DOCUMENT RESUME

ED 105 197 CE 003 485

AUTHOR	Weissmuller, Johnny J.; And Others
TITLE	CODAP: Programmer Notes for the Subroutine Library on the Univac 1108.
INSTITUTION	Air Force Human Resources Lab., Lackland AFB, Tex. Computational Sciences Div.
REPORT NO	AFHRL-TR-74-85
PUB DATE	Oct 74
NOTE	183p.; For related volumes, see CE 003 486-487
EDRS PRICE	MF-$0.76 HC-$9.51 PLUS POSTAGE
DESCRIPTORS	Career Ladders; Comprehensive Programs; *Computer Programs; Informaticn Processing; *Information Retrieval; Information Services; Information Sources; *Information Systems; Job Analysis; Occupational Clusters; *Occupational Information; Occupational Surveys; Task Analysis; *Technical Reports
IDENTIFIERS	Air Force; CODAP; Comprehensive Occupational Data Analysis Programs; UNIVAC 1108

ABSTRACT

The Comprehensive Occupational Data Analysis Programs (CODAP) package is a highly interactive and efficient system of computer routines for analyzing, organizing, and reporting occupational information. Since its inception in 1960, CODAP has grown in tandem with advances in job analysis methodology and is now capable of answering most of the wide variety of management questions which confront CODAP users. The documentation of the Univac 1108 CODAP system is being published in a series of three technical reports covering the control card and programing aspects of the system. The document contains programer notes on 100 library subroutines used by the current Univac 1108 version of CODAP. After a table of entry points, the write-ups appear in alphabetical order on subroutine name. Each write-up includes a summary of subroutine functions and a list of entry points. For each entry point, a calling sequence with descriptions of input and output arguments is provided. Comments on peculiar requirements for conversion and use of the subroutine close each write-up. (Author)

AFHRL-TR-74-85

AIR FORCE

HUMAN RESOURCES

CODAP:

PROGRAMMER NOTES FOR THE SUBROUTINE LIBRARY ON THE UNIVAC 1108

By

Johnny J. Weissmuller, Sgt, USAF
Bruce B. Barton, A1C, USAF
C. R. Rogers

COMPUTATIONAL SCIENCES DIVISION
Lackland Air Force Base, Texas 78236

October 1974
Interim Report for Period January 1973 – September 1974

Approved for public release; distribution unlimited.

LABORATORY

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235

NOTICE

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This interim report was submitted by Computational Sciences Division, Air Force Human Resources Laboratory. Lackland Air Force Base, Texas 78236, under project 7734, with Hq Air Force Human Resources Laboratory (AFSC), Brooks Air Force Base, Texas 78235. Dr. Raymond E. Christal, Chief of Occupational Research Division (OR), was project monitor, and Mr. William Phalen, OR, was the work unit scientist.

This report has been reviewed and cleared for open publication and/or public release by the appropriate Office of Information (OI) in accordance with AFR 190-17 and DoDD 5230.9. There is no objection to unlimited distribution of this report to the public at large, or by DDC to the National Technical Information Service (NTIS).

This technical report has been reviewed and is approved.

ROBERT A. BOTTENBERG, Chief
Computational Sciences Division

Approved for publication.

HAROLD E. FISCHER, Colonel, USAF
Commander

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER AFHRL-TR-74-85	2 GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CODAP: Programmer Notes for the Subroutine Library on the Univac 1108		5. TYPE OF REPORT & PERIOD COVERED Interim Jan 73-Sep 74
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Johnny J. Weissmuller Bruce B. Barton C. R. Rogers		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computational Sciences Division Air Force Human Resources Laboratory Lackland Air Force Base, Texas 78236		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62703F 77340116
11. CONTROLLING OFFICE NAME AND ADDRESS HQ, Air Force Human Resources Laboratory (AFSC) Brooks Air Force Base, Texas 78235		12. REPORT DATE October 1974
		13 NUMBER OF PAGES 180
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Offi.e)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION. DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Comprehensive Occupational Data Analysis Programs		
CODAP		Subroutines
Computer Programs	hierarchical clustering	FORTRAN
occupational survey	job descriptions	Assembly Language
task analysis	job types	work analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Comprehensive Occupational Data Analysis Programs (CODAP) package is a highly interactive and efficient system of computer routines for analyzing, organizing, and reporting occupational information. Since its inception in 1960, CODAP has grown in tandem with advances in job analysis methodology and is now capable of answering most of the wide variety of management questions which confront CODAP users. This documentation of the UNIVAC 1108 CODAP system is being published in a series of 3 technical reports covering the

(SEE REVERSE SIDE)

Block 20. ABSTRACT (Con't)

control card and programming aspects of the system. A fourth report is in preparation by the Occupational Research Division of AFHRL which covers the research and operational applications of the CODAP system. This document contains programmer notes on 100 library subroutines used by the current UNIVAC 1108 version of CODAP. After a table of entry points, the writeups appear in alphabetical order on subroutine name. Each writeup includes a summary of subroutine functions and a list of entry points. For each entry point, a calling sequence with descriptions of input and output arguments is provided. Comments on peculiar requirements for conversion and use of the subroutine close each writeup.

PREFACE

The authors wish to express their gratitude to all those who have contributed to the preparation of this series of 3 technical reports:

AFHRL-TR-74-83 CODAP: Source Program Listings

AFHRL-TR-74-84 CODAP: Control Card Specifications

AFHRL-TR-74-85 CODAP: Programmer Notes

In particular, the following people have played a vital role in the development of these documents: Dr. Raymond E. Christal, Chief of the Occupational Research Division of AFHRL, Mr. William Phalen, also of that Division, and of course the secretaries, Betty L. Brooks and Laurel J. Betz, who not only typed but also helped in proofreading.

FOREWARD

The Comprehensive Occupational Data Analysis Programs (CODAP) System has been under continuous development for the past fifteen years. In its current form, it is a joint product of the Computational Sciences Division and the Occupational Research Division of AFHRL. In general, the Occupational Research Division has developed input-output specifications for programs and program modifications, while the Computational Sciences Division has provided programming services. However, even this distinction has not always been a clear-cut one, since suggestions for program improvements have come from individuals in both Divisions.

Since its conception, development of the CODAP system has been under the general direction of Dr. Raymond E. Christal, Chief of the Occupational Research Division. Dr. Christal also personally designed many of the programs and program modifications.

The earliest CODAP programs were written by Mr. Daniel Rigney and Mr. Wayne E. Fisher. This version was later expanded and reprogrammed (under contract) for execution on an IBM 7040 by Mr. Irwin R. Oats and Mr. James R. Hills of the Computer Sciences Corporation and later of Oats-Hills Incorporated, Houston, Texas. (Mr. Oats and Mr. Hills have translated CODAP programs for execution on the IBM 360 and 370 series computers, under contract with the U.S. Marine Corps and the Department of Defense.)

Since 1970 all CODAP innovations and modifications have been programmed in-house by the Computational Sciences Division. During this period, many new programs were added to the system, and nearly all old programs were extensively modified. During the last two years the CODAP system has been completely rewritten for execution on the Univac 1108. This reprogramming was first undertaken by MSgt William D. Stacey, with the assistance of Sgt Johnny J. Weissmuller. After MSgt Stacey's departure from AFHRL, the translation and reprogramming were completed and checked out by Sgt Weissmuller, assisted by A1C Bruce B. Barton. It is this version of the CODAP system which is documented in this technical report.

Many individuals have contributed to the CODAP system, and it would be difficult to specifically enumerate each contribution. However, the authors would be remiss not to mention those individuals who have participated in the mainstream effort. It is possible, with the passing of time and the failing of memories, that the following history contains serious omissions and inaccuracies, but it is correct in terms of currently available information.

The hierarchical clustering programs, OVRLAP, GROUP and KPATH, were initially designed by Joe H. Ward, Jr. and R. E. Christal, with original programming by Daniel Rigney and Wayne E. Fisher. The remainder of the first CODAP package was designed by R. E. Christal, with numerous suggestions from I. R. Oats, J. R. Hills and others. This pioneering effort included JOBGRP, JOBSPC, ASFACT, PRDICT, GRMBRS, PRTVAR, VARSUM, GRPSUM, GRPDIF, MXTPRT, TSETUP, INPSTD, VARGEN, EXTRCT, and JOBINV. The programming for these was accomplished by I. R. Oats, J. R. Hills and D. W. Hartmann.

The table below lists the programs developed after the initial package:

Program	Initial Design	Initial Programming
SETCHK	W. L. Wells	Edward L. Adams
PROGEN	Edward L. Adams	Edward L. Adams
TSKNDX	R. E. Christal	William D. Stacey
AVALUE	R. E. Christal	William D. Stacey
DIST2X	C. R. Rogers	Johnny J. Weissmuller
CORREG	Joe H. Ward Jr. Robert A. Bottenberg	Janice Buchhorn Kathleen Hall William D. Stacey
AUTOJT	William Phalen	Paul Aron
PRIJOB	William Phalen R. E. Christal	Computer Sciences Corporation
DUVARS	William Phalen	William D. Stacey
REXALL	William Phalen Johnny J. Weissmuller R. E. Christal	Johnny J. Weissmuller
AVGPCT	William Phalen	Johnny J. Weissmuller
JDINDX	Johnny J. Weissmuller	Johnny J. Weissmuller
JOBIND	R. E. Christal	Edward L. Adams
TSKGRP	Harry Lawrence Johnny J. Weissmuller	Johnny J. Weissmuller
DIAGRM	William Phalen R. E. Christal Philip Aitken-Cade	Computer Sciences Corporation

Among other individuals who have played a meaningful part in the development, improvement and maintenance of the CODAP system, are the following:

Dr. Robert A. Bottenberg	Monitoring and Direction of Programming
Mr. Jim Souter	
Mr. C. R. Rogers	
Mr. J. Myer	Systems and Programming
Joe Morsh	Numerous Suggestions for Program Changes
Wayne Archer	
Harry Kudrick	Program Execution and Suggestions for Improvements
W. L. Wells	
Robert L. Vance	
Bill Feltner, Jr.	
Wesley C. Hill	
Steve D. Poole	
Terry D. Lewis	

INTRODUCTION

This document contains programmer notes on the 100 library subroutines used in the UNIVAC 1108 version of CODAP. The philosophy behind this subroutine package was to isolate both machine dependence and recurrent CODAP processing techniques. This should allow for a functional replacement of those operations on the user system. The information in this report will aid programmers in deciphering the logic and control of CODAP coding.

After the table of entry points, the writeups appear in alphabetical order by subroutine name. Each writeup includes a summary of subroutine functions and a list of all entry points. For each entry point, a calling sequence is provided with descriptions of the input and output parameters. Comments related to conversion, special applications, or restrictions may be found at the end of each writeup under SPECIAL NOTES.

The CODAP subroutine library is divided into two logical groups, those subroutines which are invoked by referencing the library, and those which must be explicitly included. The latter are referred to as "subprograms". Each is a specialized subroutine for only one main program. In the case of frequent change due to experimentation, some subroutines like RFILE (which should have been a "subprogram") were entered in the library. This allowed easy access to a subprogram which was used in a frequently changed main program (OVRLAP in the case of RFILE). In general, however, the convention holds. The 81 writeups which follow describe only those subroutines which may be used by referencing the library. The remaining 19 subprograms do not have writeups, as in analyzing the main programs the purpose and calling sequence of each should become clear. The following is an alphabetical list of the subprograms not mentioned elsewhere in this document:

SUBPROGRAM	MAIN PROGRAM
ALPHA	CORREG
APLWTS	VARGEN
CORRLB	CORREG
CRTOTP	CORREG
DICTRW	VARGEN
DISKLD	MTXPRT
EDTPRT	EXTRCT
I2FA	DIAGRM
OVLAP	MTXPRT
PLEVEL	CORREG
POSTAP	CORREG
PRIMSC	CORREG
PRINTR	EXTRCT
PRINT1	CORREG
REGREF	CORREG
START (STRT)	DIAGRM
TAPGEN	CORREG
TPTOCR	CORREG
ZEROST	CORREG

Any other entry points which are not found in the following table of entry points are either standard FORTRAN subroutines (like CLOSE or EXIT) or they are inline subroutines which may be found at the end of the main program.

ENTRY POINT TABLE

This is a list of the entry points into the CODAP subroutine library. The left-hand column is an alphabetized list of the entry points, and in the right-hand column is the name of the element in which the entry point may be found.

ASGA	ASGA
ASGAM	ASGAM
ASGC	ASGC
ASGCM	ASGCM
ASGCMR	ASGCM
AUTORV	AUTORV
BCDBIN	DATACV
BIN	SUBRTN
BINBCD	DATACV
BINFD	SUBRTN
BINOCT	SUBRTN
BLANK	BLANK
CARD	SUBRTN
CHKMSK	MSKOP1
CLOSEF	CFHIO
CLOSER	RFILE
COMPA	COMPA
COMPC	COMPC
COMPN	COMPN
COMPRC	COMPRC
CPI	CPI
CYCLES	CYCLES
DATE	SUBRTN
DATETIME	SUBRTN
DATETM	DATETM
DEFINE	RFILE
DELAY	DELAY
DISK	DISK
DISTX1	DISTX1
DKSTAT	DKSTAT
DREAD	DREAD
DSKRED	DISK
DSKSTO	DISK
ERII	ERII
ERIIX	ERII
ERRMSG	ROSTER
ERSMSK	MSKOP1
ERTRAN	ERTRAN
FACREJ	FACREJ
FDBIN	SUBRTN
FIELD	SUBRTN
FLD	FLD
FMTGEN	FMTGEN
FORMAT	FORMAT

ENTRY POINT TABLE (Con't)

FORM2X	FORM2X
FREDEV	FREDEV
FREE	FREE
FREES	FREES
FSORT	FSORT
FSORT2	FSORT
GETC	SUBRTN
GETFIL	GETFIL
GETMAS	GETMAS
GETMSK	MSKOP2
GETMSM	MSKOP2
GETPCD	GETPCD
GETPCT	GETPCT
GETRAN	IRANF
H2ACND	H2ACND
H2SORT	H2SORT
INFO	SUBRTN
INSERT	INSERT
IRANF	IRANF
INTIAL	INTIAL
IRAND	RANDOM
ISCAN	SUBRTN
JDCOPY	JDCOPY
JDEOF	JDEOF
LASCMP	LASCMP
LISORT	LISORT
LSORT	LSORT
MARGIN	NOHEAD
MARGNC	NOHEAD
MOVBIN	SUBRTN
MOVFD	SUBRTN
MOVFD1	SUBRTN
NEXREL	NEXREL
NEXREM	NEXREL
NOHEAD	NOHEAD
NOPAGE	NOHEAD
NRAND	NRAND
OCTBIN	SUBRTN
OMSG	OMSG
OMSGW	OMSG
OMSGWØ	OMSGØ
OMSGØ	OMSGØ
OPENF	CFHIO
OVRFLO	OVRFLO
PARTBL	SISO
PRINTC	ROSTER
PRTDIC	PRTDIC
PUTC	SUBRTN
RANF	RANF
RANSEQ	RANSEQ
READF	CFHIO
READR	RFILE

ENTRY POINT TABLE (Con't)

REPEND	REPEND
REPEN1	REPEN1
RESDRV	RESDRV
RESET	ROSTER
REWIND	TREAD
REWINM	DREAD
RPCOPY	RPCOPY
RPEOF	RPEOF
RPINDX	RPINDX
RUNID	RUNID
SAMPLE	SAMSEL
SBUILD	LINK
SCAN	SUBRTN
SELECT	SAMSEL
SETMSK	MSKOP1
SETSCN	SUBRTN
SETUP	SETUP
SETUP5	SETUP5
SETUP9	SETUP9
SHIFT	SHIFT
SICLS	SISO
SIGET	SISO
SIOPN	SISO
SOCLS	SISO
SOOPN	SISO
SOPEN3	TABL3
SOPUT	SISO
SPOST	SISO
SPREP	SISO
SRAND	RANDOM
SRREL	LINK
SRRET	LINK
SSORT	LINK
STATS	STATS
STATSS	STATS
STATUS	STATUS
TIME	SUBRTN
TMTCSC	TMTCSC
TREAD	TREAD
USEMSK	MSKOP2
WRITEF	CFHIO
WRITER	RFILE
ZERBLK	ZERBLK
ZERMSK	MSKOP1
ZERO	ZERO
ZERO1	ZERO1
ZERØ	SUBRTN

SUBROUTINE IDENTIFICATION

Name	ASGA (Assign an Input File)
Language	FORTRAN V
Date	Nov 1973
Programmer	Weissmuller

FUNCTION

ASGA will test to see if the requested file is already assigned to this run. If it is, ASGA establishes the FORTRAN UNIT number and returns. If it is not already assigned, ASGA issues an assignment request. If the request is accepted and the file is assigned, the FORTRAN UNIT number is established via the "@USE" card and control returns to the caller. If the request is rejected, however, a message is printed and the run is aborted.

ENTRY POINTS

1. ASGA

CALLING SEQUENCE

1. CALL ASGA (NAME, IUSE)

   a. Inputs

      (1) NAME is a three word array where words 1 & 2 are the filename in an A6,A4 format and the 3rd word is the negative file cycle in A1 format. Word 3 = blank will assign cycle Ø (the most recent cycle)

      (2) IUSE is a binary integer representing the FORTRAN UNIT to be associated with the requested file.

   b. Outputs

      (1) The file will be assigned and associated with the correct FORTRAN UNIT, <u>OR</u>

(2) The run will abort after a diagnostic.

