McMASTER UNIVERSITY

90

TABLE OF CONTENTS

TABLE OF CONTENTS- contd.

I. OPERATION

92

McMASTER UNIVERSITY

COMPUTATION SERVICES UNIT

HEALTH SCIENCES CENTRE

HAMILTON, ONTARIO, CANADA

L8S 4J9

TITLE	ACUMLOG
FUNCTION	To analyze HPS/3000 Log Files and append a summary record into a Master File for each JOB and SESSION
AUTHOR	P. Balnys
LANGUAGE	FORTRAN

DESCRIPTION

The program prompts the operator for the name of the LOG File to be analyzed and the name of the Master File. A Master File will be automatically built if required. The information for each JOB and SESSION on the Log File is summarized and a record written to the Master File as well as a report written to the Line Printer. A monetary charge for each JOB and SESSION is indicated as well as identification information and resource utilization.

SPECIAL CONSIDERATIONS

1) Program requires the Line Printer.
2) Program uses two temporary disc files.
3) Subroutines DATE and COSTING are utilized.

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	MTHLOG
FUNCTION	To produce a monthly report from the Master Log File.
AUTHOR	P. Balnys
LANGUAGE	FORTRAN

DESCRIPTION

This program is run after a SORT of the Master LOG File has been done. The Master Log File must be organized by ACCOUNT, USER, JOB. A report is produced containing detailed records for each JOB and SESSION as well as summary information for each JOB and USER and ACCOUNT.

SPECIAL CONSIDERATIONS

1) Program requires the Line Printer.
2) Master Log File must first be sorted by ACCOUNT name, USER name and JOB name.
3) Subroutine DATE is utilized.

McMASTER UNIVERSITY

COMPUTATION SERVICES UNIT

HEALTH SCIENCES CENTRE

HAMILTON, ONTARIO, CANADA

L8S 4J9

TITLE	SUMLOG
FUNCTION	To produce a summary report of the sorted Master Log File for chargibg purposes.
AUTHOR	P. Balnys
LANGUAGE	FORTRAN

DESCRIPTION

This program produces a report much like the MTHLOG program but no detail records for each JOB and SESSION. All Master records with the same JOB name, USER name and ACCOUNT name are summed together to produce a charging report. A summary record is also produced for each USER within an account and for each ACCOUNT.

SPECIAL CONSIDERATIONS

1) The Master Log File must be sorted.
2) The program requires the LP.

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	CARD
FUNCTION	To build a file corresponding to a USER ACCOUNT and copy card image data into this file
AUTHOR	P. Balnys
LANGUAGE	FORTRAN

DESCRIPTION

This program is used as an intermediate spooling mechanism to input data to a spooling input file.

The specified file is first purged, if it exists, and then built so that an empty file will first exist into which the data is entered.

SPECIAL CONSIDERATIONS

A special account, namely CR must be provided and contain a group corresponding to all other accounts on the system. The USER name becomes the file name

e.g. SMITH. GENERAL. CR

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	PRINTER
FUNCTION	To empty the specified disc file to the Line Printer
AUTHOR	P. Balnys
LANGUAGE	FORTRAN

DESCRIPTION

This program is used as an intermediate spooling mechanism. A specified file corresponding to a USER ACCOUNT is dumped to the Line Printer and the file is Purged and rebuilt

SPECIAL CONSIDERATIONS

Batch job

AN INTERIM SOLUTION FOR SPOOLING

I. OBJECTIVES:

1. To enable the time-share users to read cards from the Card Reader and output results to the Line Printer.
2. The procedures established should be simple for the operator and user to follow.
3. When spooling in MPE is available next April, the effect on the user should be minimal.

II. METHODS:

The method is to assign to each user of the system, who has the need for spooling, two files for their card reader input and printer output. By choosing meaningful names for the files, Groups and Accounts, they can be referred to easily. Also, since all printer files are within one account, the operator can unspool them with relatively simple utility programme.

III. PROGRAMME UNITS:

1. Programme Name: CARD
   Function : To create a card image file
   RUN Instructions:

```
:JOB MGR.CR
:RUN CARD.PUB.LIB
 USER.ACCT
    data cards
:EOD
:EOJ
```

2. Programme Name: PRINTER
   Function : To list an ASCII file to Line Printer

RUN Instructions:

```
:JOB MGR.LP
:RUN PRINTER.PUB.LIB
 USER.ACCT
:EOD
:EOJ
```

IV. LIMITATIONS:

1. There is a maximum of 2000 cards for card input and 3000 lines for printer output. This can, however, be changed by modifying the BUILD commands when the spool files are created by MGR.CR/MGR.LP initially and in the PRINTER program.

2. Neither the card or printer spool file can be accessed simultaneously by more than one running programme by the same user. That is, the user must wait until his files have been unspooled before running any programme that will require them.

V. OPERATING INSTRUCTIONS:

Create two new accounts CR AND LP with MGR as the only user in each account. The capabilities to be used are: IA, BA, ND, SF.

<u>MGR.CR/MGR.LP</u>:

Create for each user who needs spooling, the spool files with the commands:-

```
:BUILD USER.ACCT.CR;REC= -80,3,F,ASCII;DISC=2000,16
:BUILD USER.ACCT.LP;REC=-130,1,F,ASCII;DISC=3000,16;
                                                  CCTL
```

<u>CONSOLE OPERATOR</u>:

1. Run CARD as soon as the user's deck is submitted.

2. Run PRINTER when requested by the user.

96

USER.ACCT:

A. Card Input:

1. Submit the deck to run CARD and wait until the job is done.
2. Use the FILE command to equate the formal file designator for the card input file to *USER.ACCT.CR*

Examples:

i. FORTRAN Compile:-

```
:FILE CARD = USER.ACCT.CR,OLD
:FORTRAN * CARD , uslfile, listfile
```

ii. FORTRAN RUN:-

```
:FILE FTN02=  USER.ACCT.CR,OLD
:RUN programme
```

iii. EDITOR Run:-

```
:FILE CARD = USER.ACCT.CR,OLD
:EDITOR
/TEXT * CARD, UNN
```

B. Printer Output:

1. Use the FILE command to equate the formal file designator for the printer file to *USER.ACCT.LP*

Examples:

i. FORTRAN Compile:-

```
:FILE  PRINTER = USER.ACCT.LP,OLD
:FORTRAN sourcefile, uslfile, * PRINTER
```

ii. FORTRAN Run:-

```
: FILE FTN03 = USER.ACCT.LP,OLD
:RUN programme
```

2. Use:TELLOP to inform the operator of printer output.

II. UTILITY

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	Subroutine BUSYTEST (NAME, LU, IAC, IFLAG)
FUNCTION	To test whether a specified file is currently open (in use) and if so, return a flag indicating this.
AUTHOR	P. Balnys
LANGUAGE	FORTRAN

DESCRIPTION

Input Parameters:

```
Name - Actual file name with group and account if required,
       contained in this character variable of length 26.
LU   - Fortran logical unit number to be used by the calling
       program when referencing this file.
IAC  = 0 indicates READ/WRITE access is desired
     = 1 indicates APPEND access is desired
```

Output Parameters:

```
IFLAG = 0 means the file has been opened for this program use
      = 1 means the file is busy
      = -1 means a fatal error from opening the file has
        occurred
```

SPECIAL CONSIDERATIONS

RUNNING CDC-6400 DECKS ON HP/3000

The following steps will enable you to run a CDC Fortran deck on to the HP/3000.

