the tape to before the list begins.
device name	device name of tape (SYSGEN)
(Y/N)	if yes (Y) is requested, all records and the first 60 bytes of data are displayed. No (N) will display filename, last record and length.

This utility will read each file on the cartridge tape and identify it with a file number. The file name will be the first 10 characte of the snapshot. This utility is very helpful in locating file# fo use with the S.TCOP jobstream.

S.COPT

If it is desired to copy a file to tape, proceed as follows:

1. Enter the following:

```
S.COPT filename d file# tapename
where
     Filename  - name of file to be copied;
     d         - disk number of file;
     tapefile# - file number on tape;
     tapename  - device of tape (SYSGEN).
```

This will "write the file to tape at the file number selected. Subsequent files may be copied using the procedure outlined above.

2. After the last file has been copied, enter the following:

```
.WEOT TAPENAME
.WEOT TAPENAME
```

This writes an 'End of Tape' mark twice for additional informati see the CPU-6 JCL Manual

S.TCOPN

This jobstream is used to copy a file from tape to disk either by using the file number of the file name.

1. Enter the following:

S.TCOPN (file# or filename) tpename filename d
where

file#	-	number of desired file on tape or
filename	-	name of desired file on tape
tapename	-	device of tape (SYSGEN)
filename	-	filename of target file on disk to be copied to.
d	-	disk number of target file.

The jobstream will prompt for:

FILENAME Y/N?

If a search of the file by filename is desired, answer (Y) yes; a (N) no will default to a filenumber search.

```
**********
*        *
* @REST/T@REST *
*        *
**********
```

Purpose

T@REST and @REST are stand alone restore programs that allow the contents of a backup streamer tape created by the .REORG command to be restored to a Finch Drive (32 MB maximum). For additional information on the .REORG command, see the Mag Tape Documentation located in the JCL Manual.

This program may be booted off the system disk (@REST) or from a boot tape (T@REST) folowing the streamer boot procedures outlined in this section. If it is necessary to format the disk, refer to section on the Booting Off Tape.

Procedure

Booting Off Disk

After making sure the R/F button (CPU-6) is in or the DISK SELECT (MicroPlus) is on, press the SELECT button.

1. CRT displays

   ```
   WDIPL 6.2
   NAME=
   ```

2. Enter:

   ```
   @REST (NEWLINE)
   ```

3. CRT displays:

   ```
   DISK=
   ```

4. Enter:

   ```
   Disk Number (NEWLINE)
   ```

5. CRT displays:

   ```
   CODE=
   ```

6. Enter:

   ```
   Proper code (NEWLINE if none)
   ```

Booting Off Tape

T@REST and T@FMT must first be on tape in 'XCOPUT' form.

NOTE: The JCL command, MAKEBOOT, performs this function automatically placing T@REST on tape as file number 2.

After making sure the R/F button (CPU-6) is out or the DISK SELECT (MicroPlus) is off, press the SELECT button. Then perform the following:

1. CRT displays

   SB Ø.2 - STREAMER BOOT

   FILE # (ØØ1-255)

2. Enter:

   (NEWLINE)

   ØØ2 FOR T@REST

   ØØ3 FOR T@FMT

Operation T@REST (Stand Alone Restore)

Following the booting procedure, the disk(s) VOLUME NAME(S) and STATUS are displayed at Syslog (CRTO) as follows:

```
# VOL

O SYSTEM  Ø1/Ø3/83

ENTER DISK NUMBER TO BE RESTORED #
```

NOTE: Do not respond to DISK number until step #3 is fist performed, and tape drive is in the READY status.

3. Remove boot tape and insert .REORGed tape.

4. Type in disk number from above list to be restored to. The following will then be displayed at Syslog:

   ENTER OPTIONS

5. Type in the following keywords as desired; NEWLINE if none are required.

   NOTE: USE UPPER CASE ONLY

   DSIZE=2T, USIZE=2T, FSIZE=1T, SELECT=Y, CONDENSE=Y*

DSIZE, USIZE, FSIZE — If the size of the directory, used allocation list and/or free allocation list is to be different on the target disk, then the size in sectors or tracks can be specified. If not specified, the size of these will be the same on both the source and target disk.

SELECT — The SELECT option will cause the system to display the file name, old file size, new file size, old FSI and new FSI; then it will wait for a response.

If "Y" or "+" is entered, the file will be copied according to the options specified in the original "REORG" statement.

If "N" or "-" is entered, the file will not be copied.

If "C" is entered, the optimum file size and FSI will be displayed; the console will solicit the new size and FSI.

If an invalid size of FSI is entered, the information will be resolicited.

If zero (Ø) is entered for either field, the optimum value will be used.

Caution

If selective files are desired from a .REORG(ed) tape onto a single Finch system (and it is not a recovery procedure), @REST/T@REST should not be used. This procedure will CLEAR (.CLR) the disk destroying all previous data. The S.TCOP utility should be used. For additional information, see Mag Tape documentation in the CPU-6 JCL Manual.

CONDENSE - If CONDENSE=Y, type 'A', 'B', and 'E' files are copied at their true length (i.e. a 10-track file containing 8 track of data will be copied into 8 tracks on the target disk).

If CONDENSE=N, files will be the same length on the target disk as the source disk.

Once all options have been entered, the target disk is cleared with the 'sizes' on tape or the ones entered with keywords. A listing is output to Syslog (CRTO), showing the names of the files copied, their type, size, FSI and number of records (where appropriate). If the size of the file is changed during RESTORE (i.e. file is condensed), the difference in size is displayed. If a file could not be copied, the listing will display an error next to the file name.

Once all files that could be copied have been displayed, the number of errors, if any, is displayed.

At this time the system may be booted via normal system boot procedures.

Operation - T@FMT (Stand Alone Finch Format)

A. Following the Boot Procedure, the following prompt will appear:

DO YOU WISH TO WRITE A LOADER TRACK?

B. If 'YES' the disk volume names will be displayed.

#VOL

0 NAME XX/XX/XX

ENTER DISK NUMBER FOR LOADER TRACK #

Enter the number of the disk you wish to write a loader track to. Upon completion the following appears:

LOADER TRACK WRITTEN

DO YOU WISH TO FORMAT?

At this time type a (N) for no or (Y) for yes. If yes, the disk volume names will be displayed.

#VOL

0 NAME XX/XX/XX

ENTER DISK NUMBER TO BE FORMATTED #

This is the disk number to be formatted. The format will run 5 passes, indicating any errors encountered. After the fifth pass the following will appear. Refer to error message in documentation for .FORMAT.

DONE

DO YOU WISH TO WRITE A LOADER TRACK?

If previous operation was a "FORMAT OPERATION" the disk will now be ready for system initialization using a .REORG data tape. Refer to overview page IV-101,, booting off tape and T@REST operation page IV-104..

Cautions

1. When the tape drive is not in use the tape should be rewound and removed from the tape drive.

2. Do not attempt to perform I/O to the streamer tape or answer any prompts requiring the streamer tape until the device is in the 'LOAD' status.

3. When running S.TCOP and S.COPT be sure you are reading/writing the proper file number from/to the streamer. Use the S.XTDSP to display the CRT or print to the printer to be sure you have the proper file number and file name before running S.TCOP. Be sure you are not writing over an existing file number on the streamer tape when running S.COPT.