SPECIAL NOTES

This subroutine is very machine dependent. Its sole function is to specify an input file and establish its association with a designated FORTRAN UNIT. Hence this subroutine ought to be replaced by a routine more suited to the user's installation.

SUBROUTINE IDENTIFICATION

<u>Name</u>	ASGAM (Assign a Report or JD File)
<u>Language</u>	FORTRAN V
<u>Date</u>	Nov 73
<u>Programmer</u>	Weissmuller

FUNCTION

ASGAM will test to see if the requested file is already assigned to this run. If it is, ASGAM establishes the FORTRAN UNIT for this copy of the Report or Job Description file and returns. If not already assigned ASGAM will issue an assignment request. If the request is accepted and the file is assigned, the FORTRAN UNIT is established and control returns to the caller. If the request is rejected, a message is printed and the run is aborted.

ENTRY POINTS

1. ASGAM

CALLING SEQUENCE

1. CALL ASGAM (NAME,IUSE)

   a. Inputs

      (1) NAME is a three word array where words 1 & 2 are the filename in an A6,A4 format and word 3 is blank or a negative file cycle in A1 format.

      (2) IUSE is a binary integer representing the FORTRAN UNIT to be associated with the requested file.

   b. Outputs

      (1) The file will be assigned and associated with the correct FORTRAN UNIT, <u>OR</u>

(2) The run will abort after a diagnostic.

SPECIAL NOTES

1. This subroutine is very machine dependent. Its sole function is to specify an input file and establish its association with a designated FORTRAN UNIT. Hence this subroutine ought to be replaced by a routine more suited to the user's installation.

2. The subroutine is designed to handle a 3 cycle Report or Job Description file. These files are identified externally by their "M" suffix appended to the standard CODAP filename of exactly 10 characters. The most current copy of cycle is (-Ø) and the oldest copy is cycle (-2). The existence of a (-3) cycle implies an error on a previous run. See subroutine CYCLES.

SUBROUTINE IDENTIFICATION

Name	ASGC (Assign an Output File)
Language	FORTRAN V
Date	Nov 73
Programmer	Weissmuller

FUNCTION

ASGC checks the MASTER FILE DIRECTORY to see if a file with this name already exists. If none exist, a request for a blank output tape is issued. When the request is accepted the FORTRAN UNIT is established for the new file and control returns to the caller. If, however, another file with this name already exists, the last four digits of the filename are converted to binary integer, one is added and the binary integer is reconverted to an A4 format. If this is not the fifth such attempt, control transfers to the first test above. If the last 4 digits are not numeric or five changes have been made, the subroutine prints a diagnostic and the run aborts.

ENTRY POINTS

1. ASGC

CALLING SEQUENCE

1. CALL ASGC(NAME,IUSE)

   a. Inputs

      (1) NAME is a two word array containing the output filename in A6,A4 format. A (+1) file cycle is always assigned, indicating this is the most recent copy.

      (2) IUSE is a binary integer representing the FORTRAN UNIT to be associated with the requested file.

b. Outputs

(1) The file will be assigned and associated with the correct FORTRAN UNIT, OR

(2) The run will abort after a diagnostic.

SPECIAL NOTES

This subroutine is very machine dependent. Its sole function is to specify an output file and establish its association with a designated FORTRAN UNIT. Hence this subroutine ought to be replaced by a routine more suited to the user's installation.

SUBROUTINE IDENTIFICATION

Name	ASGCM (Create 3 Reel Report or JD File)
Language	FORTRAN V
Date	Nov 73
Programmer	Weissmuller

FUNCTION

ASGCM is the same as ASGC with the following exceptions:

a. ASGCM creates 3 new tapes with an "M" suffix and leaves the third reel mounted.

b. ASGCM requests the operator to reserve a tape drive and indicate the unit number to the program.

c. ASGCM writes a binary 1 to FORTRAN UNIT 2 which is a running record of the number of "BEGIN REPORT" sentenils on the Report file.

ENTRY POINTS

1. ASGCM
2. ASGCMR

CALLING SEQUENCES

1. CALL ASGCM (NAME,IUSE)

   a. Inputs

      (1) NAME is a two word array containing the new Report or Job Description filename in an A6,A4 format. For external identification purposes, an "M" is appended to the 10 character filename signaling the operators to leave the WRITE rings in place.

      (2) IUSE is a binary integer representing the FORTRAN UNIT to be associated with the requested file.

b. Outputs

(1) The file will be assigned and associated with the correct FORTRAN UNIT, OR

(2) The run will abort after a diagnostic.

2. CALL ASGCMR

a. Inputs - NONE

b. Outputs

(1) A message to the operator to release the requested tape drive from its reserved state.

SPECIAL NOTES

1. This subroutine is very machine dependent. Its sole function is to create either a Report or Job Description file with two backup copies, and establish a FORTRAN UNIT for the most recent copy. Hence this subroutine ought to be replaced by a routine more suited to the user's installation.

2. The subroutine is designed to handle a 3 cycle Report or Job description file. These files retain their WRITE ENABLE ring as they are used for output again at a later time. (See ASGAM and CYCLES). In addition, their external filename contains an "M" suffix to alert the operators that the WRITE ENABLE rings are NOT to be removed.

SUBROUTINE IDENTIFICATION

Name	AUTORV (Automatically Revert to Backup Copy)
Language	FORTRAN V
Date	Jul 74
Programmer	Weissmuller

FUNCTION

AUTORV deletes the "current copy" entry in the MASTER FILE DIRECTORY for the Report or Job Description file. This has the effect of making the first backup copy become the most current copy. AUTORV is called only AFTER it has been decided that the reversion is necessary. This is determined by checking for a third backup copy (-3 cycle). If a third backup copy is found in the MASTER FILE DIRECTORY, a reversion is necessary since a normal termination of a CODAP run will delete the references to the third backup.

ENTRY POINTS

1. AUTORV

CALLING SEQUENCE

1. CALL AUTORV (NAME,IUSE)

   a. Inputs

      (1) NAME is a two word array containing the filename of the Report or Job Description file in an A6,A4 format. The "M" suffix will be appended automatically.

      (2) IUSE is a binary integer representing the FORTRAN UNIT number that the subroutine is allowed to use for internal processing associated with NAME.

b. Outputs

(1) The first backup copy (-1 cycle) will become the most current copy (Ø cycle) upon return from the subroutine.

SPECIAL NOTES

This subroutine is very machine dependent. Its sole function is to insure that the proper reel of a 3 reel file is used for input. If backup copies of Report and Job Description files are not maintained this subroutine is not even needed. If the using installation does not have the equivalent of a MASTER FILE DIRECTORY (an online index of tapes and mass storage files), then this subroutine cannot be programmed and the responsibility for specifying the proper reels is left to the user of the program. (See CYCLES).

SUBROUTINE IDENTIFICATION

Name	BLANK (Fill an Array With Blanks)
Language	FORTRAN V
Date	Nov 73
Programmer	Weissmuller

FUNCTION

Blank is a FORTRAN DO LOOP which fills a specified portion of an array with blanks.

ENTRY POINTS

1. BLANK

CALLING SEQUENCE

1. CALL BLANK (NWORDS,IARRAY)

   a. Inputs

      (1) NWORDS is a binary integer specifying the number of words to be blanked out.

      (2) IARRAY is an array name into which the blanks are to be inserted.

   b. Outputs

      (1) IARRAY will have NWORDS of blanks inserted.

SPECIAL NOTES

1. This subroutine uses a literal ' ' to fill the array. A data word may be established if direct use of literals is not implemented in FORTRAN at the user's installation.

2. A call of " CALL BLANK(1Ø,LIST(5))" may be used to blank out only words 5 through 14 of array LIST, leaving words 1-4 and 15-end unaltered.

SUBROUTINE IDENTIFICATION

<u>Name</u>	CFHIO (COBOL File Handler I/O Interface)
<u>Language</u>	1100 Assembler
<u>Date</u>	5 Sep 74
<u>Programmer</u>	Rogers

FUNCTION

This subroutine is an interface for the COBOL File Handler. It allows for opening, reading, writing and closing of COBOL files. If a file is not catalogued or assigned when the OPEN routine is called, parameters in the calling sequence set the type of equipment and the size of the file and the appropriate request is issued.

ENTRY POINTS

1. OPENF

2. READF

3. WRITEF

4. CLOSEF

CALLING SEQUENCE

1. CALL OPENF (FIT)

   a. Inputs

      (1) FIT is the File Information Table. It is an array of parameters, file control table, and buffer area. It is used by all calling sequences. The information in the FIT is as follows:

FIT(1),FIT(2)	12-character INTERNAL FILE NAME, left justified with trailing spaces.
FIT(3)-FIT(7)	30-character EXTERNAL FILE NAME, left justified with trailing spaces. If EXTERNAL

FILE NAME is spaces, CFHIO will use the INTERNAL FILE NAME to assign the file and no @USE will be generated.

FIT(8) BLOCK SIZE of the file.

FIT(9) LRL of the record. It is always specified in words. When the file is closed via CLOSEF, the LRL will be set to zero.

FIT(10) OPTIONS are specified as fieldata characters, six characters per word, left to right.

S1 Input/Output flag

I = Input

O = Output

S2 Buffering flag

D = Double buffering

S = Single buffering

S3 Labeling flag

O = Omitted labels

S = Standard labels

U = User labels

S4 Open options

N = No rewind

R = Rewind

S5 Close options

L = Lock (Unload)

N = No rewind

R = Rewind

S6 Dynamic assign flag

N = No print

P = Print @ASG and @USE statements

FIT(11) Assignment specifications in fieldata:

S1 Assignment flag

C = Catalogue the file

T = Temporary file

U = Unconditionally catalogue the file

S2 Space requirement

F = FASTRAND (EXEC 8 default amount)

P = Position granularity

T = Track granularity

S3-S6 Number of granules in fieldata, left justified with trailing spaces.

If S1-S6 = spaces, @ASG,A is made.

FIT(12)-FIT(94) File Control Table area. This area will be used by CFHIO and should be zero originally.

FIT(95)-FIT(N+94) Buffer area, where:

N = (BLOCKSIZE + 2) * 2 for double buffering

N = (BLOCKSIZE + 2) for single buffering

b. Outputs

(1) The file will be assigned if necessary and opened <u>OR</u>

(2) The facilities rejected messages will be printed, in either case the program continues.

2. CALL READF (FIT,RECORD,$EOF)

   a. Inputs

      (1) FIT as described above.

      (2) EOF is an alternate exit to be returned to when End-of-File is encountered. If EOF is zero, the normal return is taken. End-of-File is also indicated by LRL = Ø.

      (3) RECORD is for output onlv.

   b. Outputs

      (1) RECORD is the arrav containing the next logical record on the file. The LRL is returned in the FIT.

      (2) EOF is an alternate exit to be returned to when End-of-File is encountered. If EOF is zero, the normal return is taken. End-of-File is also indicated by LRL = Ø.

3. CALL WRITEF (FIT,RECORD)

   a. Inputs

      (1) FIT as described above with the LRL set for the output record length.

      (2) RECORD is the array containing the output image.

   b. Outputs

      (1) The RECORD arrav is written to the COBOL file.

4. CALL CLOSEF (FIT)

   a. Inputs

      (1) FIT as described above.

   b. Outputs

      (1) The COBOL file is properly closed and the LRL is set to zero.

SPECIAL NOTES

1. This routine is highly specialized for the UNIVAC COBOL FILE HANDLER. Some type of replacement will be required if FORTRAN and COBOL files are not directly compatible.

2. All CODAP programs which use COBOL files (SETCHK,INPSTD,TSKNDX,TSKGRP, REXALL, and ASFACT) require a card-image file with a blocking factor of 50. On the UNIVAC 1108 this means LRL = 14, blocksize = 751 = (14x50 + 51 control words).

SUBROUTINE IDENTIFICATION

Name	COMPA (Compare Alphabetic Values)
Language	FORTRAN V
Date	Dec 73
Programmer	Rogers/Barton

FUNCTION

COMPA compares two FIELDATA words, returning a -1,Ø, or 1 depending upon their relative values. Use is particularly suited to the FORTRAN Arithmetic IF Statement.

ENTRY POINTS

1. COMPA (Function: returns integer -1,Ø, or 1)

CALLING SEQUENCE

1. COMPA (LAV1,LAV2)

   a. Inputs

      (1) LAV1 and LAV2 contain FIELDATA words which are to be compared.

   b. Outputs

      (1) The Integer Function COMPA will take on the value Ø if LAV1 and LAV2 are identical. If LAV1 is lower in the FIELDATA collating sequence than LAV2 (e.g. LAV1 = 'EARLY1' and LAV2 = 'LATER2'), the value taken is 1. Otherwise the value is -1.

SPECIAL NOTES

1. This version of COMPA is a FORTRAN V rewrite of an IBM 7040 Assembly Language subroutine. Results depend upon the collating sequence of characters on the machine used. For example, in FIELDATA, the characters of the alphabet (A-Z) precede the alphanumeric characters 1-9, while the reverse was true of IBM.

2. COMPA is used as an argument of a subroutine by the program DIST2X.

3. The function COMPA utilizes the FORTRAN FLD Function.

4. The name COMPA is declared INTEGER and EXTERNAL.

SUBROUTINE IDENTIFICATION

Name COMPC (Compare Alphabetic or Numeric Values)

Language FORTRAN V

Date Dec 73

Programmer Rogers/Barton

FUNCTION

For a pair of words, COMPC returns the value from either COMPA or COMPN, depending upon the value of a flag word in COMMON. Use is particularly suited to the FORTRAN Arithmetic IF Statement.

ENTRY POINTS

1. COMPC (Function: returns integer -1,Ø, or 1)

CALLING SEQUENCE

1. COMPC (LAVC,LAVD)

   a. Inputs

      (1) LAVC and LAVD contain the values which are to be compared. The two words must be either both alphabetic (e.g. FIELDATA) or both numeric, and

      (2) The flag word in COMMON must be set to 1 or Ø, respectivelv.

   b. Outputs

      (1) If the value of the flag word is Ø, COMPC takes on the same value as would COMPN.

      (2) If the value of the flag word is 1, COMPC takes on the same value as would COMPA. In general, COMPC returns a Ø if the values compared are "equal", a -1 if the first value (LAVC) is "larger", and a 1 if the second value (LAVD) is "larger". The specific meaning of "larger" depends upon which function, COMPA or COMPN, is selected.

SPECIAL NOTES

1. This version of COMPC is a FORTRAN V rewrite of an IBM 7040 Assembly Language subroutine. COMPA and COMPN are used directly by this function.

2. The name COMPC is declared INTEGER in DIST2X.

SUBROUTINE IDENTIFICATION

Name	COMPN (Compare Numeric Values)
Language	FORTRAN V
Date	Dec 73
Programmer	Rogers/Barton

FUNCTION

COMPN compares two binary integer words, returning a -1,Ø, or 1 depending upon their relative values. Use is particularly suited to the FORTRAN Arithmetic IF Statement.

ENTRY POINTS

1. COMPN (Function: returns integer -1,Ø, or 1)

CALLING SEQUENCE

1. COMPN (LAV1,LAV2)

   a. Inputs

      (1) LAV1 and LAV2 contain binary integers which are to be compared.

   b. Outputs

      (1) The Integer Function COMPN will take on the value Ø if LAV1 and LAV2 are identical. If LAV1 is less than LAV2, the value taken is 1. Otherwise, the value is -1.

SPECIAL NOTES

1. This version of COMPN is a very simple FORTRAN V rewrite of an IBM 7040 Assembly Language subroutine.

2. COMPN is used as an argument of a subroutine by the program DIST2X. The function name is declared INTEGER and EXTERNAL.

SUBROUTINE IDENTIFICATION

Name COMPRC (Compare Two-Word Alphabetic or Numeric Values)

Language FORTRAN V

Date Dec 73

Programmer Rogers/Barton

FUNCTION

For two ordered pairs of words, COMPRC returns the value from COMPA or COMPN or both, depending upon the values of two flag words in COMMON. Use is particularly suited to the FORTRAN Arithmetic IF Statement.

ENTRY POINTS

1. COMPRC (Function: returns integer -1, Ø, or 1)

CALLING SEQUENCE

1. COMPRC (LAVA,LAVB)

   a. Inputs

      (1) LAVA and LAVB are each a two-word subscripted arrav containing values which are to be compared. Corresponding words in the two arrays must be either both alphabetic (e.g. FIELDATA) or both numeric, and

      (2) The corresponding flag word (of a two-word COMMON array) must be set to 1 or Ø, respectivelv.

   b. Outputs

      (1) If the value of a flag word is Ø, COMPRC takes on the same value as would COMPN for the corresponding pair of input words.

      (2) If the value of a flag word is 1, COMPRC takes on the same value as would COMPA for the corresponding pair of input words. In general, COMPRC returns a Ø if both pairs of corresponding values are "equal", a -1 if LAVA(1) or LAVA(2) is "larger" than,

LAVB(1) or LAVB(2), respectively, and 1 in the reverse instance. The specific meaning of "larger" depends upon which function, COMPA or COMPN, is selected. If the pair or words subscripted (1) are not "equal", the second pair will not be examined.