1. Remove all CDC control cards,

   e.g. Job card, FTN, LGO, EOR, EOF,...etc.

2. Replace

   ```
      PROGRAM TST(INPUT,OUTPUT,....)   card
   by
      $CONTROL FILE=5,FILE=6,LABEL,MAP card
   ```

   Any other files used in the programs should be declared similarly.

3. If you have any end-of-file test in the program

   e.g. IF (EOF(5)...,...)

   then it should be removed and the end-of-file test should be included in the appropriate READ statement,

   ```
   e.g.     READ (5,100,EOF=n)...,...
        100 FORMAT (.........)
   ```

   When an EOF occurs on logical unit #5, this statement will transfer control to label n.

4. The range of integer values on HP/3000 is limited to the range between -32768 and +32767. If you feel that an integer variable in your program may take a value beyond this range, you should declare it as a REAL variable (-and change the FORMAT statements, if necessary).

5. The deck should not contain calls to library subroutines that do not exist in the HP/3000 compiler library. You should consult the manual on HP/3000 Compiler Library.

6. If your program reads alphanumeric strings under FORMAT (...,A10,...), you will need to change the corresponding variables(s) to type CHARACTER *10.

7. If you have any doubts or questions please consult the programming assistant in Room 2D9.

...P.2

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	CDCCONV
FUNCTION	To convert a CDC-6400 FORTRAN program for use on HP/3000
AUTHOR	K. Ahmed
LANGUAGE	Text Editor command language

DESCRIPTION

CDCCONV is a USE- file to be used by HP/3000 Text Editor. It replaces the 026 punch characters by the corresponding 029 punch characters. It also replaces all the asterisks (*) by apostrophes (') in all the FORMAT statements.

SPECIAL CONSIDERATIONS

The CDCCONV use file does most of the work of conversion; however, some additional minor conversions have to be done manually.

CDCCONV uses a compiled subroutine CORRECT which must be stored in the Project SL.

EXAMPLE

```
/
[illegible]T ALL
                PROGRAM TST %INPUT,OUTPUT,TAPE5#INPUT,TAPE6#OUTPUT<
   2      C
   3      C  EXAMPLE OF A CDC PROGRAM CONVERTED TO H.P, USING THE EDITOR.
   4      C
   5             READ%5,10< N
   6      10     FORMAT%I4<
   7             DO 20 I#1,N
   8             READ%5,11< X
   9      11     FORMAT%F10.5<
  10             S#10.*SIN%X<
  11             C#10.*COS%X<
  12             CALL PLOTPT%X,S,4<
  13             CALL PLOTPT%X,C,5<
  14      20     CONTINUE
  15             CALL OUTPLT
  16             WRITE%6,30<
  17      30     FORMAT%1X,*PLOT OF SINE AND COSINE FUNCTIONS*,/,
  18            1 1X,*THE Y-AXIS IS SCALED UP BY A FACTOR OF 10.*<
  19             STOP
  20             END
/
/USE CDCCONV.PUB.LIB

*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
[illegible]TRING NOT FOUND BEFORE LIMIT        Ignore these messages.
A[illegible]EPTH 3
/
/LIST ALL
   1             PROGRAM TST
   2      C
   3      C  EXAMPLE OF A CDC PROGRAM CONVERTED TO H.P, USING THE EDITOR.
   4      C
   5             READ(5,10) N
   6      10     FORMAT(I4)
   7             DO 20 I=1,N
   8             READ(5,11) X
   9      11     FORMAT(F10.5)
  10             S=10.*SIN(X)
  11             C=10.*COS(X)
  12             CALL PLOTPT(X,S,4)
  13             CALL PLOTPT(X,C,5)
  14      20     CONTINUE
  15             CALL OUTPLT
  16             WRITE(6,30)
  17      30     FORMAT(1X,'PLOT OF SINE AND COSINE FUNCTIONS',/,
  18            1 1X,'THE Y-AXIS IS SCALED UP BY A FACTOR OF 10.')
  19             STOP
  20             END
/
[illegible]     EX4
```

```
:JOB username.acctname
:EDITOR
/ADD
$CONTROL FILE=5,FILE=6,LABEL,MAP
     :
     :
     :
  (CDC source cards)
     :
     :
///
/USE CDCCONV.PUB.LIB
/LIST ALL
/KEEP filename
/END
:EOD
:FORTRAN filename
:PREPRUN $OLDPASS;PMAP
     :
  (Data cards)
     :
:EOD
:SAVE $OLDPASS, Progname     (if you want to save the
:EOJ                          compiled program for future runs).
```

These commands convert the CDC punch (Ø26) to the HP punch (Ø29) and store the source as a permanent file under filename specified. This is required only on the first run.

In case you do not want to save the source file, you should purge it at the end of the run,

```
     :
     :
:PURGE filename
:EOJ
```

otherwise it will occupy unnecessary file space.

If you have saved the compiled program (ref to MPE-JCL) (by :SAVE $OLDPASS, Programme), then in the future runs, the deck set-up should contain

```
:JOB username.acctname
:RUN Progname

     :
     :
  (Data cards)
     :
     :
:EOD
:EOJ
```

Please purge all unnecessary source and program files.

GENERAL I/O SUBROUTINES FOR HP S/3000

BY: Paul Balnys

I N D E X

McMASTER UNIVERSITY

COMPUTATION SERVICES UNIT

HEALTH SCIENCES CENTRE

HAMILTON, ONTARIO, CANADA

L8S 4J9

TITLE	GENRLIO - RL file
FUNCTION	Group of subroutines designed to handle free field input and output
AUTHOR	P. Balnys
LANGUAGE	FORTRAN and SPL

DESCRIPTION

The subroutines perform the input, conversion and output of data. Error recovery is left to the calling program. The correction of input can be performed interactively by the use of special instructions similar to those used by the EDITOR subsystem. Provision has been made for entering more than one response at a time as well as continuing over more than one line.

SPECIAL CONSIDERATIONS

Incorporate these subroutines into your program by:

```
PREP $ OLDPASS, $ NEWPASS; RL = GENRLIO
```

II. BASIC OBJECTIVES

1. To provide programmers using the HP S/3000 with the capability to handle free field input and output with minimum programming discomfort.

2. To facilitate standardization among programmers which will inhibit diversified language and machine dependence.

3. To enhance the standard FORTRAN format capabilities to include double integer representation. (i.e. integers out of the range of $\pm$32, 768.)

4. To save memory by ensuring that these subroutines are sharable (re-enterent) by different users simultaneously.

5. To allow the programmer to specify file characteristics; such as, input/output file numbers, length of input/output records, etc.

6. To provide an edit capability for user correction of invalid input.

7. To provide error recovery in control of the calling programme.

8. To perform conversion of numeric ASCII characters and special ASCII characters; such as, +,-,.,E,etc. into a specified internal representation and vice versa.

10²

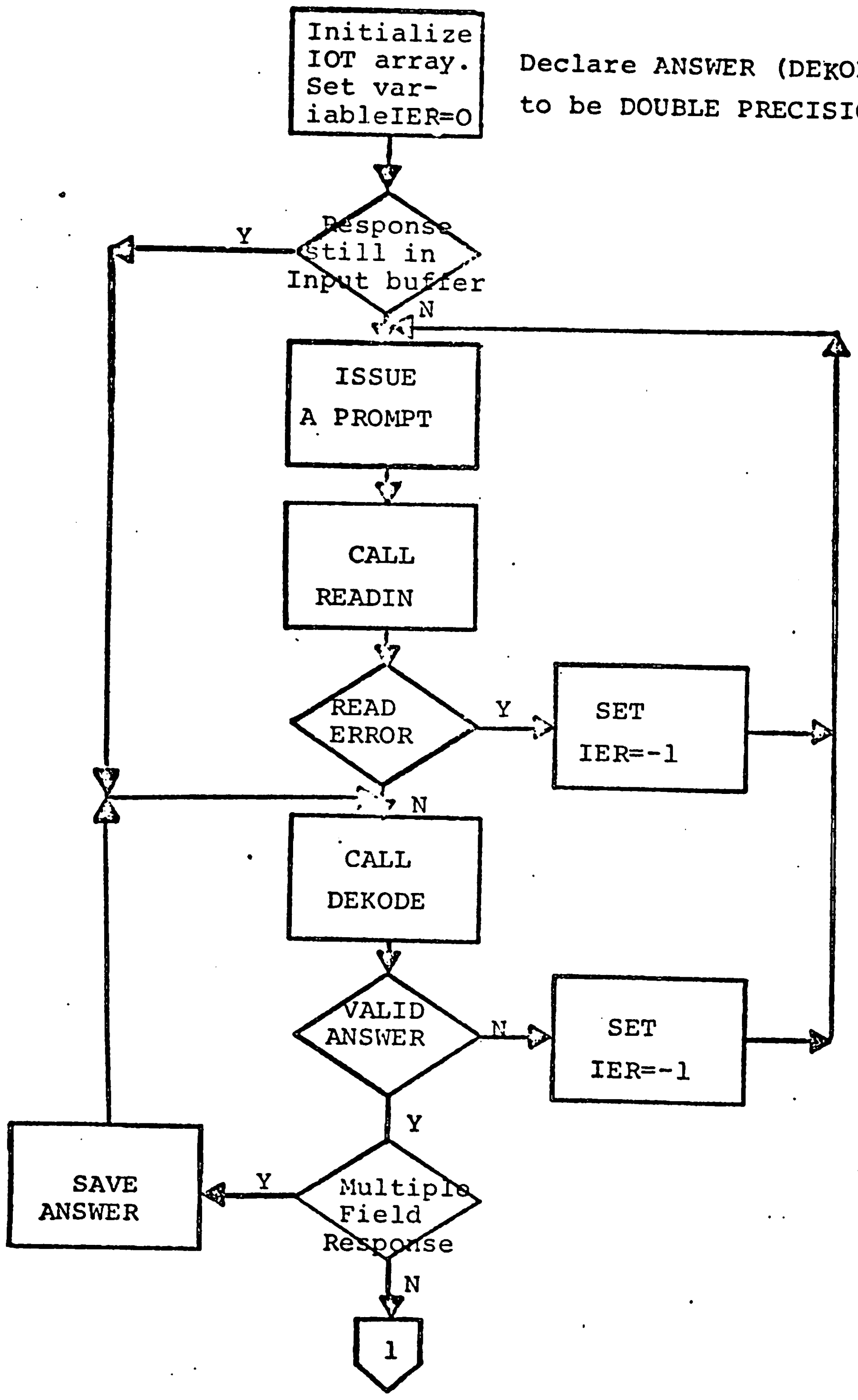
I. INTRODUCTION

The General I/O Subroutines were designed for and implimented on the HP S/3000.

There are machine-dependent characteristics of these subroutines, such as, the use of CHARACTER declarations in FORTRAN and the use of SPL (Systems Programming Language for HP S/3000).

Conversion to another system would be extensive because of the use of SPL and procedures intrinsic to HP S/3000 Operating System. The time required to do such a conversion is certainly dependent upon the installation, but would require somewhere in the neighborhood of three to four man-weeks.

V. PROGRAMME ORGANIZATION



Declare ANSWER (DEKODE parameter) to be DOUBLE PRECISION.

III. INPUT REQUIREMENTS

1. To allow user to enter more than one response during one input operation.

2. To provide a continuation facility to allow a user to enter input over one record size.

3. To establish a meaning for several key symbols; such as, END, EXIT, etc.

4. To provide the facility for interpreting the symbols YES, NO, PROMPT LONG and PROMPT SHORT.

5. To provide interactive edit capabilities for invalid input corrections.

IV. OUTPUT REQUIREMENTS

1. To provide capabilities in moving characters from one buffer to another; from and to specified locations.

2. To facilitate character repetition when building an output buffer.

3. To provide spacing control when writing a buffer to a specified file.

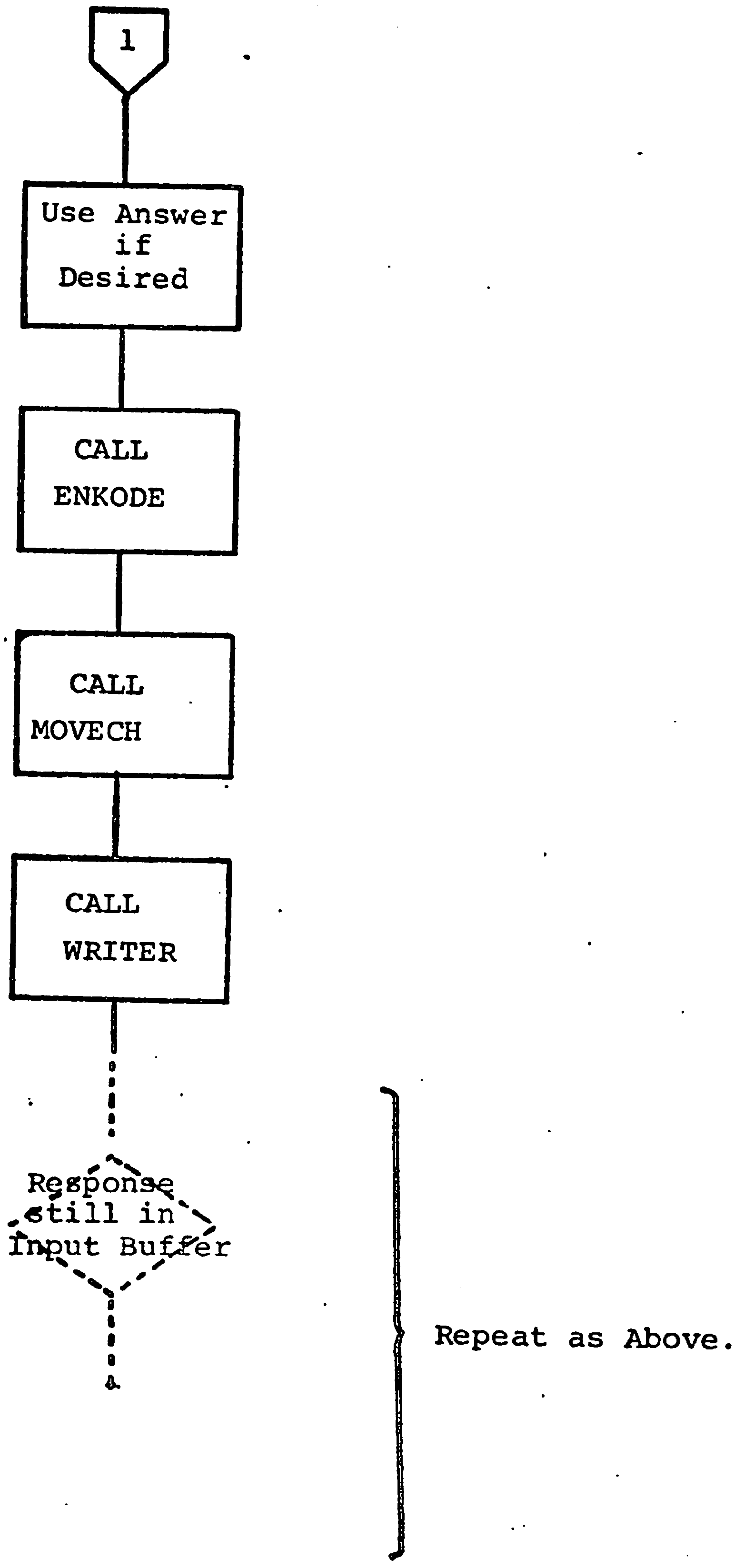
VI. L I M I T A T I O N S   A N D   A S S U M P T I O N S

1. When correcting an invalid response, the user may use a maximum of 10 correction instructions, separated by blanks, at one time.