SPECIAL NOTES

1. This version of COMPRC is a FORTRAN V rewrite of an IBM 7040 Assemblv Language subroutine. COMPA and COMPN are used directlv bv this function.
2. The name COMPRC is declared INTEGER in DIST2X.

SUBROUTINE IDENTIFICATION

Name	CPI (Pack Characters into a Word)
Language	FORTRAN V
Date	April 1974
Programmer	Stacey/Barton

FUNCTION

CPI generates characters from three words of information and packs them left justified into a single word.

ENTRY POINTS

1. CPI

CALLING SEQUENCE

1. CALL CPI (NR,IAXP,LAP,LFW,NFC,KTFW,KTFC,ICMA,LOPT)

   a. Inputs

      (1) NR is an integer format repetition factor (e.g. the '7' in 7A2).

      (2) IAXP is the alpha format character (e.g. I,A,X,etc.), left justified.

      (3) LAP is an integer format field length (e.g. the '2' in 7A2).

      (4) ICMA is either a parenthesis or a comma, left justified alpha.

      (5) LOPT is an integer option flag. For LOPT = Ø, if IAXP is 'I' the subroutine changes it to 'A'. For LOPT = 2, IAXP is changed to 'A' for either 'I' or 'X'. For LOPT = 1, the IAXP input is left unchanged.

(6) KTFW and KTFC are simplv passed bv CPI between FMTGEN and INSERT.

(7) LFW and NFC are for output onlv.

b. Outputs

(1) LFW contains the left justified alpha characters generated from ICMA, NR, IAXP, and LAP. For example, if ICMA = '(ƀƀƀƀƀ' ,NR=7, IAXP = 'Aƀƀƀƀƀ', and LAP=2, after a call to CPI, LFW would contain '(7A2ƀƀ' , And

(2) NFC contains the integer number of characters packed into LFW. In the example above, NFC would equal 4. And

(3) ICMA is set to a left justified alpha comma, OR

(4) If NR or LAP exceeds 99, the run aborts after a diagnostic.

SPECIAL NOTES

1. This subroutine is highly specialized for use bv FMTGEN to create FORTRAN formats from CODAP Format Cards.

2. The subroutine is not machine dependent, except that a 6-character word is assumed, so little or no conversion need be required.

SUBROUTINE IDENTIFICATION

Name	CYCLES (Establish new copy of Report or JD file)
Language	FORTRAN V
Date	Nov 73
Programmer	Weissmuller

FUNCTION

CYCLES checks the MASTER FILE DIRECTORY to see if a third backup exists. If it does, AUTORV is called, otherwise CYCLES continues. Next, CYCLES checks to see if a new copy is already assigned to this run. If it is, the FORTRAN UNIT is established and control returns to the caller. If not, an assignment request is issued for the oldest backup copy, (-2 cycle). The assignment request specifies that if the run terminates normally all reference to this backup cycle be deleted from the MASTER FILE DIRECTORY. This physical reel will receive a copy of the most current version plus all information generated in this run. Its FORTRAN UNIT is set equal to IUSE. Next, the most current copy (-Ø) is assigned. A FORTRAN UNIT of 28 is established if the file is a Report file (IUSE = 26) or a FORTRAN UNIT of 27 is established if the file is a Job Description file (IUSE = 25). At this point the physical reel ID of FORTRAN UNIT "IUSE" (the oldest backup) is entered into the MASTER FILE DIRECTORY as a new copy of the file (+1 cycle).

Hence, if a Report file is designated (IUSE = 26), CYCLES assigns the most current copy to FORTRAN UNIT 28 and the new copy to be created (by writing over the oldest backup) to FORTRAN UNIT 26 (= IUSE).

If, however, a Job Description File is designated (IUSE = 25), CYCLES assigns the most current copy to FORTRAN UNIT 27 and the new copy to be created (by writing over the oldest backup) to FORTRAN UNIT 25 (= IUSE).

Control returns to the caller upon assignment of the files.

ENTRY POINTS

1. CYCLES — CYCLES itself does NOT copy from the current copy to the newest copy. See RPCOPY and JDCOPY.

CALLING SEQUENCE

1. CALL CYCLES (NAME, IUSE, IASGD)

   a. Inputs

      (1) NAME is a two word array containing the Report or Job Description filename in an A6,A4 format. An "M" suffix is appended to the standard CODAP filename for external use and to remind the operators that the WRITE ENABLE rino must be left in place.

      (2) IUSE a binary integer representing the FORTRAN UNIT to be associated with the newest copy of the file to be created. Must be either 25 or 26.

      (3) IASGD (output only)

   b. Outputs

      (1) IASGD will be set to 1 if the File was already assigned prior to calling CYCLES, or IASGD will be set to Ø if the subroutine CYCLES had to assign the file.

      (2) Either

         (a) The most current copy of the Job Description file associated with FORTRAN UNIT 27 and the new copy to be created is associated with FORTRAN UNIT 25 (IF IUSE = 25)

         (b) The most current copy of the Report file associated with FORTRAN UNIT 28 and the new copy to be created will be associated with FORTRAN UNIT 26 (If IUSE = 26) OR

(c) The run will abort ue to an invalid NAME being specified.

SPECIAL NOTES

1. This subroutine is very machine dependent. Its sole function is to cycle new and old copies of the Report or Job Description files in preparation to actually copy and append new information onto the newest cycle.

2. This subroutine requires a MASTER FILE DIRECTORY (an online index of tapes and mass storage files) or its equivalent. See AUTORV.

SUBROUTINE IDENTIFICATION

Name	DATACV (Data Conversions, FIELDATA/binary)
Language	1100 Assembler
Date	May 73
Programmer	Weissmuller

FUNCTION

DATACV has two entry points. One entry point is used to convert FIELDATA to binary and conversely, the other entry point is used to convert binary numbers to their FIELDATA representation. These subroutines are used instead of ENCODE or DECODE statements because they provide for an error return.

ENTRY POINTS

1. BCDBIN
2. BINBCD

CALLING SEQUENCE

1. CALL BCDBIN (IBCD, IBIN, NCHAR, $ERR)

   a. Inputs

      (1) IBCD is a FIELDATA string of characters, left adjusted.

      (2) IBIN is for output only.

      (3) NCHAR is the number of characters to attempt to convert to binary. If NCHAR=2 only the 2 left most characters will be used.

      (4) ERR is a FORTRAN statement number to which control will be passed if the leftmost NCHAR characters of IBCD contain a non-numeric. Note that preceding blanks are considered to be zeroes, but that embedded or trailing blanks are considered errors.

b. Outputs

(1) IBIN is the binary integer value of the leftmost NCHAR characters of IBCD, OR

(2) Control is transferred to statement ERR of the calling program.

2. CALL BINBCD (IBIN, IBCD, NCHAR, $ERR, IFLAG)

a. Inputs

(1) IBIN is a full word binary integer.

(2) IBCD is for output only.

(3) NCHAR is the number of character positions available to receive the FIELDATA representation of IBIN.

(4) ERR is a FORTRAN statement number to which control will be passed if the FIELDATA representation of IBIN exceeds NCHAR characters.

(5) IFLAG is a binary integer which is equal to either Ø or 1. If IFLAG = Ø, preceding zeroes of IBCD will be left as zeroes. If IFLAG = 1, preceding zeroes of IBCD will be converted to blanks.

b. Outputs

(1) IBCD is a FIELDATA string NCHAR characters long, left adjusted, OR

(2) Control is transferred to statement ERR of the calling program.

SPECIAL NOTES

1. Not only is this program very machine dependent because it is in assembler, but is is also dependent on the FIELDATA code structure. (See attachment.) This subroutine ought to be rewritten for the user's installation.

D.6. 80-COLUMN CARD CODE, SYMBOL, XS-3, FIELDATA, EBCDIC, BCD CONVERSION TABLE

Table D-6 cross references printer symbols with the card punch, XS-3, Fieldata, EBCDIC, and BCD codes.

80-Column Card Code	High-Speed Printer Symbol	XS-3	Fieldata	EBCDIC	BCD
		Octal	Octal	Hexidecimal	Octal
12-1	A	24	06	C1	61
12-2	B	25	07	C2	62
12-3	C	26	10	C3	63
12-4	D	27	11	C4	64
12-5	E	30	12	C5	65
12-6	F	31	13	C6	66
12-7	G	32	14	C7	67
12-8	H	33	15	C8	70
12-9	I	34	16	C9	71
11-1	J	44	17	D1	41
11-2	K	45	20	D2	42
11-3	L	46	21	D3	43
11-4	M	47	22	D4	44
11-5	N	50	23	D5	45
11-6	O	51	24	D6	46
11-7	P	52	25	D7	47
11-8	Q	53	26	D8	50
11-9	R	54	27	D9	51
0-2	S	65	30	E2	22
0-3	T	66	31	E3	23
0-4	U	67	32	E4	24
0-5	V	70	33	E5	25
0 6	W	71	34	E6	26
0-7	X	72	35	E7	27
0 8	Y	73	36	E8	30
0-9	Z	74	37	E9	31
0	0	03	60	F0	12
1	1	04	61	F1	01

Table D-6 80-Column Card Code, Symbol, XS-3, Fieldata, EBCDIC, BCD Conversion Table (Part 1 of 3)

80 Column Card Code	High Speed Printer Symbol	XS-3	Fieldata	EBCDIC	BCD
		Octal	Octal	Hexadecimal	Octal
2	2	05	62	F2	02
3	3	06	63	F3	03
4	4	07	64	F4	04
5	5	10	65	F5	05
6	6	11	66	F6	06
7	7	12	67	F7	07
8	8	13	70	F8	10
9	9	14	71	F9	11
12	+	20	42	50	60
11	−(minus)	02	41	60	40
12-0	?	23	54	6F	72
11-0	!	43	55	5A	52
0-1	/	04	74	61	21
2-8	&	63	46	7A	00 or 60*
3-8	=	35	44	7B	13
4-8	(apos)	56	72	7C	14
5-8	:	21	53	7D	15
6-8	>	76	45	7E	16
7-8	@	40	00	7F	17
12-3-8		22	75	4B	73
12-4-8	)	75	40	4C	74
12-5-8	[	17	01	4D	75
12-6-8	<	36	43	4E	76
12-7-8	#	37	03	4F	77
11-3-8	$	42	47	5B	53
11-4-8	*	41	50	5C	54
11-5-8	]	01	02	5D	55
11-6-8	;	16	73	5E	56
11-7-8	Δ	57	04	5F	57
0-2-8	≠	60	77	E0	32
0-3-8	(comma)	62	56	6B	33

Table D-6 80 Column Card Code, Symbol, XS-3, Fieldata, EBCDIC, BCD Conversion Table (Part 2 of 3)

80 Column Card Code	High Speed Printer Symbol	XS-3 Octal	Fieldata Octal	EBCDIC Hexadecimal	BCD Octal
0 4 8	(	61	51	6C	34
0 5 8	%	55	52	6D	35
0 6 8	\	15	57	6E	36
0-7 8	¤	77	76	6F	37
Blank	ƀ	00	05	40	20

NOTE

1 Symbols are for standard Univac Type 750 Series High Speed Printers

*00 is used as a stop code on tapes using even parity

Table D–6 80-Column Card Code, Symbol, XS-3, Fieldata, EBCDIC, BCD Conversion Table (Part 3 of 3)

SUBROUTINE IDENTIFICATION

Name DATETM (Get Date and Time from the System)

Language FORTRAN V

Date Jun 74

Programmer Weissmuller

FUNCTION

DATETM will query the System to retreive the current date and time, recode the month, reformat the information and return.

ENTRY POINTS

1. DATETM

CALLING SEQUENCE

1. CALL DATETM (IDATE, ITIME)

   a. Inputs

      (1) IDATE is for output only (2 word array)

      (2) ITIME is for output only (2 word array)

   b. Outputs

      (1) IDATE will contain the date, month, and year in an A6, A3 format, (e.g., '12 Jan 74')

      (2) ITIME will contain the current time in an A6, A2 format. (e.g. '15:25:Ø1')

SPECIAL NOTES

1. This subroutine is very machine dependent but should be easy to replace on any machine.

SUBROUTINE IDENTIFICATION

Name DELAY (Timing delay in seconds)

Language 1100 Assembler

Date Sep 73

Programmer Weissmuller

FUNCTION

DELAY will cause a program to pause for the specified number of seconds. (30 seconds is the maximum allowed.) Its primary usage is within the file assignment routines which will solicit for a tape drive every 30 seconds until one becomes available.

ENTRY POINTS

1. DELAY

CALLING SEQUENCE

1. CALL DELAY (NSEC)

   a. Inputs

      (1) NSEC is a binary integer indicating the number of seconds to pause. (Maximum allowed is 3∅.)

   b. Outputs

      (1) Execution of the program will be suspended for the specifiied period.

SPECIAL NOTES

1. This subroutine is very machine dependent. Moreover, if the user's installation does not allow or provide for repeated requests for tape drives, this subroutine may not even be needed.

SUBROUTINE IDENTIFICATION

Name	DISK (Random Access I/O)
Language	FORTRAN V
Date	Aug 73
Programmer	Weissmuller

FUNCTION

This subroutine's primary purpose was to serve as an interface for the conversion of the CODAP system from an IBM 7Ø4Ø to a UNIVAC 1108. The 7Ø4Ø version was in assembler, so to minimize recoding, the 1108 version was coded in FORTRAN to simulate the 7Ø4Ø version.

ENTRY POINTS

1. DISK
2. DSKSTO
3. DSKRED

CALLING SEQUENCES

1. CALL DISK (NREC, LENGTH)

   a. Inputs

      (1) NREC is a binary integer denoting the maximum number of random records to be written.

      (2) LENGTH is a binary integer indicating the maximum length (in words) of any given record.

   b. Outputs

      (1) A random access file established for FORTRAN unit 29.

2. CALL DSKSTO (NWORDS, IARRAY, IREC, ISW)

   a. Inputs

      (1) NWORDS is a binary integer representing the number of words to be written to the random access unit (29).

(2) IARRAY is the location of the first of NWORDS to be written or stored on unit 29.

(3) IREC is a binary integer specifying the index or key of the record to be written. IREC may be any number from 1 to the NREC value specified in CALL DISK (NREC, LENGTH)

(4) ISW is a binary integer equal to Ø or 1. If ISW = Ø, DSKSTO will simulate an IBM 7Ø4Ø random access routine and block all outputs by 465 words. If ISW = 1 (actually ≠ Ø), DSKSTO will store the information provided into a single record.

b. Outputs

(1) NWORDS of information beginning with core location IARRAY will be stored in record number IREC and will be blocked by 465 if ISW = Ø, and unblocked otherwise. <u>OR</u>

(2) The run will abort if an IREC greater than NREC is specified.

3. CALL DSKRED (NWORDS, IARRAY, IREC, ISW)

a. Inputs

(1) NWORDS is a binary integer indicating the number of words to be read from the random access unit (29).

(2) IARRAY is the location in core of the first available word to receive the information just read.

(3) IREC is a binary integer specifying the index or Key of the record to be read. Note: A request to read a record which has not been written causes an error termination. In DSKSTO IREC may be any number within the range 1 to NREC, while in DSKRED, IREC must not only be in that range, but also must have appeared in a call to DSKSTO.

(4) ISW is a binary integer equal to Ø or 1. If ISW = Ø DSKRED will simulate an IBM 7Ø4Ø random access routine and read information in blocks of 465 words. If ISW = 1 (≠ Ø), DSKRED will read unblocked data.

b. Outputs

(1) NWORDS of information will be read into IARRAY from record IREC of the random access unit (29). <u>OR</u>

(2) The run will abort if:

(a) an IREC greater than NREC is specified

(b) an IREC which was not written with DSKSTO is specified

(c) NWORDS is greater than the number of words actually in the specified record IREC.

SPECIAL NOTES

1. The DISK entry point must be called prior to either DSKSTO or DSKRED, and the DSKSTO call must logically precede the DSKRED call.
2. This subroutine is very machine dependent and is in fact a replacement of an IBM 7Ø4Ø machine dependent routine. The value of this routine is questionable. If the user's installation has a FORTRAN compiler which allows direct coding of random access reads and writes, this routine is unnecessary for new programs, but allows easy conversion of older programs. If, on the other hand, FORTRAN does not allow such direct references, this routine could be coded in the user's assembler language and pretty much replace the direct references as they appear in the UNIVAC 11Ø8 FORTRAN V version.
3. This routine is much less powerful than direct FORTRAN V random I/O statements. The FORTRAN V random access statements allows

the user to specify a list of variables to be written, and the list need not be of items contiguous in core. DSKSTO, however, will only access sequential core locations and hence any type of "skipping around" must be accomplished by moving all values into a single array (if not already equivalenced). Generally speaking these moves of one array into another are done via TMTCSC. Moreover, in order to use DSKRED, one must know the number of words in the Record prior to reading it. Hence it was decided that DSKSTO and DSKRED would not be used in new programs even though doing so would have greatly simplified conversions for other installations.

SUBROUTINE IDENTIFICATION

<u>Name</u> DISTX1 (Binary Search for Proper Interval of Distribution)

<u>Language</u> FORTRAN V

<u>Date</u> May 73

<u>Programmer</u> Weissmuller

FUNCTION

DISTX1 determines within which, if any, interval on a range a given value lies.