2. Double integer capabilities are not present in FORTRAN, but a set of ASCII characters can be converted to an internal double integer and back to ASCII again, using DEKODE and ENKODE respectively. The value returned from DEKODE must not be used in FORTRAN but can be passed directly to ENKODE.

3. Input records and output records are always assumed to be of character type.

4. Declare ANSWER (DEKODE parameter) to be DOUBLE PRECISION.


1
Use Answer if Desired
CALL ENKODE
CALL MOVECH
CALL WRITER
Response still in Input Buffer
Repeat as Above.

PROGRAMME NAME:	READIN
FUNCTION:	To read a single or multiple line of ASCII characters from a specified input file into an input buffer. To interpret and execute the field correction instructions, namely: MOD, I, D, R, "CR". (See Note:)
FORTRAN CALL:	CALL READIN (IOT, INBUFF, IER)
ARGUMENT DESCRIPTION:	
INPUT..IOT......	I/O table 12 elements long containing the following information:
IOT(1)...	Input file number.
IOT(2)...	Output file number.
IOT(3)...	Input record length. (Indicates number of characters)
IOT(4)...	Output record length. (Indicates number of characters)
IOT(5)...	Input buffer size, i.e., the dimensions of INBUFF, a character array.
IOT(6)...	Count of the number of remaining responses.
IOT(7)...	Pointer to the next field in the input buffer.
IOT(8)...	Number of characters in the input buffer.
IOT(9)...	=0 to not interpret "YES" and "NO" as special responses. =1 to map "YES" into 1 and "NO" into 2.
IOT(10)..	=0 to indicate that long prompts are to be issued. =1 to indicate that short prompts are to be issued.
IOT(11)..	=0 means blanks are delimeters and commas are treated as alpha characters. =1 means commas are delimeters and blanks are treated as alpha characters except preceeding blanks in a field are ignored.
IOT(12)..	=0 means have processed an entire response. >0 means have not processed an entire response. There is at least one field yet to be decoded.
OUTPUT..INBUFF......	The input buffer of ASCII characters

VII. SUBROUTINE DESCRIPTION

PROGRAMME NAME:	DEKODE
FUNCTION:	To decode the current field in the input buffer, INBUFF, into a specified internal (binary) representation. The conversion is performed on integer and real numbers only. If the string to be decoded is not integer or real, a flag is returned indicating an alphanumeric response has been detected (See Note 1:) except as follows: 1. If "YES" or "NO" is detected and IOT(9) indicates that interpreting is to be done, then 1 or 2 is returned respectively. 2. If "PROMPT LONG" or "PROMPT SHORT" is detected, then IOT(10) is set to 0 or 1 respectively. 3. Comments are enclosed by <<.....>> and are ignored.
FORTRAN CALL:	CALL DEKODE (IOT, INBUFF, ANSWER, FTYP)
ARGUMENT DESCRIPTION:	
INPUT..IOT......	I/O table described in READIN.
INBUFF....	The input buffer of ASCII characters to be decoded.
FTYP.....	Indicates the type of internal representation desired: =0 signifies alphanumeric type. =1 signifies single word integer. =2 signifies double word integer. =3 signifies floating point number. =4 signifies double precision floating point number.
OUTPUT..ANSWER..	Word or words containing the decoded number or pointer to first character of alphanumeric field in INBUFF. Must declare ANSWER as DOUBLE PRECISION and equivalence it to a single integer and a real

106

	read in with unnecessary (2 or more consecutive) blanks suppressed.
IER..	=0 if valid read.
	=1 if End of Data encountered.
	=-1 if physical end of file reached.
	=-2 if buffer size exceeded.

<u>NOTE</u>:

For an interactive user, special editing instructions may be used to correct invalid responses.

Upon detecting an invalid response, the calling programme may issue an errxor message, the prompt at which the invalid response occurred and call READIN.

At this time, the user may enter the MOD instruction which means modify. Upon detecting the MOD instruction, the incorrect response is displayed, so that, the user may indicate the character(s) to be changed through three other instructions, namely:

Ichar	-	Insert character(s) after this point.
Dchar	-	Delete character at this point.
Rchar	-	Replace character(s) at this point.

More than one of the above three instructions may be entered on the same line or on different lines to correct more than one part of the invalid response.

To leave the modify mode, the user has only to enter a carriage return, whereupon, control returns to the calling programme ready to enter the decoding routine.

The modify instructions do not alter any remaining responses already in the input buffer.

After an error has occurred and the MOD instruction is not used, then, the response entered will simply replace what was ever in the input buffer and decoding will resume.

After an error has occurred and the MOD instruction is used but the correction instructions are not, then, the response entered will replace whatever was in the input buffer and decoding continues.

Can have a maximum of 10 corrections at one time. If you have I's or R's then one blank is inserted.

PROGRAMME NAME:	ENKODE
FUNCTION:	To convert an internal (binary) number to ASCII characters according to a specified format.
FORTRAN CALL:	CALL ENKODE (VALUE, TYPE, KIND, WIDTH, DIGITS, STRING).
ARGUMENT DESCRIPTION:	
INPUT..VALUE....	The variable containing the number to be converted.
TYPE.....	The type of internal representation. =1 for integer. =2 for double integer. =3 for real. =4 for double precision real.
KIND.....	The kind of conversion desired. =0 signifies default, i.e., if single word integer use I6 format, if double word integer use I12 format, if floating point use Gw.d format. =1 signifies Iw. =2 signifies Fw.d. =3 signifies Ew.d. =4 signifies Dw.d. =5 signifies Nw.d. =6 signifies Mw.d.
WIDTH....	Width of entire ASCII string including all special characters.
DIGITS...	Number of fractional digits desired.
OUTPUT.STRING...	Output array containing the ASCII characters

<u>NOTE</u>:

If an error occurs, such as WIDTH too small for result specified by KIND, if DIGITS <0 or WIDTH ≤ 0, then, the output STRING will contain #'s.

117

DEKODE ----- Continued

OUTPUT..FTYP.......	Type of conversion performed if any.
	<0 indicates the number of character. within the alphanumeric field encountered.
	=0, for Blank field found. i.e. No characters in response.
	=1 for single word integer.
	=2 for double word integer.
	=3 for floating point number.
	=4 for double precision floating point number.
	=11 signifies "HELP" detected.
	=12 signifies "END" detected.
	=13 signifies "EXIT" detected.
	=14 signifies "PAUSE" detected.

NOTE 1:

If an alphanumeric response (not real or integer) has been detected then ANSWER is a single word integer which points to the first character of this response in INBUFF and the absolute value of FTYP specifies the number of characters in this response. Because of this, one must equivalence ANSWER to an integer variable and use the integer variable when FTYP <0.

NOTE 2:

A response may contain one or more fields delimited by blanks or commas. Each field is decoded one at a time, returning the decoded result and type specified. When the end of a response (last field decoded) is reached, IOT(12) is set to 0.

PROGRAMME NAME:	WRITER
FUNCTION:	To write the ASCII characters in the array OUTLIN from a specified starting and ending position.
FORTRAN CALL:	CALL WRITER (IOT, OUTLIN, IBEG, IEND, CARR).

ARGUMENT DESCRIPTION:

INPUT..IOT.......	The I/O table described in READIN.
OUTLIN....	The character array containing the characters to be written to the specified file.
IBEG......	Indicates the starting position in OUTLIN to begin writing.
IEND......	Indicates the ending position in OUTLIN for writing.
CARR......	Carriage control indicator. (See carriage control codes).

SPECIAL NOTE:

If IEND is negative, the character array OUTLIN is scanned up to the last non-blank character and written to the output file starting from OUTLIN(1).

At any time, if there are more characters in OUTLIN, the output record size will hold, an automatic skip to the next line is performed and the remaining characters are written to the output file with single spacing.

C A R R I A G E   C O N T R O L   C O D E S

INTEGER CODE	MEANING
. ↑	
.	
.	
3	- triple spacing
2	- double spacing
1	- single spacing
0	- no spacing
-1	- skip 1 page
-2	- skip 2 pages
-3	- skip 3 pages
.	
.	
. ↓	

108

PROGRAMME NAME:	MOVECH
FUNCTION:	To move a string of ASCII characters from array STRING to array OUTLIN from and to a specified location. Also, to provide for character repetition in OUTLIN.
FORTRAN CALL:	CALL MOVECH (STRING, IBEG, NCHAR, OUTLIN).
ARGUMENT DESCRIPTION:	
INPUT..STRING....	Character array containing the ASCII characters to be moved. First character in location 1.
IBEG......	>0 signifies transfer from STRING(1) to OUTLIN (IBEG) going from left to right. <0 signifies transfer from STRING (NCHAR) to OUTLIN (IBEG) going from right to left.
NCHAR.....	Character counter indicator. ≥ 0 signifies the number of characters in STRING starting at location 1. <0 means repeat the character found in STRING(1) the absolute value of NCHAR times.
OUTPUT..OUTLIN...	Character array receiving the string of characters.

APPENDIX B:

KEY SYMBOLS AND MEANING

SYMBOL		MEANING
$	-	Continuation line character (last character on the line).
&	-	Delimiter between responses in a multiple response.
" " or ","	-	Delimiters between fields in a response. Two consecutive commas signify a null or unknown field in a response. May use one or the other as delimiters, not both. (See IOT(11) description)
END	-	To signify the end of the current minor level of inquiry.
EXIT	-	To indicate that control is to return to the previous major level of inquiry.
PAUSE	-	To indicate a break during interaction. The calling programme is to issue messages as to what information to retain and then release the system. The system can be re-entered at this same point at a later time.
HELP	-	Signifies to the calling programme that an information block is to be accessed and information extracted to aid the user during this interaction.
PROMPT LONG	-	Sets IOT(10) to 0.
PROMPT SHORT	-	Sets IOT(10) to 1.

APPENDIX A:

FIELD CORRECTION INSTRUCTIONS

INSTRUCTION		MEANING
MOD	-	To indicate to the I/O Routines that correction instructions are to follow so the last response is re-written to the interactive device.
Icccc...c	-	To insert one or more contiguous characters at the location after the I. If a blank is to be inserted, only one at a time can be inserted.
D	-	To delete the character directly above the D.
R	-	To replace the character(s) directly above the R.
	-	If a blank is to be the character, only one blank at a time can be done.
"CR"	-	To leave the MODify mode and return to the calling programme.
$	-	To skip the number of characters in the buffer specified by IOT(3).

NOTE:

Allowed a maximum of 10 correction instructions at one time.

PLOTTING ROUTINES

THE PLOTTING ROUTINES PLOTPT AND OUTPLT, AVAILABLE ON CDC/6400 HAVE BEEN MODIFIED FOR USE ON HP/3000.

THESE ROUTINES ENABLE A ROUGH GRAPH TO BE PLOTTED BETWEEN PAIRS OF (X,Y) VALUES. EACH PLOT OCCUPIES ONE LINE-PRINTER PAGE (132 CHARACTERS WIDE) A LINE AT THE BOTTOM IS AVAILABLE FOR A HEADING. UP TO 700 POINTS CAN BE PLOTTED ON EACH PLOT. THE SCALE IS CHOSEN AUTOMATICALLY TO BE REASONABLY "NICE", BUT THE USER CAN SPECIFY HIS OWN SCALE BY MEANS OF A CALL TO THE SCALE SUBROUTINE.

A DESCRIPTION OF EACH OF THE ROUTINES IN THE PLOT PACKAGE FOLLOWS:

A) THE FIRST STEP IS TO INITIALIZE THE PLOTTING MATRIX BY THE CALL

```
CALL INITPLT(N)
```

WHERE N IS THE FORTRAN LOGICAL DEVICE NUMBER ON WHICH THE PLOT IS TO APPEAR.

B) TO PLACE THE COORDINATES (X,Y) OF A POINT IN THE CURRENT PLOT TOGETHER WITH CODE-NUMBER OF THE CHARACTER TO BE PLOTTED, WE EXECUTE

```
CALL PLOTPT(X,Y, CODE-NUMBER)
```

X AND Y MUST BE *real* QUANTITIES (OR EXPRESSIONS) GIVING THE COORDINATES AND THE "CODE-NUMBER" IS AN INTEGER THAT CORRESPONDS TO CHARACTER AS TABULATED ON THE NEXT PAGE.

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	PLOT ROUTINES
FUNCTION	To plot a graph between two variables on line-printer.
AUTHOR	Converted to HP/3000 by K. Ahmed
LANGUAGE	FORTRAN

DESCRIPTION

The package consists of the following subroutines:

INITPLT (n)	- initializes plot matrix; n= logical device no. for output file
PLOTPT(x,y,m)	- stores the point (x,y) in the plotting matrix; m is an integer code for the plot-character to be used.
OUTPLT	- when all the points have been stored, a call to OUTPLT prints out the graph on the specified file (logical device #n,

SCALE (xmin, xmax, ymin, ymax), specifies the scale of plot [optional]

SPECIAL CONSIDERATIONS

Upto 700 points can be plotted on the graph. If SCALE is not specified, a suitable scale is automatically chosen.

A SIMPLE EXAMPLE (THE RESULTING PLOT APPEARS ON THE NEXT PAGE)

```
:JOB username.accountname
:FORTRAN
$CONTROL FILE=5,FILE=6,LABEL,MAP
          PROGRAM TST
          CALL INITPLT (6)
          DO 1 I=1,50
          X=I-1
          Y=SIN(0.1*X)
          CALL PLOTPT(X,Y,4)
          CALL PLOTPT(X,COS(0.1*X),46)
1         CONTINUE
          CALL OUTPLT
          WRITE (6,10)
10        FORMAT (1X,'SIN/COS')
          STOP
          END
:EOD
:PREPRUN $OLDPASS,$NEWPASS;RL=PLOTRL.PUB.LIB
:EOJ
```

c) IF WE WISH TO CHOOSE THE SCALES OF THE PLOT, WE EXECUTE (*optional*)

```
CALL SCALE (XMIN, XMAX, YMIN, YMAX)
```

POINTS (X,Y) WILL NOW BE PLOTTED ONLY IF

$XMIN \leqslant X \leqslant XMAX, \quad YMIN \leqslant Y < YMAX$

D) WHEN ALL THE POINTS HAVE BEEN PLACED WE CAN CAUSE THE PLOT TO BE PRINTED BY EXECUTING

```
CALL OUTPLT
```

AT THE END OF THIS CALL THE PLOTTING MATRIX IS AUTOMATICALLY RE-INITIALIZED AND THE SCALE IS RESET TO THE DEFAULT OPTION.