ENTRY POINTS

1. DISTX1

CALLING SEQUENCE

1. CALL DISTX1 (MAXN,IBOT,ITOP,IFIND,ICOMP,ISUB)

   a. Inputs

      (1) MAXN is the binary integer number of intervals.

      (2) IBOT is the array of inclusive lower bounds, one for each of MAXN intervals.

      (3) ITOP is the array of inclusive upper bounds, one for each of MAXN intervals. Both upper and lower bounds must be specified in ascending order and cannot overlap, but "gaps" between intervals are permitted.

      (4) IFIND is the value to be matched against the intervals.

      (5) ICOMP is the EXTERNAL-ized name of the comparison function (e.g. COMPA or COMPN) appropriate to the value being tested.

      (6) ISUB is for output only.

b. Outputs

(1) ISUB is the binary integer interval number (array subscript for IBOT and ITOP) within which the value lies. If the value is not within any interval specified, ISUB = Ø.

SPECIAL NOTES

1. The subroutine utilizes a binary search technique with a variable comparator function; hence COMPA, COMPN, and/or other user-specified functions need be supported.

SUBROUTINE IDENTIFICATION

Name	DKSTAT (Disk Status)
Language	110C Assembler
Date	Sep 73
Programmer	Weissmuller

FUNCTION

DKSTAT is the subroutine used by all the "assign file" subroutines. This is the routine which actually tests to see if a mass storage file is already assigned to the run. See STATUS for a similar function on tape files.

ENTRY POINTS

1. DKSTAT

CALLING SEQUENCE

1. CALL DKSTAT(ID,ICODE1,ICODE2,IFLAG)

   a. Inputs

      (1) ID is a two word array which contains the FORTRAN UNIT left adjusted in FIELDATA in the first word and the second word blank. The association between the FORTRAN UNIT and the external filename must be established prior to this call.

      (2) ICODE1 is output only.

      (3) ICODE2 is output only.

      (4) IFLAG is output only.

   b. Outputs

      (1) ICODE1 will contain the equipment code associated with the specified FORTRAN UNIT if IFLAG=Ø.

(2) ICODE2 will contain the FILE MODE if IFLAG=Ø.

(3) IFLAG will = Ø if the file associated with the specified FORTRAN UNIT is a mass storage file that is currently assigned. Otherwise IFLAG=1. Note: A tape file fails and sets IFLAG-1. (See STATUS)

SPECIAL NOTES

1. This subroutine is very machine dependent and should either be replaced or eliminated. This routine is used to avoid requesting a file which is already assigned.

2. ICODE1 and ICODE2 are never used, but are included to maintain a calling sequence similar to STATUS, the subroutine which serves an identical purpose for tape files.

SUBROUTINE IDENTIFICATION

Name	DREAD (Disk Read of FORTRAN Print)
Language	1100 Assembler
Date	Oct 73
Programmer	Weissmuller

FUNCTION

DREAD will read a FORTRAN written print file from mass storage. This subroutine is used primarily for reading Reports written to mass storage by the EXTRCT program. See TREAD for a similar function on tape files.

ENTRY POINTS

1. DREAD
2. REWINM

CALLING SEQUENCES

1. CALL DREAD(NWORDS,LINE,JSEQ,$NEW,$EOF)

   a. Inputs

      (1) NWORDS is for output only

      (2) LINE is a 22 word array for output only

      (3) JSEQ is for output only

      (4) NEW is the FORTRAN statement number to jump to if a new report sentinel is read

      (5) EOF is the FORTRAN statement number to jump to if an end of file is encountered in an attempt to read the next line

   b. Outputs

      (1) NWORDS is a binary integer which is the number of words in the print image array LINE

(2) LINE contains the next print line read. It has a maximum length of 22 words and may be printed thusly:

```
    WRITE(6,100) (LINE(I),I=1,NWORDS)
100 FORMAT(22A6)
```

(3) JSEQ is a b.nary integer which is set equal to the index number of the current Report being read. This value is set only when the $NEW return is used.

2. CALL REWINM

a. Inputs - None

b. Outputs

(1) The print image file read by DREAD is rewound

SPECIAL NOTES

1. This is a very machine dependent subroutine and should either be replaced or eliminated. The primary function of this subroutine is to read print images from mass storage and in particular, print images in Report File Format. [Report File Format simply means that individual reports are preceded by a sentinel of the form 'BEGIN REPORT',N where N is a binary integer written out in 1A6 format.]

2. Both DREAD and TREAD assume FORTRAN UNIT 28 is the print image file. This association must be established prior to calling either subroutine.

SUBROUTINE IDENTIFICATION

Name	ERII (Executive Request to Interactivity Interrupt)
Language	1100 Assembler
Date	May 74
Programmer	Rogers

FUNCTION

ERII provides a method by which operators may query the program as to its name and/or status. This subroutine sets up a message buffer and a separate activity which the operators may interrogate.

ENTRY POINTS

1. ERII
2. ERIIX

CALLING SEQUENCE

1. CALL ERII (ISW, NWORDS, MESAGE, IFLAG)

   a. Inputs

      (1) ISW is a FIELDATA "P" or "L" which sets a program switch to either Print all operator responses or only enter them into the Log. Basically a "P" is used for interactive runs and a "L" is used for batch runs. All standard CODAP programs are designed for batch runs.

      (2) NWORDS is a binary integer indicating the number of words to be printed from MESAGE. Note: Since this call is only done once, NWORDS should be large enough to accommodate the longest message to be written.

      (3) MESAGE is an array of at least NWORDS which contains the message to be displayed for the operators. ERII always picks up the current contents of MESAGE, so MESAGE may be

changed at any time in the main program to update the status.

(4) IFLAG is for output only.

b. Output

(1) If the operator solicits ERII, the current contents of MESAGE will be displayed on the console screen. To solicit ERII, the operator must type in:

```
II  rrrrrr X,
```

where "II" is the interactivity interrupt command for EXEC 8, rrrrrr is a 1 to 6 character run identification code and X is an operator response. X may be "P", "C" or "M". The "P" will cause MESAGE to be printed, "C" will set IFLAG to a non-zero value, and "M" will solicit and transmit a message from the operator to the executing program.

(2) For the OVRLAP and GROUP programs, IFLAG is used to force checkpoints prior to the normally scheduled time. These programs set the IFLAG variable to zero and every time IFLAG is found to be non-zero, a checkpoint is taken. IFLAG is set to a non-zero value only if the operator enters a response that begins with a "C".

2. CALL ERIIX

a. Inputs - None

b. Outputs

(1) The separate activity established by the call to ERII is halted. This must be done in order to avoid an abnormal termination of the run.

SPECIAL NOTES

1. This subroutine is very, very machine dependent and its sole function is to allow program - operator interaction. In many installations this practice is highly discouraged. ERII may be removed from the system with very little difficulty and no ill effects will arise.

SUBROUTINE IDENTIFICATION

Name ERTRAN (FORTRAN V - EXEC 8 Interface)

Language 1100 Assembler

Date -

Programmer UNIVAC

FUNCTION

ERTRAN is a FORTRAN V - EXEC 8 Interface subroutine which allows access to system information and processing capabilities. This is a UNIVAC subroutine and no symbolic coding is available. This write-up is included for reference only.

ENTRY POINTS

1. ERTRAN (Subroutine Call)
2. NERTRN (Function: returns I/O status word)

CALLING SEQUENCES FOR COMMON USAGES

1. CALL ERTRAN (2)

   a. Inputs

      (1) Binary integer number 2.

   b. Output

      (1) The run takes an error termination.

      Note: This calling sequence is used only after an unrecoverable error has been detected.

2. CALL ERTRAN (6, IMAGE)

   a. Inputs

      (1) Binary integer number 6.

      (2) An array, IMACE, which contains a valid EXEC 8 control card image which ends with the sequence "␢.␢". See attachment for valid EXEC 8 control statements from ERTRAN.

b. Output

(1) EXEC 8 performs the requested action, OR

(2) Run aborts if requested action was not possible.

3. IOERR = NERTRN (6, IMAGE)

a. Inputs

(1) Binary integer number 6.

(2) IMAGE is an array containing a valid EXEC 8 control card image which ends with "b̸.b̸". Normally in this usage, IMAGE contains a request to assign a file.

(3) IOERR is for output only.

b. Outputs

(1) Either the request action is completed and IOERR = 0, OR

(2) IOERR is set equal to the error status word or zero.

Note: This Calling sequence is used in the subroutine GETFIL.

4. CALL ERTRAN (9, IDATE, ITIME)

a. Inputs

(1) Binary integer number 9.

(2) IDATE is for output only.

(3) ITIME is for output only.

b. Outputs

(1) IDATE contains the current date in FIELDATA in for form MMDDYY.

(2) ITIME contains the current time in FIELDATA in the form HHMMSS.

Example: If called at 7:52 p.m. and 14 seconds, on
August 19, 1974, then
IDATE = "081974" and
ITIME = "195214"

This calling sequence is used in the subroutine DATETM.

SPECIAL NOTES

1. This subroutine is obviously very machine dependent. Calli g sequences 1 and 4 are rather trivial and may be easily duplicated or removed. Calling sequences 2 and 3 however, are crucial to the file manipulation subroutines (SETUP, SETUP6, and SETUP9). If the calling sequences 2 and 3 cannot be simulated, file assignments and their associations with the proper FORTRAN UNITS may be forced directly onto the user of each individual program. In that event, program documentation will have to be augmcnted.

3.4.6. ERTRAN - EXECUTIVE REQUEST ROUTINE

- Purpose

ERTRAN routine allows the FORTRAN programmer to reference some of the executive request functions, namely ABORT$ (abort run), ERR$ (error return), EXIT$ (normal exit), CSF$ (generate control statement), SETC$ (set condition word), COND$ (retrieve condition word), and DATE$ (request date and time) The executive requests are described in *UNIVAC 1100 Multi-Processor System Operating System, EXEC 8 Programmer Reference, UP-4144* (current version).

- FORTRAN V Reference

ERTRAN may be called by CALL ERTRAN (args) or referenced as a function by I = NERTRN (args). If the function reference is used, the function value consists of the error or status information or zero if no error status is provided.

The reference has three forms

ERTRAN(k) for k = 1, 2, or 3

ERTRAN(k, ARG1) for k = 6, 7, or 8

ERTRAN(k, ARG1, ARG2) for k = 9, where ARG1 and ARG2 represent (INTEGER) type variables, and k determines the executive request function.

- Assembler Language Reference

ERTRAN cannot be referenced by the assembler language programmer; however, each of the executive request functions can be referenced directly in assembler language

- Routines Referenced

NERR$ by way of entry points NERR$ and FIELD$ and executive request functions ABORT$, ERR$, EXIT$, CSF$, SETC$, COND$, DATE$, and PRINT$.

- Description

If k = 1, the executive request function ABORT$ is referenced. All current activities are terminated and the run is terminated in an abort condition.

If k = 2, the executive request function ERR$ is referenced. The Err Mode condition is set. If the programmer has established an Err Mode routine, control will be returned to it. Otherwise, standard Err Mode action will occur terminating the run.

If k = 3, the executive request function EXIT$ is referenced. The routine provides normal program termination.

k = 4 and k = 5 are illegal.

If k = 6, the executive request function CSF$ is referenced. The routine provides the user with a means of submitting an executive control statement image (ARG1) for interpretation and processing. The image submitted must be an array or a Hollerith field containing one of the control statements listed below. The control statement must not be longer than 80 characters and may be terminated by the character sequence: blank, period, blank.

@ADD	Add to run stream
@ASG	Assign a file
@BRKPT	Breakpoint symbiont output files
@CAT	Catalog a file
@CKPT	Produce check point dump of this run
@FREE	Deassign a file
@LOG	Message to log a file
@MODE	Set mode and/or noise constant for tape file
@QUAL	File qualification
@RSTRT	Restart run whose check point dump was saved by @CKPT
@START	Schedule an independent run
@SYM	Queue files
@USE	Associate internal to external file name

If k = 7, the executive request function SETC$ is referenced. The routine places (sets) the contents of the lower third (bits 11-00) of ARG1 in the corresponding third of the run condition word. The lower two thirds of the run condition word is used as a flag which can be either tested by the control statement @TEST or retrieved by the FORTRAN call CALL ERTRAN (8, ARG1) and then tested.

If k = 8, the executive request function COND$ is referenced. The routine retrieves the condition word and makes it available to the user in ARG1.

If k = 9, the executive request function DATE$ is referenced. The routine supplies the user with the current date and time in ARG1 and ARG2 respectively. The date in ARG1 is in the format MMDDYY where MM represents the month (01-12), DD the day (01-31), and YY the last two digits of the year (00-99). The time in ARG2 is in the format HHMMSS where HH represents the hours (00-24), MM the minutes (00-60), and SS the seconds (00-60).

If k is negative or greater than 9, error termination results. Error termination also results when the CALL ERTRAN(arg) is used and the status word is negative. When the I = NERTRN (arg) function is used, error termination will not take place, based on the status of the operation. ERTRAN has 97 instructions and 70 data words for a total main storage requirement of 167 words.

SUBROUTINE IDENTIFICATION

Name	FACREJ (Facility rejection messages)
Language	1100 Assembler
Date	Oct 73
Progr.. mer	Blakley

FUNCTION

FACREJ translates the EXEC 8 I/O status word into appropriate error messages. The I/O status word is a 36 bit word in which each bit denotes a particular error condition.

ENTRY POINTS

1. FACREJ

CALLING SEQUENCE

1. CALL FACREJ (IOERR)

   a. Inputs

      (1) IOERR is the EXEC 8 I/O status word returned by an I/O operation.

   b. Outputs

      (1) A printed list of error messages from the I/O operation.

SPECIAL NOTES

1. This subroutine is extremely machine dependent. If the user's installation has sufficient diagnostics prov'.ded by the system, this subroutine is completely useless. It may be dropped from the system with no ill side affects.

SUBROUTINE IDENTIFICATION

Name	FLD (Extract a Field of Bits) .
Language	1100 Assembler
Date	--
Programmer	UNIVAC

FUNCTION

FLD is a UNIVAC function in FORTRAN V which allows access to any string of bits within a single 36 bit computer word. This function is used to extract the "minutes" field out of a word containing the time in HHMMSS format.

ENTRY POINTS

1. FLD (Function: returns an integer value)

CALLING SEQUENCE

1. IVAL = FLD(IBIT,NBITS,IWORD)

   a. Inputs

      (1) IBIT is a binary integer in the range 0 to 35 which identifies the first bit of the string to be extracted. Bits are counted from left (Ø) to right (35) unless overidden by COMPILER (FLD=R) statement which reverses the sequence.

      (2) NBITS is a binary integer in the range 1 to 36 which denotes the number of bits in the string tc be extracted.

      (3) IWORD is an integer, real, logical, or typeless expression from which the string is to be extracted.

      (4) IVAL is output onlv.

b. Outputs

(1) IVAL will contain the extracted field right adjusted with leading zeroes. If "VAL" rather than "IVAL" were used, the resulting binary integer represented by the extracted string would be converted to floating point notation then stored in VAL.

SPECIAL NOTES

1. Even though this is an intrinsic FORTRAN V function it could easily be replaced by an assembler function at the user's installation. Note, however, FLD may be used on the left side of an equals sign in FORTRAN V, and this is probably not allowed in other FORTRANs. Minor reprogramming could remove such references if they occur.

2. If IBIT or NBITS above are not in the allowable range highly erratic results will occur. The statement COMPILER (FLD=ABS) is used to force these arguments to be positive values, as all negative numbers are not in the accepted range.

SUBROUTINE IDENTIFICATION

Name FMTGEN (FORTRAN Format Generator)

Language FORTRAN V

Date Apr 73

Programmer Stacey/Barton

FUNCTION

FMTGEN reads CODAP Format Cards (see attachment) and generates FORTRAN formats which are used to read raw data card images.

ENTRY POINTS

1. FMTGEN

CALLING SEQUENCE

1. CALL FMTGEN (LOPT)

   a. Inputs

      (1) LOPT is a binary integer option flag which determines the type of field specified for Task variables (T) and blanks (b̸).
      LOPT = Ø produces an A format for T's and an X format for b̸'s.
      LOPT = 1 produces an I format for T's and an X format for b̸'s.
      LOPT = 2 produces an A format for both T's and b̸'s; this enables the transfer of data from card columns corresponding to blanks on the Format Cards.

      (2) The number of CODAP Format Cards must be specified in the second-from-last word of the named COMMON area.

      (3) The proper Format Card deck must be provided (see attachment).

b. Outputs

(1) Most of the data and control information are passed in a named COMMON area between FMTGEN and the calling program. This 5504-word area contains arrays for the output of the generated FORTRAN format, tables for relative [illegible] of Task, History, and Check data, the actual Check characters, tables for pointers to input card boundaries and card [illegible] item counts, and one-word counts of Task, History, and Check items formatted, OR

(2) Format Card errors will cause the run to abort after a diagnostic.

SPECIAL NOTES

1. This subroutine is tailored expresslv for SETCHK and INPSTD, the programs through which data enter the CODAP system.

2. FMTGEN requires the subroutines ERTRAN, CPI and INSERT, but is subject to no more than the same restrictions as are these subroutines. In particular, a 6-character word is assumed.