E) IN ORDER TO USE THESE SUBROUTINES, THE USER MUST SPECIFY THE PLOT LIBRARY IN THE PREP (OR PREPRUN) COMMAND, E.

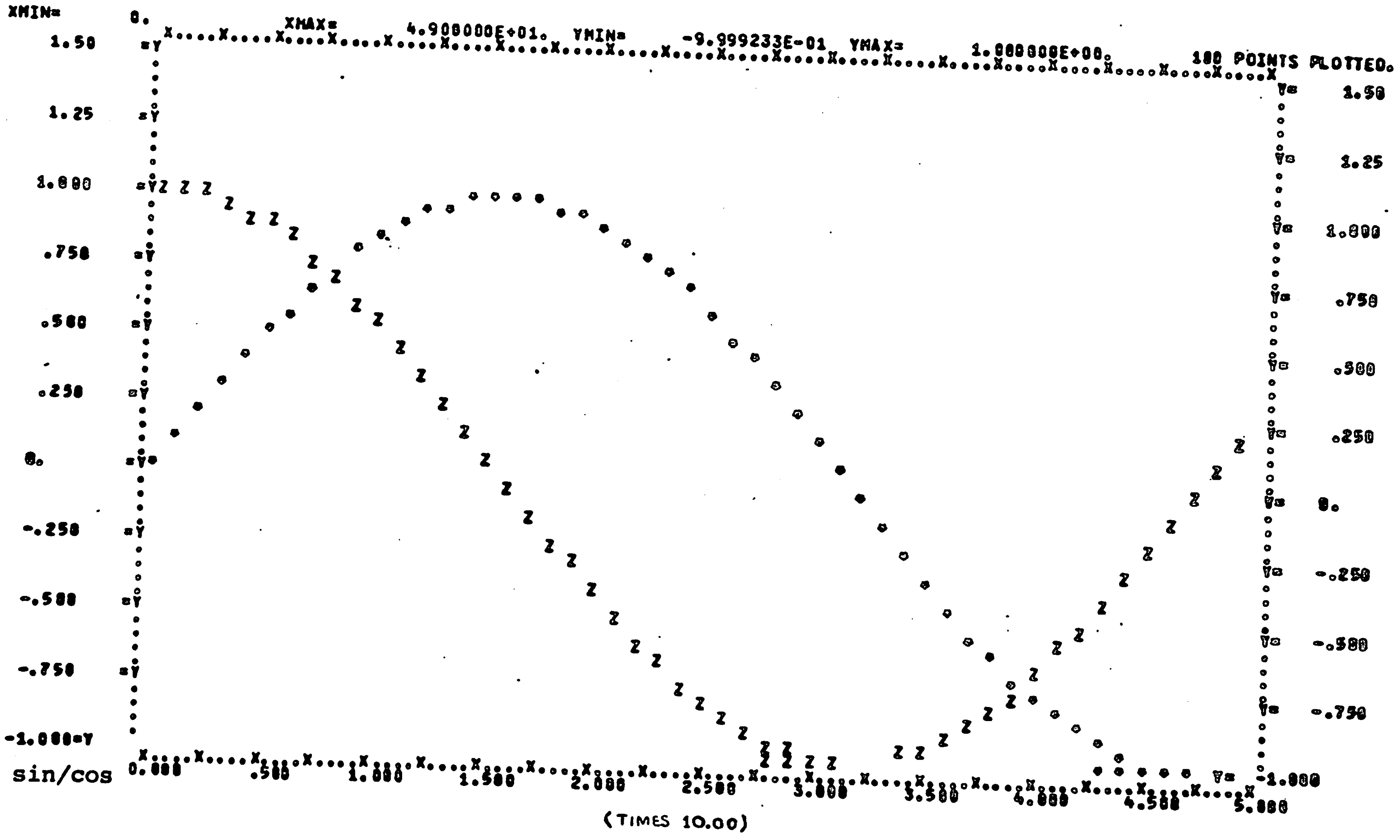
```
:PREP $OLDPASS, $NEWPASS;RL=PLOTRL.PUB.LIB
```

Table of Character Values

Character	Value	Character	Value	Character	Value	Character	Value
(blank)	0	0	10	A	21	N	34
=	1	1	11	B	22	Ø	35
+	2	2	12	C	23	P	36
-	3	3	13	D	24	Q	37
*	4	4	14	E	25	R	38
/	5	5	15	F	26	S	39
(	6	6	16	G	27	T	40
)	7	7	17	H	28	U	41
,	8	8	18	I	29	V	42
.	9	9	19	J	30	W	43
		$	20	K	31	X	44
				L	32	Y	45
				M	33	Z	46

III. STATISTICAL

112


XMIN= 0. XMAX= 4.900000E+01. YMIN= -9.999233E-01 YMAX= 1.000000E+00. 100 POINTS PLOTTED.
1.50
1.25
1.000
.750
.500
.250
0.
-.250
-.500
-.750
-1.000
sin/cos
0.000
.500
1.000
1.500
2.000
2.500
3.000
3.500
4.000
4.500
5.000
(TIMES 10.00)

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	CCSS (Conversational Computer Statistical System)
FUNCTION	Provides basic descriptive statistical procedures with subsetting capability.
AUTHOR	R. Kronmal
LANGUAGE	FORTRAN

DESCRIPTION

This interactive system consists of a set of programs for managing and analysing data sets with miltiple record types. It requires no computer programming experience to use.

The system provides options for:

1. Code sheet entry
2. Clean data file
3. Put data into file
4. Tabling
5. Listing
6. Scattergram plot
7. Summary
8. Cumulative density plot
9. Bar graph

SPECIAL CONSIDERATIONS

The system requires the interim spooling mechanisms as explained in CARD and PRINTER.

GROWTH/CCSS

114

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE CHISQ

FUNCTION Analyses 2-way contingency tables

AUTHOR C. Goldsmith/P. Balnys

LANGUAGE FORTRAN

DESCRIPTION

This is an interactive program providing the following options for analysing 2-way contingency tables with up to 20 rows and up to 20 columns.

1) Data input
2) Printout of table
3) Expected values, chi-square contributions and percentages
4) Chi-square
5) Fisher-Irwin Exact Test (for 2 x 2 tables)
6) Tests of Symmetry and Agreement (for square tables)
7) Comparison of Columns
8) Correlation Analyses
9) Ability to analyse another table

SPECIAL CONSIDERATIONS

This program requires the generalized I/O routines (GENRLIO)

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE TRAND

FUNCTION generates uniform random numbers between 0.0 and 1.0

AUTHOR

LANGUAGE SPL

DESCRIPTION

TRAND is a subroutine with calling sequence

CALL TRAND (Y,R).

Y is an output parameter containing the random number

R is the seed for the random number generator

R is set for the first call to TRAND and returns as the seed for the next call to TRAND.

SPECIAL CONSIDERATIONS

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	SAMPLE
FUNCTION	Selects random sample groups from a given population
AUTHOR	J.W. Bush
LANGUAGE	FORTRAN

DESCRIPTION

This program is run interactively. The user enters the size of the population from which the sampling to be done, and the size of the samples required. He then enters a number between 0 and 1 for the random number generator (TRAND). The selected samples are printed out in sorted order.

SPECIAL CONSIDERATIONS

required subroutines: RSMPL
TRAND
SHUFL
SORT

limitations: the combined total of the samples must be less than 5000.

IV. EDUCATION

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE MACMAN

FUNCTION Simulation of Blood circulation in Human Body

AUTHOR C.J. Dickinson

LANGUAGE FORTRAN

DESCRIPTION

"MACMAN" is a synthetic person who has a heart inside a chest, systemic arteries and arterioles, a capillary bed, and veins collecting blood from the capillary bed and returning it to the heart. "MACMAN" thus has a complete systemic circulatory system, and when the heart is working it will circulate blood. To speed up computation, the heart is treated as a single chamber filling the right atrium and pumping blood out into the aorta. The pulmonary circulation is regarded as simply a parallel path, and not (as in life) in series with the systemic circulation. However, this makes the model unrealistic only when one side of the heart is able to pump much less than the other (e.g. because of valve disease). "MACMAN" cannot therefore simulate the effects of valve lesions but it can simulate most types of generalized heart disease. "MACMAN" also possesses synthetic baro-receptors similar in operation to these which are normally situated at the bifurcation of the common carotid artery and at the aortic arch. These act in such a way as to stabilize blood pressure.

SPECIAL CONSIDERATIONS

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	MACPUF
FUNCTION	Simulation of Respiratory System
AUTHOR	C.J. Dickinson
LANGUAGE	FORTRAN

DESCRIPTION

"MACPUF" is a computer model of the lungs and airways, the pulmonary and systemic circulation, the tissues and the brain which is designed to study gas transport between lungs and tissues, the control of ventilation, hydrogen in regulation, and complex interactions between them.

SPECIAL CONSIDERATIONS

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	MACPEE
FUNCTION	Simulation of Kidney and Body Fluids
AUTHOR	C. J. Dickinson
LANGUAGE	FORTRAN

DESCRIPTION

"MACPEE" is a digital computer model of the systemic circulation and kidneys designed to study the circulation and the kidneys together. The heart and systemic circulation are virtually identical with those in "MACMAN" and initial values for systemic arterial and venous resistances, cardiac contractility, blood volume and blood viscosity, and baroreceptor stabilizing function are in general the same as in "MACMAN"; but "MACPEE" operates over periods of days whereas "MACMAN" operates over minutes. "MACPEE" is a very much more complex model since it incorporates not only the whole of "MACMAN" but also has:-

1) kidneys, excreting water, urea, sodium and potassium

2) a gut absorbing water, protein and electrolytes

3) thirst, governing water control in accordance with the body's needs

4) an endocrine system with individual control of the secretion rates of vasopressin (antidiuretic hormone), aldosterone, and angiotensin.

SPECIAL CONSIDERATIONS

McMASTER UNIVERSITY

COMPUTATION SERVICES UNIT

HEALTH SCIENCES CENTRE

HAMILTON, ONTARIO, CANADA

L8S 4J9

TITLE	CPR (Cardio-pulmonary Resuscitation)
FUNCTION	A question-answer dialogue to take the student through steps of Cardio-pulmonary Resuscitation.
AUTHOR	Dr. E. Hoffer (Mass. Gen. Hosp.) adapted to H.P./3000 by K. Ahmed.
LANGUAGE	FORTRAN

DESCRIPTION

A Fortran driver program is used to take the student through a text file (consisting of card-image records) through multiple choice question-answer sequences. Usually the questions are repeated until the correct answer is typed; in some cases explanation and instructions are provided.

The program also refers the student to a booklet containing Electro-cardiogram traces.

SPECIAL CONSIDERATIONS

The driver program is fairly general and could be used on other text files of similar nature.

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE THYROID

FUNCTION Refer to description

AUTHOR G.D. Anderson

LANGUAGE fortran

DESCRIPTION

The THYROID Programme estimates the probability that a patient has underactive, normal, or overactive thyroid function based on symptoms observed on the patient. The probabilities calculated by this programme are based on data from 879 patients with thyroid disease.

SPECIAL CONSIDERATIONS

McMASTER UNIVERSITY
COMPUTATION SERVICES UNIT
HEALTH SCIENCES CENTRE
HAMILTON, ONTARIO, CANADA
L8S 4J9

TITLE	BONETUMR
FUNCTION	Refer to description
AUTHOR	G. D. Anderson
LANGUAGE	FORTRAN

DESCRIPTION

The BONE TUMOR Programme is an example of the use of elementary probability and an expression called Bayes Theorem to estimate the relative probabilities for several types of Bone Tumor given specific radiological observations on a patient with a Bone Tumor present. The probability estimates made by this Programme are based on information from a large number of patients with primary Bone Tumors which is stored within the Programme.

SPECIAL CONSIDERATIONS

McMASTER UNIVERSITY

COMPUTATION SERVICES UNIT

HEALTH SCIENCES CENTRE

HAMILTON, ONTARIO, CANADA

L8S 4J9

TITLE	BPFILM
FUNCTION	Refer to description
AUTHOR	C. Goldsmith
LANGUAGE	FORTRAN

DESCRIPTION

This program will help you to analyse your own "OBSERVER" variation in the clinical measurement of Blood Pressure. The pre-requisite for running this program consists of going through Film #MP 21 entitled "Blood Pressure Readings" and writing down your Blood Pressure measurements for each of the 14 patients.

SPECIAL CONSIDERATIONS

V. USER TRAINING

FORTRAN

McMASTER UNIVERSITY MEDICAL SCHOOL
HAMILTON, ONTARIO

123

F O R T R A N

Introduction

The FORTRAN/3000 Language is based upon the ANSI Standard FORTRAN. In addition, FORTRAN/3000 has many extensions which expand the capabilities and increase the power of the language.

Example of a FORTRAN Program

Following is an example of a FORTRAN Source Program, illustrating a few of the extensions to the language. These are indicated by bracketted numbers across from a particular line and are explained below the example.

EXAMPLE:

```
1)     $CONTROL LABEL, MAP, FILE=4
             PROGRAM TEST
             DIMENSION ID(10)
2)           CHARACTER  X( 5)
             I=4
3)      5    READ (I,10,END=30)ID,X
             WRITE(6,20)ID(1), ID,X
       10    FORMAT(10I5,5A1)
4)           WRITE(8@ID(1),10)ID,X
5)     20    FORMAT(1X,"RECORD NUMBER",
             I6,/,1X,10I6,5(1X,A1))
             IF(ID(1).LT.10)GO TO 5
       30    STOP
             END
```

Special Features of FORTRAN/3000

1) The $CONTROL Command indicates list and compilation options.

2) Arrays and variables of character type may be declared. Each element in the array occupies one-half of a 16-bit word.

3) In the read statement there is an action label reference i.e. upon detection of an end-of-file control will branch to the statement labelled with the number 30.

4) In this write statement, there is reference to a direct-access file. The value of the arithmetic expression following the "@" character, indicates the record number.

FORTRAN

5) In addition to the "nH" representation for outputting Hollerith characters, the characters may simply be enclosed in quotes.

*The following pages describe further, the special features of HP S/3000 FORTRAN. For more detailed descriptions, please refer to the pages in the FORTRAN Reference Manual as indicated on the right hand side in brackets.

ANSI Standard FORTRAN.