FORMAT CARDS

These cards describe the SETCHKed raw data file. There must be a format card to describe each raw data card within a case. If a case consists of 20 raw data cards, then 20 format cards are required. There are no restrictions on the layout of raw data cards or format cards except that each raw data card must contain a control number in columns 1-4 and each format card must contain 'C...' in columns 1-4. Raw data fields are defined by the following format characters:

C Defines the beginning of the case control number. This character must appear in column 1 of each format card.

H Defines the beginning of a history variable.

T Defines the beginning of a task variable.

. Defines a continuation of any of the above. A task variable may occupy no more than 6 columns (T.....). The length of task variables (in terms of columns) must remain constant for all tasks.

b̸ A blank column defines a skip.

other Any other character is defined as a 'check' character. A check character must have a matching character in the corresponding column of the raw data. A maximum of 500 'check' characters is allowed.

SUBROUTINE IDENTIFICATION

<u>Name</u>	FORMAT (Retrieve Print Format)
<u>Language</u>	FORTRAN V
<u>Date</u>	Oct 73
<u>Programmer</u>	Weissmuller

FUNCTION

FORMAT decodes a print format from the Variable Dictionary on the front of a History or KPATH file. This subroutine also returns flags indicating the type and length of variables.

ENTRY POINTS

1. FORMAT

CALLING SEQUENCE

1. CALL FORMAT(LDICT(14,N),IMAT,NCHAR,ITYPE,IFLD1,IFLD2,$ERR)

   a. Inputs

      (1) LDICT is a two dimensional array which contains the Variable Dictionary entries. There are 15 words per entry and up to 16 entries per block. The format is contained in the last two words of the entry (words 14 and 15). N is number of the entry within the current block.

      (2) IMAT, NCHAR, ITYPE, IFLD1, and IFLD2 are for output only.

      (3) ERR is the FORTRAN statement number to jump to if the dictionary entry does not contain a valid format.

   b. Outputs

      (1) IMAT is a two word array which contains the format in FIELDATA, left adjusted with all extra blanks omitted. <u>OR</u> IMAT contains ' NO FORMAT ' or ' INVALID ' if the error is used.

(2) NCHAR is a binarv integer which is a count of the non-blank characters in IMAT.

(3) ITYPF is:

(a) -1 if the variable is alphanumeric ("A" format)

(b) Ø if the variable is binary integer ("I" format)

(c) +1 if the variable is floating point ("F" format)

(4) If ITYPE is:

(a) -1, IFLD1 is a binarv integer indicating a repetition factor for a background variable (e.g. 1 for 1A3). IFLD2 is a binary integer indicating the number of alphanumeric characters to be used (e.g. 3 for 1A3).

(b) Ø, IFLD1 is a binary integer indicating the number of decimal digits required to print the maximum value (e.g. 7 for I7).

IFLD2 is not used.

(c) +1, IFLD1 is a binary integer indicating the maximum number of spaces required to print the value of the variable. This includes "+" or "-" and a decimal point (e.g. 1Ø for F1Ø.3).

IFLD2 is a binary integer indicating the number of decimal digits to print following the decimal point (e.g. 3 for F1Ø.3).

SPECIAL NOTES

1. This subroutine uses the FORTRAN V FLD function, negative DO LOOP indexing and ENCODES and DECODES. For those reasons it is rather machine dependent. It mav be easilv replaced bv an assemblv subroutine.

2. FORMAT expects the format to depict a single word variable and hence will return an invalid response to variables like SSAN (A6,A3) or ORGANIZATION (1ØA6,A4).

SUBROUTINE IDENTIFICATION

Name	FORM2X (Formats for DIST2X)
Language	FORTRAN V
Date	Dec 73
Programmer	Barton

FUNCTION

FORM2X encodes a format for DIST2X output according to parameters set by that program.

ENTRY POINTS

1. FORM2X

CALLING SEQUENCE

1. CALL FORM2X (KX,LY,FIRST,MAT)

   a. Inputs

      (1) KX,LY and FIRST are program-generated parameters which control Computed GO TO chains within the subroutine.

      (2) MAT is for output only.

   b. Outputs

      (1) MAT is an array for the ENCODE-d FORTRAN format.

SPECIAL NOTES

1. This subroutine is an expedient resulting from the conversion of DIST2X from the IBM 7040 to the UNIVAC 1108. IBM supported the use of a FORTRAN statement number in the WRITE format parameter which led, via inline Computed GO TO chains, to the appropriate fixed FORMAT statement. Since UNIVAC does not have this capability, but allows variable formats by ENCODE-ing the specifications into arrays, FORM2X was written to use the same parameter values to control the contents of the words in ENCODE lists. Obviously, FORM2X is highly specialized, but reduces the machine dependence of DIST2X.

SUBROUTINE IDENTIFICATION

Name FREDEV (Free Device)

Language 1100 Assembler

Date Nov 73

Programmer Weissmuller

FUNCTION

FREDEV erases FORTRAN V's record of a unit assignment. For each FORTRAN unit (Ø thru 3Ø) there is an entrv in a table (NTAB$) which associates the FORTRAN unit with the external file. If the external file is de-assigned using ERTRAN during a FORTARN program, FORTRAN is not aware of it and it attempts to rewind the file at the end of the program. This causes an abnormal termination. This subroutine, then, erases the NTAB$ entrv, thereby informing FORTRAN the device is no longer assigned and should not be rewound.

ENTRY POINTS

1. FREDEV

CALLING SEQUENCE

i. CALL FREDEV (IUNIT)

   a. Inputs

      (1) IUNIT is a binarv integer representing the FORTRAN unit to be released.

   b. Outputs

      (1) FORTRAN's record of the unit is erased.

SPECIAL NOTES

1. This subroutine is extremely machine dependent but mav not be needed at all in other FORTRAN's. It's only usage is in the subroutine FREE.

SUBROUTINE IDENTIFICATION

Name	FREE (Free a File)
Language	FORTRAN V
Date	Nov 73
Programmer	Weissmuller

FUNCTION

FREE will release a file and its associated device so that they may be used by other runs. This is most important with tape files as the tape drives are at a premimum.

ENTRY POINTS

1. FREE

CALLING SEQUENCE

1. CALL FREE(IUNIT)

   a. Inputs

      (1) IUNIT is a binary integer representing the FORTRAN unit to be released.

   b. Outputs

      (1) The FORTRAN unit is freed and the device is released. (SEE FREDEV)

SPECIAL NOTES

1. This subroutine is machine dependent to the extent that it uses ERTRAN to interact with the system to release a file and its device. This ought not be difficult to program in assembler if necessary.

SUBROUTINE IDENTIFICATION

Name	FREES (Free a File, but Save the Tape Drive)
Language	FORTRAN V
Date	Nov 73
Programmer	Weissmuller

FUNCTION

FREES will release a tape file but retain its associated drive so that it may be used by a subsequent assignment. This is very important as the tape drives are at a premimum.

ENTRY POINTS

1. FREES

CALLING SEQUENCE

1. CALL FREES (IUNIT)

   a. Inputs

      (1) IUNIT is a binary integer representing the FORTRAN unit to be released.

   b. Outputs

      (1) The FORTRAN unit is freed and the drive is retained.

SPECIAL NOTES

1. This subroutine is machine dependent to the extent that it uses ERTRAN to interact with the system to release a file and not its drive.

SUBROUTINE IDENTIFICATION

Name	FSORT (FORTRAN-EXEC 8 SORT Routine)
Language	FORTRAN V
Date	Oct 73
Programmer	Weissmuller

FUNCTION

FSORT is a FORTRAN subroutine which is called to sort all records on a given FORTRAN unit. This routine calls LINK and TABL3 which are routines used to access the UNIVAC Sort Package. Records are read from the specified FORTRAN unit, passed to the sort package, then rewritten to the FORTRAN unit in the sorted sequence.

ENTRY POINTS

1. FSORT
2. FSORT2

CALLING SEQUENCE

1. CALL FSORT (IUNIT,NWORDS,ITYPE,IORDER)

   a. Inputs

      (1) IUNIT is a binary integer denoting the FORTRAN unit.

      (2) NWORDS is a binary integer indicating the maximum number of words in the largest record to be sorted.

      (3) ITYPE is a binary integer equal to either

      Ø if an alphanumeric sort is desired or a

      1 if a numeric sort is required.

(4) IORDER is a binary integer equal to either

Ø if ascending sequence is desired, or

1 if descending sequence is required.

b. Outputs

(1) All records on FORTRAN unit IUNIT will be in the requested sequence based on the first word of the record.

2. CALL FSORT2 (IUNIT,NWORDS,ITYPE,IORDER,JTYPE,JORDER)

a. Inputs

(1) Same as above, plus

(2) JTYPE is a binary integer equal to either

Ø if an alphanumeric sort is needed on the second variable, or

1 if a numeric sort is needed on the second variable.

(3) JORDER is a binary integer equal to either

Ø if ascending sequence is desired for the secondary sort, or

1 if descending sequence is desired on the secondary sort.

b. Outputs

(1) All records on FORTRAN unit IUNIT will be in the requested sequences based on the first two words of the records.

SPECIAL NOTES

1. This is a specialized subroutine which ought to be replaced by an interface to the user's installation sort package.

2. Note: The FORTRAN unit must contain unformatted records in order to work properly as full-word sort fields are assumed.

SUBROUTINE IDENTIFICATION

Name GETFIL (Get a Tape File)

Language FORTRAN V

Date Oct 73

Programmer Weissmuller

FUNCTION

GETFIL will repeatedly reissue any file assignment request passed to it. Only tape file assignments should be requested via this subroutine as it issues a message to the operators every two minutes which reads "WAITING ON TAPE DRIVE. AE" The assignment request is reissued every 15 seconds until accepted or an error status other than "WAIT" is returned.

ENTRY POINTS

1. GETFIL

CALLING SEQUENCE

1. CALL GETFIL (IMAGE)

   a. Inputs

      (1) IMAGE is an array which contains a valid file assignment control statement.

   b. Outputs

      (1) Either the tape file is assigned, <u>or</u>

      (2) The run terminates with diagnostics.

SPECIAL NOTES

1. This subroutine is rather machine dependent but is used strictly within the file assignment package. As the file assignment package is apt to be entirely rewritten at each installation this routine is not very significant.

SUBROUTINE IDENTIFICATION

Name	GETMAS (Get a Mass Storage File)
Language	FORTRAN V
Date	Oct 73
Programmer	Weissmuller

FUNCTION

GETMAS will repeatedly reissue any file assignment request passed to it. This subroutine will issue the request and if a "WAIT" status is returned, 30 seconds later, the request will be reissued. This is repeated until either the request is accepted or until an error status other than "WAIT" is returned.

ENTRY POINTS

1. GETMAS

CALLING SEQUENCE

1. CALL GETMAS (IMAGE)

   a. Inputs

      (1) IMAGE is an array which contains a valid file assignment control statement

   b. Outputs

      (1) Either the file is assigned, <u>or</u>

      (2) The run terminates with diagnostics.

SPECIAL NOTES

1. This subroutine is rather machine dependent but is used strictly within the file assignment package. As the file assignment package is apt to be entirely rewritten at each installation this routine is not very significant.

SUBROUTINE IDENTIFICATION

Name	GETPCD (Get Program Control Table Data)
Language	1100 Assembler
Date	May 74
Programmer	Weissmuller

FUNCTION

GETPCD will retrieve information from the PCT (Program Control Table). This table contains information about the run-id, the account being charged, and the running time. This routine, though elegant, is by no means essential.

ENTRY POINTS

1. GETPCD

CALLING SEQUENCE

1. CALL GETPCD(IDEC,IVAL)

   a. Inputs

      (1) IDEC is a binary integer (Ø-24Ø) indicating which word of the PCT is to be copied into IVAL.

      (2) IVAL is for output only.

   b. Output

      (1) IVAL will contain the contents of the appropriate word of the PCT.

SPECIAL NOTES

1. This is an extremely machine dependent subroutine but it may be deleted with no ill side-effects. Its primary function was to provide independent time estimates for individual programs.

SUBROUTINE IDENTIFICATION

Name	GETPCT (Get Program Control Table Data)
Language	FORTRAN V
Date	May 74
Programmer	Weissmuller

FUNCTION

GETPCT will retrieve information from the PCT (Program Control Table). This table contains information about the run-id, the account being charged, and the running time. This routine, though elegant, is by no means essential.

ENTRY POINTS

1. GETPCT

CALLING SEQUENCE

1. CALL GETPCT (IOCT,IVAL)

   a. Inputs

      (1) IOCT is a octal integer (Ø-Ø36Ø) indicating which word of the PCT is to be copied into IVAL.

      (2) IVAL is for output only.

   b. Output

      (1) IVAL will contain the contents of the appropriate word of the PCT.

SPECIAL NOTES

1. This is an extremely machine dependent subroutine, but it may be deleted with no ill side-effects. Its primary function was to provide independent time estimates for individual programs.

..S.3 PROGRAM CONTROL TABLE (PCT)

The material presented in this subsection is a supplement to the PRM, UP-4144, Revision 2.

4.8.3.1 The Element PCT (Based On Level 27.0)

*'s and +'s as well as the Reserved Word for User are explained at the end of the table.

WORD	TAGS				
0∅	AA	ORIGINAL RUN IDENTITY			
01	AB	GENERATED RUN IDENTITY			
02	AC	TOTAL ACCUMULATED RUN TIME FOR ALL COMPLETED TASKS			
03	AD	ESTIMATED RUN TIME (200 MICROSECONDS INC.)			
04	AE AY AZ AX	ACCT PRTY	ABORT$ IND	T OPTION	CORE QUEUE ADDRESS
05	AL AH	QUAL. TBL. START (REL)			ACTIVITY NAME TBL OR ∅
06	EA EB EC	CONTINGENCY MASK (PROG)		CONTIG CN	PROG CONTINGENCY ADDR OR ∅
07	AG DY AF	CHCPU COUNT		LOG COUNT	ESI ACCNT NUMBER
10	AM	* TOTAL I/O REQUEST COUNT			
11	AN	TOTAL I/O DATA TRANSFER COUNT			
12	AO	RESERVED FOR USER			
13	AP	CORE-SECOND ACCUMULATION			
14	AQ	CORE-BLOCK-SECOND			
15		WAIT$ COUNT			WAIT$ CHAIN
16	AS	RUN START TIME AND DATE (TDATE$ FORMAT)			
17	BB	XQT OPTIONS (BIT 25=A-----BIT ∅=Z)			
20	BC	CONDITION WORD			
21	BD	MOST RECENT QUALIFIER (12 CHARACTERS) ↕			
22		INITIAL VALUE WILL BE PROJECT IDENTITY			
23	BI	ACCOUNT NUMBER (12 CHARACTERS)			
24		(2ND WORD -- ACCT NO)			
25	CA CB	CORE QUANTUM USED			CORE QUANTUM
26	CC	TOTAL ACCUMULATED RUN TIME (200 MICROSECONDS INC.)			
27	CI CE	REAL TIME ACTIVITY COUNT			SWAP LOCK COUNTER
30	IL CG CH IK	TS SEG LD	PROG.TYPE	TYPE AND LEVEL	PROGRAM SIZE
31	CL	ACT REL VIA AWAIT$ (BIT 35=ACTID35-BIT 1=ACTID1-BIT ∅ ALWAYS ∅			
32	CM	MASK CONTROL WORD OF EXISTING ACTIVITIES			
33	CN	ACTIVITY MASK OF ACTIVITY 1			
34	CO	ACTIVITY MASK OF ACTIVITY 2			
35	CP	ACTIVITY MASK OF ACTIVITY 3			
		:			
72	DS	ACTIVITY MASK OF ACTIVITY 32			
73	DT	ACTIVITY MASK OF ACTIVITY 33			
74	DU	ACTIVITY MASK OF ACTIVITY 34			
75	DV	ACTIVITY MASK OF ACTIVITY 35			
76	DW DX	RELATIVE ADDRESS OF LAST ASA			CURRENT ACTIVITY COUNT
77	IE IP CP IF	T/S CH	COND OF RUN	MAX RTL	LINKAGE TO CPOOL$ REF

WORD	TAGS	RUN CONTROL SECTION			
TSASA	JP EN	ASA T/S	CORE CLEAR-ING FLAG	GR PGE FLAG	PCT ABORT MASK
101	IA++	CHAIN FOR EXEC WORKERS (ASA) ATTACHED TO USER'S PCT			
102	IM IN IC ID	CHECKPOINT COUNT	DISP MSG HANDLER BUSY DO NOT TERM	ERR FLAG	DEACT COUNT
103	ED EP	LAST PCT ABSOLUTE ADDRESS			NEXT PCT ABSOLUTE ADDRESS
104	EQ	IBANK CORE DESCRIPTOR			
105	ES	DBANK CORE DESCRIPTOR			
106	ER EY	RELATIVE IBANK VALUE			IB/DB SEPARATOR VALUE
107	ET	CORE QUANTUM TIME			
110	EU	FILENAME OF PROGRAMS PF USED BY			
111	EV	LOAD$ AND PMD			
112	EW BA EX	TS EXIT	NOT USED	FILE HEADER TABLE REC. ADDRESS	
113	IJ IQ CD	TS PCTABT	QNM FLAG	NOT USED	NUMBER OF SEGMENTS
114	JI JJ JK JL	TS ESI CN	ESI CON PN	ESI CON CNT	ESI CONT ROUTINE ADDRESS
115	WG MW WE JA	TS TIMING	MCORE-LCORE FLAG	INS QUANT	COUNT FOR OUTSTANDING I/O REQUESTS
116	JD JB JC	PCT ITEM CHAIN START			PCT ITEM CHAIN END
117	JG JE	TS PCT TM	DO NOT USE	PCT TM FLAG	ER DACT CHAIN
120	JM JZ JX JN	ESI AC TS	* *	* * *	ES ACTIVITY COUNT
121	JO JR NB JY	RLIST TS	DEACT ERRF	FORCED PAC	RLIST BUFFER LINK
122	JS JQ	SUSPEND FLAG			BUFFER ADDR. FOR PMD
123	JT JU JW JV	TS PCTLNK	SIZE IND	PMD FLAG	NOT USED
124	KA	PROGRAM START (TIME AND DATE)			
125	KB	PROGRAM NAME (1ST 6 CHARACTERS)			
126	KC	NAME (LAST 6 CHARACTERS)			
127	KD	PROGRAM VERSION NAME (1ST 6 CHARACTERS)			
130	KE	VERSION NAME (LAST 6 CHARACTERS)			
131	KF KG	POSITIONS ASSIGNED TYPE 030			TRACKS ASSIGNED TYPE 030
132	KH KJ	POSITIONS ASSIGNED TYPE 031			TRACKS ASSIGNED TYPE 031
133	KL KM	POSITIONS ASSIGNED TYPE 032			TRACKS ASSIGNED TYPE 032
134	KN KO	POSITIONS ASSIGNED TYPE 033			TRACKS ASSIGNED TYPE 033
135	KP KQ	POSITIONS ASSIGNED TYPE 034			TRACKS ASSIGNED TYPE 034
136	KR KS	POSITIONS ASSIGNED TYPE 035			TRACKS ASSIGNED TYPE 035
137	KU KV	POSITIONS ASSIGNED TYPE 036			TRACKS ASSIGNED TYPE 036
140	KW KX	POSITIONS ASSIGNED TYPE 037			TRACKS ASSIGNED TYPE 037
141	KY	TIME OF LAST TRACK SECOND CALCULATION			
142	KZ	(SCRATCH) TRACK SECONDS TYPE 030			
143	LB	TRACK SECONDS TYPE 031			
144	LC	TRACK SECONDS TYPE 032			
145	LI	TRACK SECONDS TYPE 033			
146	LJ	TRACK SECONDS TYPE 034			

SUBROUTINE IDENTIFICATION

Name H2ACND (Ascending Sort on Right Half Word)

Language FORTRAN V

Date Mar 74

Programmer Barton

FUNCTION

H2ACND sorts an array into ascending sequence on the right half (H2) of each word.

ENTRY POINTS

1. H2ACND

CALLING SEQUENCE

1. CALL H2ACND (NNDUP,LPACK)

   a. Inputs

      (1) NNDUP is the binary integer number of words in the array to be sorted.

      (2) LPACK is the array (name).

   b. Outputs

      (1) The words are reordered such that the values of the right halves of the words in subscript order grow larger.

SPECIAL NOTES

1. This subroutine performs a straightforward bubble sort, using the FORTRAN FLD Function for halfword access.

2. H2ACND is called by OPTRAN.

SUBROUTINE IDENTIFICATION

<u>Name</u> H2SORT (Descending Sort on Right Half Word)

<u>Language</u> FORTRAN V

<u>Date</u> Dec 73

<u>Programmer</u> Barton

FUNCTION

H2SORT sorts an array into descending sequence on the right half (H2) of each word.

ENTRY POINTS

1. H2SORT

CALLING SEQUENCE

1. CALL H2SORT (NRESP,LPACK)

   a. Inputs

      (1) NRESP is the binary integer number of words in the array to be sorted.

      (2) LPACK is the array (name).

   b. Outputs

      (1) The words are reordered such that the values of the right halves of the words in subscript order grow smaller.

SPECIAL NOTES

1. This subroutine performs a straightforward bubble sort, using the FORTRAN FLD Function for halfword access.

2. H2SORT is called by JOBIND.

SUBROUTINE IDENTIFICATION

Name INSERT (Pack Characters Into Next Array Locations)

Language FORTRAN V

Date Jun 73

Programmer Stacey/Weissmuller

FUNCTION

INSERT packs a given number of the leftmost characters from a word into the next available character positions of a COMMON array up to 1000 words long.

ENTRY POINTS

1. INSERT

CALLING SEQUENCE

1. CALL INSERT (KTFW,KTFC,LFW,NFC)

a. Inputs

(1) KTFW is the integer subscript of the next COMMON array lotation.

(2) KTFC is the integer count of characters filled (Ø-5) in that location.

(3) LFW contains the characters to be inserted, left justified.

(4) NFC is the integer count of characters to be inserted (1-6).

b. Outputs

(1) The specified characters are appended to the COMMON array, and

(2) KTFW and KTFC are updated, OR

(3) If KTFW becomes greater than 1000, the run aborts after a diagnostic.

SPECIAL NOTES

1. This subroutine is highly specialized for use by CPI and FMTGEN to create FORTRAN formats from CODAP Format Cards. As a typical example, suppose words 6 and 7 of the COMMON (format) array contained the string '12Xb̸b̸b̸ b̸b̸b̸b̸b̸b̸' and LFW contained ',13I2b̸' (so that KTFW = 6, KTFC = 3, and NFC = 5). After a call to INSERT, KTFW = 7, KTFC = 2, and the two array words contain the string '12X,13 I2b̸b̸b̸b̸'.
2. The subroutine is not machine dependent, except that a 6-character word is assumed, so little or no conversion is required.

SUBROUTINE IDENTIFICATION

Name IRANF (Initialize Random Number Function)

Language FORTRAN V

Date Feb 74

Programmer Weissmuller

FUNCTION

IRANF has two entry points. The first entry point selects and prints a seed for the random number function, RANF. The selected seed is a modification of the FIELDATA date and time. The second entry point is called by RANF and keeps track of updating the seed from IRANF.

ENTRY POINTS

1. IRANF

2. GETRAN

CALLING SEQUENCE

1. CALL IRANF

   a. Inputs - None

   b. Outputs

      (1) A seed is generated for RANF and GETRAN and,

      (2) The value of the seed is printed in octal format.

2. CALL GETRAN (ANS)

   a. Inputs

      (1) ANS is for output only.

   b. Outputs

      (1) ANS is a floating point random number between Ø.Ø and 1.Ø.

SPECIAL NOTES

1. Other than using ERTRAN, this is a rather standard FORTRAN subroutine.

SUBROUTINE IDENTIFICATION

Name	JDCOPY (Job Description File Copy)
Language	FORTRAN V
Date	Nov 73
Programmer	Weissmuller

FUNCTION

JDCOPY will copy the Job Description file from FORTRAN unit 27 onto unit 25. Unit 25 will contain all previous Job Descriptions and will be positioned to receive additional descriptions. This routine is called after CYCLES in SETUP9.

ENTRY POINTS

1. JDCOPY

CALLING SEQUENCE

1. CALL JDCOPY

   a. Inputs

      (1) None

   b. Outputs

      (1) A complete copy of all Job Descriptions will be written onto unit 25, or

      (2) The run will error terminate on sequence error.

SPECIAL NOTES

1. This routine is standard FORTRAN except for the PARAMETER statement for the variable dimensions which may easily be replaced.

2. This routine appears only in SETUP9 which is called from either JOBSPC or JOBGRP, as they are currently the only programs which can expand the Job Description file.

SUBROUTINE IDENTIFICATION

Name	JDEOF (Find Job Description File Mark)
Language	FORTRAN V
Date	Nov 73
Programmer	Weissmuller

FUNCTION

If two programs attempt to expand the Job Description file within a single run, the first program will call JDCOPY to form a new copy of the file and all subsequent programs will call JDEOF which should simply position the new Job Description to receive additional reports. Since JOBSPC and JOBGRP are the only two programs capable of calling JDEOF and they are rarely ever stacked into a single run, this subroutine is not adequately checked out.

ENTRY POINTS

1. JDEOF

CALLING SEQUENCE

1. CALL JDEOF
   a. Inputs
      (1) None
   b. Outputs
      (1) The new copy of the Job Description is rewound and read until a file mark is found, then backspaced over the file mark.

SPECIAL NOTES

1. This subroutine is standard FORTRAN.
2. The new copy of the Job Description file must be on FORTRAN unit 25.

SUBROUTINE IDENTIFICATION

Name	LASCMP (Logical Alphanumeric String Comparison)
Language	FORTRAN V
Date	Jun 73
Programmer	Weissmuller

FUNCTION

LASCMP compares a string of FIELDATA characters against a range of FIELDATA characters. If the test string is within the range, the function returns a 1, otherwise a zero is returned.

ENTRY POINTS

1. LASCMP (Function: returns binary integer 1 or Ø)

CALLING SEQUENCE

1. ISW = LASCMP(ITEST,LOW,IHI,NWORDS)

   a. Inputs

      (1) ITEST is a FIELDATA character string NWORDS long.

      (2) LOW is a FIELDATA character string NWORDS long.

      (3) IHI is a FIELDATA character string NWORDS long.

      (4) NWORDS is a binary integer giving the length in words of all arrays.

      (5) ISW is for output only.

   b. Outputs

      (1) ISW = 1 if the string ITEST is greater or equal to LOW and less than or equal to IHI in the FIELDATA collating sequence. ISW = Ø if ITEST is not in that range.

SPECIAL NOTES

1. This version of LASCMP is a FORTRAN V rewrite of an IBM 7040 assembly language subroutine. Results are dependent on the collating sequence of characters on the machine used.

2. This subroutine uses the FORTRAN V FLD function on half-word boundaries to evaluate the alphanumeric ranges.

SUBROUTINE IDENTIFICATION

Name LINK (FORTRAN V Linkage to UNIVAC Sort Package)

Language 1100 Assembler

Date --

Programmer UNIVAC

FUNCTION

LINK is a FORTRAN V - EXEC 8 interface which allows FORTRAN to directly call the UNIVAC Sort Package. This write-up and any symbolic listings are provided for reference only.

ENTRY POINTS

1. SBUILD (Called from TABL3(entry = SOPEN3))
2. SRREL
3. SSORT
4. SRRET

CALLING SEQUENCE

1. CALL SRREL (IMAGE,NWORDS)

   a. Inputs

      (1) IMAGE is an array containing the record to be sorted.

      (2) NWORDS is the length in words of the sort record.

   b. Outputs

      (1) This entry point releases one record at a time to the sort package. No action is taken until SSORT is called.

2. CALL SSORT

   a. Inputs - None

b. Outputs

(1) All records which were passed via the SRREL entry are sorted in accord with the control information established with SOPEN3. (See TABL3) Though the records are sorted upon return from this entrv point, they are not useable until retrieved via SRRET.

3. CALL SRRET (IMAGE,NWORDS,$EOF)

a. Inputs

(1) EOF is the FORTRAN statement number to jump to if an end of file condition is detected on the sorted file.

(2) IMAGE and NWORDS are for output only.

b. Outputs

(1) Either, IMAGE will contain the next record from the sorted file, and

(2) NWORDS will be the length in words of IMAGE, OR

(3) Upon reaching an end-of-file, control will jump to the FORTRAN statement number designated by EOF.

SPECIAL NOTES

1. This subroutine is very machine dependent, but its usage is restricted to the subroutine FSORT. If FSORT is rewritten for the user's installation, this routine and TABL3 will not be needed.

SUBROUTINE IDENTIFICATION

Name	LISORT (Sort Pointer Array into Ascending Sequence)
Language	FORTRAN V
Date	Jul 73
Programmer	Weissmuller

FUNCTION

LISORT accepts two arrays, an array of values and an array of pointers (from 1 to NWORDS) and reorders the pointer arrav into an ascending sequence based on the array of values. The array of values remains unchanged.

ENTRY POINTS

1. LISORT

CALLING SEQUENCE

1. CALL LISORT (NWORDS,ARRAY,IPOINT)

   a. Inputs

      (1) NWORDS is a binary integer indicating the number of words in the arrays ARRAY and IPOINT.

      (2) ARRAY is an array of floating point or binary integer values.

      (3) IPOINT is an array of binary integers in ascending sequence, usually 1 through NWORDS.

   b. Outputs

      (1) IPOINT will be re-sequenced such that the contents of IPOINT(1) will be the subscript in ARRAY of the smallest val ιe in ARRAY.

SPECIAL NOTES

1. This subroutine is standard FORTRAN.

2. IPOINT may contain values such as 2ØØ through 3ØØ if only ARRAY(2ØØ) through ARRAY(3ØØ) are to be sorted. This technique is used to sort tasks independently from duties even though the values for both are in the same array in some programs.

3. See LSORT for descending sequence.

SUBROUTINE IDENTIFICATION

Name	LSORT (Sort Pointer Array into Descending Sequence)
Language	FORTRAN V
Date	May 73
Programmer	Weissmuller

FUNCTION

LSORT accepts two arrays, an array of values and an array of pointers (from 1 to NWORDS) and reorders the pointer array into a descending sequence based on the array of values. The array of values remains unchanged.

ENTRY POINTS

1. LSORT

CALLING SEQUENCE

1. CALL LSORT (NWORDS,ARRAY,IPOINT)

   a. Inputs

      (1) NWORDS is a binary integer indicating the number of words in the arrays ARRAY and IPOINT.

      (2) ARRAY is an array of floating point or binary integer values.

      (3) IPOINT is an array of binary integers in ascending sequence, usually 1 through NWORDS.

   b. Outputs

      (1) IPOINT will be re-sequenced such that the contents of IPOINT(1) will be the subscript in ARRAY of the largest value in ARRAY.

SPECIAL NOTES

1. This subroutine is standard FORTRAN.

2. IPOINT mav contain values such as 2ØØ through 3ØØ if onlv ARRAY(2ØØ) through ARRAY(3ØØ) are to be sorted. This technique is used to sort tasks independently from duties even though the values for both are in the same array in some programs.
3. See LISORT for ascending sequence.

SUBROUTINE IDENTIFICATION

Name MSKOP1 (Mask Operations: Type 1)

Language FORTRAN V

Date Oct 73

Programmer Weissmuller

FUNCTION

MSKOP1 is a set of subroutines designed to handle several MASK arrays from a Job Description File. These MASK arrays have 1 bit for each case in a study. If a particular case is in this group, the corresponding bit is equal to 1, otherwise it is zero. The first three entry points are designed to create a mask array as is done in JOBGRP and JOBSPC. The fourth entry point is designed to use existing mask arrays.

ENTRY POINTS

1. ZERMSK
2. SETMSK
3. ERSMSK
4. CHKMSK

CALLING SEQUENCE

1. CALL ZERMSK (NCASE,MASK,NWORDS)

   a. Inputs

      (1) NCASE is a binary integer which is a count of the number of cases in the study.

      (2) MASK is an array to become a mask array.

      (3) NWORDS is for output only.

b. Outputs

(1) NWORDS is a binary integer which is a count of the number of full 36 bit computer words required to contain the mask array for NCASE cases. Since the sign bit of the computer word is not used NWORDS = (NCASE + 34)/35 and truncated to the integer number.

(2) MASK will contain zeroes in the first NWORDS locations, and is ready to receive a mask array.

2. CALL SETMSK (ICASE,MASK)

a. Inputs

(1) ICASE is a binary integer denoting the sequence number of this case. (1 to NCASE)

(2) MASK is an array which is being made into a mask array.

b. Outputs

(1) The bit position in MASK identifying case number ICASE is set to 1 signifying this case is a member of this grouo.

3. CALL ERSMSK (ICASE,MASK)

a. Inputs

(1) ICASE is a binary integer denoting the sequence number of this case. (1 to NCASE)

(2) MASK is a mask array.

b. Outputs

(1) The bit position in MASK identifying case number ICASE is set to Ø signifying this case is NOT a member of this group.

4. CALL CHKMSK (ICASE,MASK,$NOT)

   a. Inputs

      (1) ICASE is a binary integer denoting the sequence number of this case. (1 to NCASE).

      (2) MASK is an existing mask array.

      (3) NOT is a FORTRAN statement to jump to if the bit in MASK corresponding to ICASE is not equal 1.

   b. Outputs

      (1) Either, control will return to the statement following the calling sequence if ICASE is in the group in question, <u>or</u>

      (2) Control will pass to FORTRAN statement number NOT if ICASE is <u>NOT</u> in the group.

SPECIAL NOTES

1. This subroutine uses the FORTRAN V FLD function on the left of an equals sign and will have to be rewritten into the user's assembly language.

2. While this subroutine may test or alter several mask arrays, MSKOP2 is designed to access only one mask array.

3. The COMPILER(FLD=ABS,R) reverses the bit counter in the FLD function from a left to right count (Ø-35) to a right to left count (35-Ø)

SUBROUTINE IDENTIFICATION

<u>Name</u> MSKOP2 (Mask Operations: Type 2)

<u>Language</u> FORTRAN V

<u>Date</u> Oct 73

<u>Programmer</u> Weissmuller

FUNCTION

MSKOP2 is a set of subroutines designed to retrieve and use a single mask array from a Job Description File.