HP S/3000 FORTRAN conforms to the American National Standard Institute's Standard for FORTRAN.

Non-Standard Features. (FORTRAN B-1, B-4)

There are some restrictions and extensions on S/3000 FORTRAN compared to ANSI FORTRAN, of which the most important are:

1) Character type data may be used.

2) An array may have up to 255 dimensions.

3) Direct-access files may be referenced.

CDC 6400 vs HP S/3000.

There are a few differences between HP S/3000 FORTRAN and CDC 6400 FORTRAN, such as:

1) The character *, in CDC FORTRAN, is used in format statements to signify Hollerith fields but in HP S/3000 FORTRAN, the " must be used.
2) In CDC FORTRAN, file specifications are indicated on the Program Card and compiler options on the FTN Card, whereas, in HP FORTRAN, the file specifications and compiler options are indicated on the $CONTROL Card before the Source Program.
3) HP FORTRAN allows the Program Card to be optional, i.e., it may or may not be included.
4) The word sizes differ between the two Computers. CDC 6400 uses a 60 bit word, while HP S/3000 uses a 16 bit word. The smaller word size restricts the accuracy and the absolute value which may be stored in a word.

Word Size

Accuracy (FORTRAN 2-17)

<u>NOTE</u>: The largest and smallest values a <u>simple integer may have in HP S/3000</u> are respectively, 32767 and -32768.

$CONTROL Card (FORTRAN 11-2, 11-4)

There is a FORTRAN statement which precedes each main or subprogram unit and defaults if omitted. It begins with the characters $CONTROL starting in column one. Following the letters $CONTROL are various key words separated by commas, indicating list and compilation options of which the most useful are:

1) LABEL - this causes the compiler to emit a list of labels and their respective addresses after each compiled program unit. This is useful in tracing back execution errors.

$CONTROL LABEL

2) MAP - this parameter sends a symbol table dump to the list-file after each program unit. This listing gives the type of all variables and their relative addresses in the program unit.

$CONTROL MAP

3) FILE=*integer* - this parameter is required for files whose FORTRAN file numbers do not explicitly appear in the I/O statement, such as READ (I,10) ID,X where I has an assigned value.

$CONTROL FILE=*integer*

e.g. $CONTROL LABEL, MAP,
FILE=4

The CHARACTER declaration is a Type Statement indicating that the variables and arrays following the declaration are of type character. The length of character symbolic names can be specified in two ways:

CHARACTER
(FORTRAN 4-16)

1) Through the length attribute following the CHARACTER heading (*x).

e.g. CHARACTER*80 BUFF1

2) Through individual length attributes following character symbolic names.

e.g. CHARACTER BUFF2(80)

In the first example, BUFF1 is referred to as a <u>character variable</u> with length 80. In the second example, BUFF2 is referred to as a <u>character array</u> with 80 elements.

Elements in the character array are referred to in the same way as in integer arrays.

e.g. BUFF2(9)="Z"

Elements in the character variable are referred to differently. The beginning element and number of consecutive elements must be specified.

e.g. BUFF1[3:5]="ABCDE"

This example indicates that the string, ABCDE, will be placed in the character variable BUFF1 starting at the third element.

Free Format (FORTRAN 9-43,9-45)

Free-field input/output is formatted conversion according to format and/or edit control characters imbedded in the data. The FORMAT Statement is not required. For free-field input/output, an * instead of a FORMAT Statement indentifier is used.

Input

e.g. READ(5,*)list elements

The data items on file 5 to be read are separated by data item delimiters which may be a comma, a blank space, or any ASCII character that is not a part of the data item.

Output (FORTRAN 9-46)

For free-field output, predefined field and edit descriptions are used.

End-of-file Test (FORTRAN 8-3)

Action Label References may be used in read or write statements. The most commonly used action label reference is in the form:

```
READ(file, format label,
     END=label)
```

If an end-of-file is encountered then control is transferred to the statement indicated by the label after END=, at which point the programmer decides upon the course of action.

e.g. READ(5,10,END=30)

NOTE: There is also an ERR=label reference for error conditions.

Two types of access to files on disc devices are available through the file system. These are sequential or direct.

Sequential Files
(FORTRAN 8-1,8-2,8-5,9-2)

In sequential access, as many records as necessary are used in sequence until the entire list of variables has been transmitted. After having read or written one record on a sequential file, the record pointer is automatically moved to the next record.

Direct-access Files
(FORTRAN 8-1,8-2,8-5,9-2)

Direct-access features in FORTRAN allow the access of any record in a file at random. The form of a direct-access read is:

```
        READ (file@record,10)list elements
```

where file is a positive integer constant or variable indicating the file unit number and record is an arithmetic expression, constant or integer variable, the value of which indicates a specific record.

```
        e.g.  WRITE(8@ID(1),10)ID,X
```

This will cause the values of ID and X to be written to file 8 in the record specified by the value of ID(1) and according to the format referenced by label 10.

Equating logical unit numbers with actual file names
(Section II-JCL)

When reading or writing to other than standard input or output files (files 5 and 6), a file equate must be done before executing the program.

The name for a FORTRAN logical file is created by concatenating the word FTN with the two digit representation of the integer name.

```
        e.g.  file 8 is FTN08
```

This name, FTN08, is the Formal File Designator.

File Command

To be able to read or write on an existing permanent file or temporary file, a correspondence between the name of the file, i.e. the Actual File Designator, and the Formal File Designator must be established. The files are equated using the File Command.

Permanent Files

For example, there is an existing permanent file by the name of *XXX* and the programmer wishes to read and/or write on this file through file unit number 3.

Before running the program, the following file equate is performed:

```
:FILE FTN03=XXX,OLD
```

In some cases, a temporary file is required meaning records do not have to be permanently saved. An example of this would be writing card images to a file and then reading them back as required. A typical file equate for a temporary file is:

Temporary Files

```
:FILE FTN02;REC=-80,,,ASCII
```

NOTE: An Actual File Designator was not required since a permanent file was not referenced. All records in this file will be lost upon completion of the program.

@. USER. SUPPØRT

PRØGRAM	SØURCE	
ACUMLØG	SACUMLØG	
	SCØSTING	- subroutine
	SDATE	- subroutine
BØNETUMR	SBØNETMR	
CARD	SCARD	
CCSS	SCCSS	
	SASØRT	- subroutine
	SCLEAR	- "
	SGP1	- "
	SGP2	- "
	SGP3	- "
	SGP4	- "
	SGPLØT	- "
	SLISTS	- "
	SRRCD	- "
	STMN1	- "
	STRAM	- "
	SBARG	- "
SL (CDC Conversion Program) (SL file)	CDCCØNV	- (ASCII USE FI
	SCØRRECT	- subroutine
CCSS TEST FILES	CGRØWTH	
	DGRØWTH	

PROGRAM	SOURCE
CHISQ	SCHISQ
CHITREND	SCHITRND
	SPROB - subroutine
	STREND - subroutine
CPR	SCPR
	CPRDATA - data file
GENRLIO (CRL file)	SGENIO
KSAMP	
MACMAN	SMACMAN
MACPEE	SMACPEE
MACPUF	SMACPUF
MTHLOG	SMTHLOG
	SDATE -subroutine
PLOTRL (CRL file)	SPLOTPT
PRINTER	SPRINTER
SAMPLE	

PRØGRAM	SØURCE
	SBUSY - (subroutine BUSYTEST)
THYRØID	STHYRØID
SUMLØG	SSUMLØG
	TRAND - subroutine.