ENTRY POINTS

1. GETMSK
2. GETMSM
3. USEMSK

CALLING SEQUENCE

1. CALL GETMSK (ISEQ,ID1,ID2,MATRIX,$ERR)

   a. Inputs

      (1) ISEQ is a binary integer which is a unique identification number for each report or group. This number is found in card columns 76-80 of the Request card.

      (2) ID1 and ID2 together comprise an 8 character internal identification code for all files under a given study. These two words are the first two words of the Communications Region.

      (3) MATRIX is a 6 character identification code found in word 6 of the Communications Region. MATRIX is blank if the input file is a History file and is equal to either 'TASKb̸b̸' or 'TIMEb̸b̸' if a KPATH file is the input.

(4) ERR is the FORTRAN statement number in the calling program to jump to if the group identified as number ISEQ is not found or if ID1, ID2, or MATRIX fails to match the Job Description file used.

b. Outputs

(1) Either, the group's mask array is retrieved and ready to be used by USEMSK, <u>OR</u>

(2) Control is passed to FORTRAN statement ERR.

2. CALL GETMSM(ISEQ,ID1,ID2,MATRIX,$ERR,NMEM,IDREPT,ITITLE)

a. Inputs

(1) Same as above.

(2) NMEM, IDREPT, and ITITLE are for output only.

b. Outputs

(1) Same as above, plus

(2) NMEM is a binary integer count of the number of members in the specified group.

(3) IDREPT is a 6 character FIELDATA identification of the requested group.

(4) ITITLE is an 8 word FIELDATA array which contains the title of the Report which first identified this group.

3. CALL USEMSK (ICASE,$NOT)

a. Inputs

(1) ICASE is a binary integer which is the sequence number of a case on either the History or KPATH file.

(2) NOT is a FORTRAN statement number to jump to if case number ICASE is not in the group specified in GETMSK or GETMSM.

b. Outputs

(1) Either, control will return to the statement following the call to USEMSK if ICASE is in the group, OR

(2) Control will jump to FORTRAN statement number NOT if ICASE is not in the group.

SPECIAL NOTES

1. This subroutine uses the FORTRAN V FLD function and a COMPILER directive to change the bit counter to a Right to Left count.

2. This routine is limited to processing a single mask array at a time. For simultaneously using multiple MASKS see MSKOP1.

SUBROUTINE IDENTIFICATION

Name	NEXREL (Mount Next Reel)
Language	FORTRAN V
Date	Sep 74
Programmer	Weissmuller

FUNCTION

NEXREL will free the tape file currently assigned and associated with a specific FORTRAN unit, then assign another reel of a file to the same tape drive.

ENTRY POINTS

1. NEXREL
2. NEXREM

CALLING SEQUENCE

1. CALL NEXREL (NAME,IUSE)

   a. Inputs

      (1) NAME is a three word array where words 1 & 2 are the filename in an A6,A4 format and word 3 is the negative file cycle (backup number) in a 1A1 format. Word 3 = blank will assign cycle Ø (the most recent copy). All words are in FIELDATA.

      (2) IUSE is a binary integer representing the FORTRAN unit which is currently associated with a tape file to be freed. Also, the file specified by NAME above is to be mounted on the tape drive currently used by the file to be released or freed, then associated with this FORTRAN unit.

   b. Outputs

      (1) Either the tape reels on the physical tape drive are switched and the FORTRAN unit IUSE is associated with the file NAME, <u>OR</u>

(2) The run aborts due to an invalid NAME with the appropriate diagnostics.

2. CALL NEXREM (NAME,IUSE)

   a. Inputs

      (1) The same as for NEXREL.

   b. Outputs

      (1) The same as for NEXREL except the new file assigned will be an "M" or 3-reel cycled file.

SPECIAL NOTES

1. As part of the file assignment package this subroutine will probably be replaced or obsoleted on any non-UNIVAC system.

SUBROUTINE IDENTIFICATION

Name	NOHEAD (Change Page Boundaries)
Language	1100 Assembler
Date	Sep 73
Programmer	Weissmuller

FUNCTION

NOHEAD was written to suppress the EXEC 8 heading lines and allow CODAP programs to define their own page boundaries. Generally speaking CODAP programs provide 5 blank lines at the top of each page instead of the default 6 lines. Individual programs vary as to the number of blank lines at the bottom of the page. The program DIAGRM, for example, suppresses all page boundaries as the diagram may be several pages long.

ENTRY POINTS

1. NOHEAD
2. MARGIN
3. NOPAGE

CALLING SEQUENCE

1. CALL NOHEAD
   a. Inputs - None
   b. Outputs
      (1) Automatic page overflow occurs at line 66 (the tear strip) and skips 5 lines at the top of the next page.
2. CALL MARGIN
   a. Inputs - None

b. Outputs

(1) Automatic page overflow is reset to the system standard, i.e. leave six blank lines at the top and 3 blank lines at the bottom of a 66 line page.

3. CALL NOPAGE

a. Inputs - None

b. Outputs

(1) All automatic page overflow is suppressed. When forced, a new page boundary is the line after the tear strip, otherwise no boundaries are recognized.

SPECIAL NOTES

1. This is a very machine dependent subroutine, however, an assembler subroutine at the user's installation may easily replace this one.

SUBROUTINE IDENTIFICATION

Name	NRAND (Random Number Generator)
Language	1100 Assembler
Date	Nov 73
Programmer	Hutchinson

FUNCTION

NRAND will accept a seed value and return a random number and an updated seed value.

ENTRY POINTS

1. NRAND

CALLING SEQUENCE

1. CALL NRAND (ISEED,PROB)

   a. Inputs

      (1) ISEED is a variable which may contain any value to be used as a seed to the random number generator. NOTE: The value of ISEED is updated by the subroutine and therefore a constant may not be used. Within CODAP, some form of the time of day in FIELDATA is generally used as a seed.

      (2) PROB is for output only.

   b. Outputs

      (1) The value of ISEED is modified such that the next call will yield a new random number.

      (2) PROB will be a random number in floating point notation in the range Ø.Ø to 1.Ø.

SPECIAL NOTES

1. Though this subroutine is in Assembler, it ought to be easy to rewrite for any machine.

2. This routine is used in conjunction with IRAND and SRAND within RANDOM tor selecting random subsamples of a specified size.

SUBROUTINE IDENTIFICATION

Name OMSG (Operator Messages)

Language 1100 Assembler

Date Jan 74

Programmer Rogers

FUNCTION

OMSG provides a method for a program to communicate with the system console operator and visa versa. This routine is used by a few programs to provide additional information to the operator. It also has the capability to receive operator responses.

ENTRY POINTS

1. OMSG
2. OMSGW

CALLING SEQUENCE

1. CALL OMSG (NWORDS,MESAGE)

   a. Inputs

      (1) NWORDS is a binary integer count of the number of words in the MESAGE array.

      (2) MESAGE is an array which contains the FIELDATA message to be displayed on the operator's console.

   b. Outputs

      (1) The first NWORDS of MESAGE is displayed on the operator's console.

2. CALL OMSGW (NOUT,MESOUT,NIN,MESIN)

   a. Inputs

      (1) NOUT is a binary integer count of the number of words in the MESOUT array.

(2) MESOUT is an array which contains a FIELDATA message which solicits an operator response.

(3) NIN is a variable which contains a binary integer which is a limit to the number of words of response that the operator may return. NOTE: NIN is set equal to actual number of words returned and therefore a constant must not be used for NIN.

(4) MESIN is an array for output only.

b. Outputs

(1) The first NOUT words of MESOUT is displayed on the operators console and the routine waits for a response. (Hence, the "W" suffix)

(2) Upon receiving a response of NIN words or less, NIN is set equal to the number of words returned in the MESIN array. If an operator attempts to enter more than NIN words, his message is erased before he completes it.

SPECIAL NOTES

1. This subroutine is very machine dependent, but its function is not essential to the CODAP system. It may be removed with little consequence.

SUBROUTINE IDENTIFICATION

<u>Name</u>	OMSGO (Operator's Messages: Other Console)
<u>Language</u>	1100 Assembler
<u>Date</u>	Jan 74
<u>Programmer</u>	Rogers

FUNCTION

OMSGO is identical to OMSG except all messages are directed to a secondary console.

ENTRY POINTS

1. OMSGO
2. OMSGOW

CALLING SEQUENCE - See OMSG

SUBROUTINE IDENTIFICATION

Name	OVRFLO (DIST2X Flip-Flop Disk/Table Merge)
Language	FORTRAN V
Date	Dec 73
Programmer	Barton

FUNCTION

When a core table of DIST2X output information is filled, OVRFLO merges the table into a flip-flop pair of mass storage files.

ENTRY POINTS

1. OVRFLO

CALLING SEQUENCE

1. CALL OVRFLO (MINF,MAXF,N,LIMIT,KSW,ITABLE,LIN,LOU)

   a. Inputs

      (1) MINF is a decrement index (the table is scanned backwards).

      (2) MAXF is the maximum number of standard 5-word entries in the table.

      (3) N points to the first word of the current entry.

      (4) LIMIT is a constant.

      (5) KSW is a flag.

      (6) ITABLE is the core table.

      (7) LIN and LOU are the FORTRAN unit numbers for the flip-flop files.

   b. Outputs

      (1) The contents of ITABLE are merged with the contents of unit LIN onto unit LOU, and the units are then reversed.

SPECIAL NOTES

1. This subroutine is an expedient resulting from the conversion of DIST2X from the IBM 7040 to the UNIVAC 1108. Inline coding was removed to a subroutine to reduce the size of the main program and to eliminate a bad transfer from a FORTRAN DO loop. Obviously, OVRFLO is highly specialized, but reduces the machine dependence of DIST2X.

SUBROUTINE IDENTIFICATION

<u>Name</u> PRTDIC (Print Variable Dictionarv)

<u>Language</u> FORTRAN V

<u>Date</u> May 73

<u>Programmer</u> Stacey

FUNCTION

PRTDIC will read a prepositioned History or KPATH file and print a standard format Variable Dictionary. This routine is used in PRDICT, VARGEN, and PROGEN.

ENTRY POINTS

1. PRTDIC

CALLING SEQUENCE

1. CALL PRTDIC (IUNIT,IOUT,NHEAD,IHEAD,NDICT)

   a. Inputs

      (1) IUNIT is a binary integer representing the FORTRAN unit associated with the History or KPATH input file.

      (2) IOUT is a binary integer representing the FORTRAN unit associated with the output file, generally = 6 for the printer or = 26 for the Report file.

      (3) NHEAD is a binary integer count of the number of 14 word FIELDATA heading cards to be printed at the top of the first page.

      (4) IHEAD is an array of 14 word FIELDATA heading cards to be printed.

(5) NDICT is a binary integer count of the number of blocks of dictionary information on the History or KPATH file. This number is the fourth word of the Communications Region

b. Outputs

(1) The Variable Dictionary will be printed or written to the Report file.

SPECIAL NOTES

1. This is a standard FORTRAN subroutine.

SUBROUTINE IDENTIFICATION

Name	RANDOM (Random Subsample Selector)
Language	FORTRAN V
Date	Nov 73
Programmer	Hutchinson/Weissmuller

FUNCTION

RANDOM is a subroutine with two entry points. The first entry establishes the random number generator seed, the size cf the total population and the exact number of cases to be randomly selected. The second entry simply returns a 1 or a Ø depending on whether the current case ought to be selected or not.

ENTRY POINTS

1. IRAND
2. SRAND

CALLING SEQUENCE

1. CALL IRAND(ISEED,NPOP,NSAMP)

   a. Inputs

      (1) ISEED is any value to be used as a seed value for the random number generator (NRAND). The time of day in FIELDATA is normally used.

      (2) NPOP is a binary integer count of the total population size from which the random subsample is to be selected.

      (3) NSAMP is a binary integer count of the number of cases to be randomly selected.

   b. Outputs

      (1) The random subsample selection routine is initialized and SRAND may be called.

2. CALL SRAND(INSAMP)

   a. Inputs

      (1) INSAMP is for output only.

   b. Outputs

      (1) INSAMP is a binary integer variable which is set equal to -1 if the current case is <u>not</u> to be selected or is set equal to +1 if the current case <u>is</u> to be selected.

SPECIAL NOTES

1. This subroutine is standard FORTRAN.

SUBROUTINE IDENTIFICATION

Name	RANF (Random Number Function)
Language	FORTRAN V
Date	Feb 74
Programmer	Weissmuller

FUNCTION

A function reference to RANF will be replaced bv a random floating point value in the range from Ø.Ø to 1.Ø inclusively. Note: An initialization routine IRANF must be called prior to the first reference to RANF.

ENTRY POINTS

1. RANF (Function: floating point value (Ø.Ø-1.Ø)

CALLING SEQUENCE

1. PROB = RANF(X)

   a. Inputs

      (1) X is a dummy variable. It is neither used nor altered.

      (2) PROB is for output only.

   b. Outputs

      (1) PROB will be set equal to a random floating point value between Ø.Ø and 1.Ø inclusively.

SPECIAL NOTES

1. This is a standard FORTRAN function.
2. Prior to using RANF, the routine IRANF must be called.

SUBROUTINE IDENTIFICATION

<u>Name</u>	RANSEQ (Randomly Sequence an Array)
<u>Language</u>	FORTRAN V
<u>Date</u>	Oct 74
<u>Programmer</u>	Weissmuller

FUNCTION

RANSEQ will accept an input array of up to 300 items in length and randomly resequence the array.

ENTRY POINTS

1. RANSEQ

CALLING SEQUENCE

1. CALL RANSEQ (NWORDS, IARRAY)

   a. Inputs

      (1) NWORDS is a binary integer specifying the number of words in IARRAY.

      (2) IARRAY is an array name which contains elements to be resequenced.

   b. Outputs

      (1) IARRAY will be randomly reordered.

SPECIAL NOTES

1. This subroutine uses NRAND to generate random numbers, and H2SORT to sort an internal pointer array based on the random numbers.

SUBROUTINE IDENTIFICATION

Name REPEND (Report End and Request Card Punch)

Language FORTRAN V

Date Nov 73

Programmer Weissmuller

FUNCTION

REPEND updates the 'last report sentinel' on FORTRAN unit 2 and punches a Request card for the report just written to the Report file, as well as initializing the Report file for the next report.

ENTRY POINTS

1. REPEND

CALLING SEQUENCE

1. CALL REPEND (ISTUDY,IDREPT,ITITLE,NREP)

   a. Inputs

      (1) ISTUDY is a 4 character FIELDATA code for the account or study.

      (2) IDREPT is a 6 character FIELDATA identification for the Report just written.

      (3) ITITLE is an 8 word FIELDATA title that is printed on the first line of the report.

      (4) NREP is for output only.

   b. Outputs

      (1) A Request card is punched with the study, the report ID, the report title, and the report number.

      (2) The report number is updated on FORTRAN unit 2.

      (3) NREP is a binary integer which is the updated report number.

      (4) The next 'BEGIN REPORT' NREP sentinel is written to the Report file.

SPECIAL NOTES

1. This subroutine is coded in standard FORTRAN and is used by every program which writes to the Report file. (JOBIND uses a special version called REPEN1.)

SUBROUTINE IDENTIFICATION

Name	REPEN1 (Report End and Request Card Punch)
Language	FORTRAN V
Date	Nov 73
Programmer	Weissmuller

FUNCTION

REPEN1 updates the 'last report sentinel' on FORTRAN unit 2 and punches a Request card for the report just written to the Report file as well as initializing the Report file for the next report. This subroutine is used only in JOBIND and is a modified version of REPEND.

ENTRY POINTS

1. REPEN1

CALLING SEQUENCE

1. CALL REPEN1 (ISTUDY,IDREPT,ITITLE,IVX,JREP,NREP)

   a. Inputs

      (1) ISTUDY is a 4 character FIELDATA code for the account or study.

      (2) IDREPT is a 6 character FIELDATA identification for the Report just written.

      (3) ITITLE is a 7 word FIELDATA title that is printed on the first line of the report.

      (4) IVX is a 'visual control' variable value. It may be up to 6 FIELDATA characters.

      (5) JREP is a binary integer which identifies the report number associated with the group that serves as input.

b. Outputs

(1) A Request card is punched with the study, the report ID, the report title, the value of the visual ID variable (generally case control number), the report number of the input group and the report number for this individual job description.

(2) The report number is updated on FORTRAN unit 2.

(3) NREP is a binary integer which is the updated report number.

(4) The next 'BEGIN REPORT' NREP sentinel is written to the Report file.

SPECIAL NOTES

1. This subroutine is coded in standard FORTRAN.

SUBROUTINE IDENTIFICATION

<u>Name</u>	RESDRV (Reserve a Tape Drive)
<u>Language</u>	FORTRAN V
<u>Date</u>	Nov 73
<u>Programmer</u>	Weissmuller

FUNCTION

RESDRV asks the operator to reserve a tape drive then indicate the drive number to the program. This is used when creating Report and Job Description files so that all three copies will onlv use one tape drive. Alternate methods are being examined and this routine may soon become obsolete. See FREES.

ENTRY POINTS

1. RESDRV

CALLING SEQUENCE

1. CALL RESDRV (IUNIT)

   a. Inputs

      (1) IUNIT is for output only.

   b. Outputs

      (1) IUNIT is a binary integer denoting the tape drive reserved for this program.

SPECIAL NOTES

1. This subroutine is part of the file handling package and will probably be replaced.

SUBROUTINE IDENTIFICATION

Name	RFILE (Random Access File for OVRLAP)
Language	1100 Assembler
Date	May 74
Programmer	Rogers

FUNCTION

RFILE is a direct access file designed for the OVRLAP program. This routine establishes OVRLAP's input as a "READ ONLY" file and prevents the entire file from being checkpointed as well as reducing the overhead checking inherent in FORTRAN V's direct access coding.

ENTRY POINTS

1. DEFINE
2. WRITER
3. CLOSER
4. READR

CALLING SEQUENCE

1. CALL DEFINE (MAXREC,MAXLRL,ISTUDY)

   a. Inputs

      (1) MAXREC is a binary integer count of the maximum number of records to be written to thib file.

      (2) MAXLRL is a binary integer indicating the maximum length in words of the longest record.

      (3) ISTUDY is a 4 character FIELDATA code for the account or study. This is used to generate filenames.

   b. Outputs

      (1) A direct access file is assigned and sufficient mass storage area is secured. WRITER may now be called.

2. CALL WRITER (IREC,NWORDS,IMAGE)

   a. Inputs

      (1) IREC is a binary integer index number for the record to be written.

      (2) NWORDS is a binary integer count of the number of words in the record array IMAGE.

      (3) IMAGE is the record array.

   b. Outputs

      (1) The contents of IMAGE is written as record number IREC in the direct access file.

3. CALL CLOSER

   a. Inputs - None

   b. Outputs

      (1) The direct access file established by the call to DEFINE and loaded with information via the calls to WRITER is closed and reassigned as a "READ-ONLY" file. The file is now ready to be accessed by OVRLAP via the calls to READR.

4. CALL READR (IREC,NWORDS,IMAGE)

   a. Inputs

      (1) IREC is the binary integer index number of the record to be read. Note: DEFINE, WRITER and CLOSER must be called prior to the first call to READR.

      (2) NWORDS is for output only.

      (3) IMAGE is an array for output only.

   b. Outputs

      (1) NWORDS is a binary integer count of the number of words in the record which was stored in IMAGE.

(2) IMAGE is an array which contains the record.

SPECIAL NOTES

1. This subroutine is very machine dependent. It was necessitated bv properties of the checkpointing system and the excessive error checking of FORTRAN. This routine, along with DISK and inline direct access FORTRAN V statements, will undoubtedly be replaced at other installations.

SUBROUTINE IDENTIFICATION

Name ROSTER (Roster Control Cards and Error Messages)

Language FORTRAN V

Date Nov 73

Programmer Weissmuller

FUNCTION

ROSTER is a set of subroutines designed to print control cards and error messages.

ENTRY POINTS

1. RESET
2. PRINTC
3. ERRMSG

CALLING SEQUENCE

1. CALL RESET
   a. Inputs - None
   b. Outputs
      (1) Writes a new control card listing page and resets counters.
2. CALL PRINTC
   a. Inputs - None
   b. Outputs
      (1) Rosters the last card read bv doing a re-read (FORTRAN unit Ø or 3Ø).
      (2) Automatic page overflow if required.
3. CALL ERRMSG (NWORDS,IMSG)
   a. Inputs
      (1) NWORDS is a binary integer count of the number of words in the error message array IMSG.

(2) IMSG is a FIELDATA array of up to 9 words in length which contains an error message. IMSG is sometimes passed as a literal delimited by quote marks.

b. Outputs

(1) The contents of IMSG are printed single spaced under the last card rostered.

SPECIAL NOTES

1. This subroutine uses FORTRAN V's re-read ability on FORTRAN unit Ø or 3Ø. The re-read is accomplished by re-accessing FORTRAN's buffer for the card file and may be coded in assembly language if necessary.

SUBROUTINE IDENTIFICATION

Name RPCOPY (Report File Copy)

Language FORTRAN V

Date Nov 73

Programmer Weissmuller

FUNCTION

RPCOPY will copy the Report file from FORTRAN unit 28 to unit 26. Unit 26 will be left positioned to receive additional reports. This subroutine is called after CYCLES in SETUP and SETUP9.

ENTRY POINTS

1. RPCOPY

CALLING SEQUENCE

1. CALL RPCOPY

   a. Inputs

      (1) None

   b. Outputs

      (1) FORTRAN unit 26 will contain all previous reports and will be positioned to receive more.

SPECIAL NOTES

1. This routine is standard FORTRAN with calls to the TREAD subroutine. (REWIND is an entry point into TREAD).

SUBROUTINE IDENTIFICATION

<u>Name</u>	RPEOF (Reposition Report File for Multiple Programs)
<u>Language</u>	FORTRAN V
<u>Date</u>	Nov 73
<u>Programmer</u>	Weissmuller

FUNCTION

RPEOF will reposition a Report file if several program executions write to a Report file within a single run. The first program in the runstream will call RPCOPY and create a new copy of the Report file. All subsequent programs will call RPEOF and set the new copy to receive additional reports.

ENTRY POINTS

1. RPEOF

CALLING SEQUENCE

1. CALL RPEOF (IUNIT,NREP)

   a. Inputs

      (1) IUNIT is a binary integer denoting the FORTRAN unit number of the Report file. (generally IUNIT = 26)

      (2) NREP is for output only.

   b. Outputs

      (1) NREP is a binary integer indicating the number of the last report sentinel on the Report file.

      (2) The Report file is positioned to receive additional reports.

SPECIAL NOTES

1. This subroutine is standard FORTRAN.

SUBROUTINE IDENTIFICATION

Name	RPINDX (Report File Index)
Language	FORTRAN V
Date	Jul 73
Programmer	Weissmuller

FUNCTION

RPINDX will print an index of the Report file assigned to FORTRAN unit 28.

This subroutine is called when option 2 of the EXTRCT program is specified.

ENTRY POINTS

1. RPINDX

CALLING SEQUENCE

1. CALL RPINDX

   a. Inputs - None

   b. Outputs

      (1) A printed index of the Report file assigned to FORTRAN unit 28.

SPECIAL NOTES

1. This subroutine is standard FORTRAN.

SUBROUTINE IDENTIFICATION

<u>Name</u> RUNID (Retrieve Run ID and Study ID)

<u>Language</u> 1100 Assembler

<u>Date</u> Apr 74

<u>Programmer</u> Weissmuller

FUNCTION

RUNID will retrieve the run ID and the account or study ID. This routine is used only in INPSTD and is not at all essential even there.

ENTRY POINTS

1. RUNID

CALLING SEQUENCE

1. CALL RUNID (IDRUN,ISTUDY)
   - a. Inputs
     - (1) IDRUN and ISTUDY are for output only.
   - b. Outputs
     - (1) IDRUN is a 6 character FIELDATA identification of a run. By installation policy the first four characters are an individual's ID code.
     - (2) ISTUDY is a 4 character FIELDATA code for the account or study.

SPECIAL NOTES

1. This is a highly specialized subroutine which is used only in INPSTD for punching a card to be returned to the programmers for timing studies. An attempt is being made to estimate the running time of OVRLAP from information accumulated by INPSTD. This routine and the punched card may easily be eliminated.

SUBROUTINE IDENTIFICATION

Name	SAMSEL (Sample Selection Subroutine)
Language	FORTRAN V
Date	Jun 74
Programmer	Weissmuller

FUNCTION

SAMSEL has two entry points, one for determining the sampling logic and one for actually selecting based on that logic.

ENTRY POINTS

1. SAMPLE
2. SELECT

CALLING SEQUENCE

1. CALL SAMPLE (IDSAMP,IDVAR,NVAR,IERR,IPRT)

   a. Inputs

      (1) VARIABLE INTERACTION CARD - See Sample Selection attachment Section I.

      (2) IDSAMP is an identification code for this sample. (Normally sequential binary integers.)

      (3) IERR is a binary integer error count (normally = Ø).

      (4) IPRT is a binary integer print flag. If IPRT = 1, PRINTC is called and the VARIABLE INTERACIION CARD is rostered. (See ROSTER) If IPRT ≠ 1, no rostering occurs.

      (5) IDVAR is a 9 word array for output only.

      (6) NVAR is for output only.

   b. Outputs

      (1) The VARIABLE INTERACTION CARD is read from the card reader.

(2) The IDVAR array contains a binary integer for each variable ID found on the INTERACTION CARD. Background or VXXX variables are flagged by making them negative. For example, if VØ15 or V15 is encountered, the corresponding element of IDVAR = -15. Conversely, if CØ15 or C15 is found, the proper element of IDVAR = +15.

(3) NVAR is the number of variable IDs found on the INTERACTION CARD. A maximum of 9 is allowed.

(4) IERR is a binary integer which is incremented by 1 for each error detected.

(5) The INTERACTION CARD is rostered if IPRT = 1.

2. CALL SELECT (IDSAMP,LVALS,$REJECT,$NOFIND)

a. Inputs

(1) IDSAMP is an identification code for the desired sample. It must match exactly an IDSAMP created by a call to SAMPLE.

(2) LVALS is an array of binary integers equal to 1 or Ø. A given element of LVALS will equal 1 if the value of the corresponding variable specified in IDVAR is in the required range, and Ø otherwise. The main program is responsible for keeping track of variable ranges and determining whether or not a particular case has data within these ranges. Note: The array LVALS is altered by the subroutine.

(3) REJECT is the FORTRAN statement number to jump to if the current case does not meet the sample restriction requirements.

(4) NOFIND is the FORTRAN statement number to jump to if no match is found for IDSAMP.

b. Outputs

(1) Either, there is a normal return from the subroutine which indicates the case should be added to the sample, and LVALS(1) ≠ Ø, or

(2) Control is returned to FORTRAN statement number REJECT, which means the case is not in the sample and LVALS(1) = Ø, or

(3) Control is returned to FORTRAN statement number NOFIND because no match was found for IDSAMP.

SPECIAL NOTES

1. This routine uses ERTRAN to dynamically assign FORTRAN unit 4 as a temporary storage unit for the logical operations table. Other than that, however, SAMSEL is coded in rather standard FORTRAN.

SAMPLE SELECTION

Section I VARIABLE INTERACTION CARD

Columns

2 Number of variable ranges used. (Maximum of 9) This also specifies the number of cards in Section II.

4-80 Variables and operators optionally nested in parens.

OPERATORS - Two logical operators may be used. These will perform a logical "AND" or a logical "OR". These operators are denoted by the following special characters:

Logical "AND" is denoted by "&".

Logical "OR" is denoted by ":" (Colon).

VARIABLES - The variable names may be abbreviated to least number of significant characters. For example, C007 may be written "C7", V020 may be written "V20", etc.

PARENS - If used, the parens must appear in matched pairs. The use of parens will cause the enclosed expression to be evaluated prior to all lesser or non-enclosed expressions. The normal method of evaluating the logical expressions is to evaluate all expressions containing "&" (which is the logical "AND") from left to right, then evaluate all expressions containing ":" (which is the logical "OR"). In other words, a logical "AND" has precedence over a logical "OR", as is the case in most MATH or LOGIC

Systems. Parens may be used for readability or to override the normal hierarchy.

<u>Section II</u> VARIABLE SPECIFICATION CARD (one card per variable)

Columns

2 1 = The variable is 1-6 characters in size

2 = The variable is 7-12 characters in size

blank = The variable is computed (Cxxx)

4- 7 Variable identification (Vxxx or Cxxx)

9-80 Variable low and high ranges. Values must be left adjusted in their fields. Both low and high must be specified.

If cc 2 is 1: Punch low in cc 9-14, high in cc 15-20.

If cc 2 is 2: Punch low in cc 9-20, high in cc 21-32.

If cc 2 is blank: Same as 2 except values must contain a decimal point.

EXAMPLES

If C007 = Number of non-zero task responses, V003 = Grade.

```
 3  C7&V003:V3
    C007  400.  683.
 1  V003  1  3
 1  V003  4  9
```

Then this setup will select all Airman First and below who perform between 400 and 683 tasks and all Sergeants and above regardless of the number of tasks they perform.

SUBROUTINE IDENTIFICATION

<u>Name</u> SETUP (Standard File Setup and Assignment Routine)

<u>Language</u> FORTRAN V

<u>Date</u> Nov 73

<u>Programmer</u> Weissmuller

FUNCTION

SETUP is the highest level subroutine in the file assignment package. Removing or changing SETUP will obsolete the following subroutines: ASGA, ASGAM, ASGC, ASGCM, AUTORV, CYCLES, DKSTAT, FACREJ, JDCOPY, JDEOF, RPCOPY, RPEOF, STATUS.

Generally, SETUP establishes the relationship between the filenames on the FILENAMES card and corresponding FORTRAN units in the calling seauence. Additional FORTRAN scratch units may be defined which are unrelated to the filenames on the FILENAMES card.

This subroutine, or some similar version, is called by every main program in the CODAP system if any files are involved.

ENTRY POINTS

1. SETUP

CALLING SEQUENCE

1. CALL SETUP (I1,I2,I3,I4,I5,I6,J1,J2,J3,J4,J5,NREP)

   a. Inputs

      (1) I1 through I6 are binary integers in the range (1Ø-28) denoting the FORTRAN unit to be associated with the corresponding filename on the FILENAMES card.

      (2) J1 through J5 are binary integers normally in the range 2Ø-24 denoting FORTRAN mass storage scratch units to be assigned.

      (3) NREP is for output only.

b. Outputs

(1) The FILENAMES card is read from the card reader.

(2) For an I value = 25-28 a Report or Job Description file or files is assigned. If the I value is:

(a) 25: A new copy of the Job Description is created by assigning the most recent copy of the file to FORTRAN unit 27, assigning the oldest copy to unit 25 and copying from 27 to 25. The most recent copy on 27 is released and unit 25 is left mounted to receive additional Job Descriptions. (See CYCLES, JDCOPY, JDEOF). Note: This action is performed onlv in the SETUP9 version, but is included herc for documentation. SETUP9 was created because this copying requires 5000 words of core but can only be called by JOBSPC or JOBGRP.

(b) 26: Similar to the above except the Report file is used. Unit 28 has the most recent copy and 26 becomes the new copy. This action is performed for each run which contains at least one program which writes to the Report file. (See CYCLES, RPCOPY, and RPEOF)

(c) 27: The most recent copy of the Job Description file is assigned for input only. (See ASGAM)

(d) 28: The most recent copy of the Report file is assigned for input only. (See ASGAM)

NOTE: An "M" suffix and cycle number are added to the filename found on the FILENAMES card for all assignments in this range (25-28).

(3) For an I value = 1Ø-14, an input file is assigned and the FORTRAN unit number relationship is established. (See ASGA)

(4) For an I value = 15-19, a new catalogued tape is assigned for output and the FORTRAN unit number relationship is established (See ASGC), or if the filename is equal to 'SCRATCH', a scratch tape with the proper FORTRAN unit designation is assigned.

(5) For an I value = 2Ø-24, a scratch tape with the proper FORTRAN unit will be assigned.

(6) For an I value less than 1Ø (other than Ø) or greater than 28 an error message is printed and the run aborts.

(7) For an I or J value equal to Ø, no action is taken. Note: FORTRAN unit Ø cannot be assigned to a file since the UNIVAC FORTRAN V convention sets FORTRAN units Ø and 3Ø as the reread units. This convention could be overridden, but is not within the CODAP system.

(8) For any value of J, an attempt is made to assign a temporarv mass storage scratch file. If the J value is not in the range 1-29 an error occurs. Moreover, no check is made to see if the I and J values conflict.

(9) NREP is a binary integer set equal to the last report sentinel on the Report file if and only if an I value of 26 is used and a valid Report file was specified on the corresponding field of the FILENAMES card.

SPECIAL NOTES

1. This subroutine will undoubtedly be replaced by any installation which does not have a UNIVAC. This documentation is provided to ease the rewriting involved.

SUBROUTINE IDENTIFICATION

Name SETUP5 (File Assignments for EXTRCT)

Language FORTRAN V

Date Nov 73

Programmer Weissmuller

FUNCTION

SETUP5 is identical to SETUP except an I value in the range 15-19 will create a new 3-reel "M" suffix file rather than a normal single reel file. (Ref SETUP: CALLING SEQUENCE 1b(4) - See ASGCM)

ENTRY POINTS

1. SETUP5

CALLING SEQUENCE

1. CALL SETUP5 (I1,I2,I3,I4,I5,I6,J1,J2,J3,J4,J5,NREP)

   See SETUP

SUBROUTINE IDENTIFICATION

Name SETUP9 (File Assignments for JOBGRP and JOBSPC)

Language FORTRAN V

Date Jun 74

Programmer Weissmuller

FUNCTION

SETUP9 is identical to SETUP except it allows a new copy of the Job Description file to be created. Because the copying procedure requires 5000 additional words of core, this routine was formed from the standard SETUP.

ENTRY POINTS

1. SETUP9

CALLING SEQUENCE

1. CALL SETUP9 (I1,I2,I3,I4,I5,I6,J1,J2,J3,J4,J5,NREP)

   See SETUP

SUBROUTINE IDENTIFICATION

Name	SHIFT (Shift Characters Right Within Words)
Language	FORTRAN V
Date	Dec 73
Programmer	Weissmuller

FUNCTION

SHIFT causes a given number of the leftmost characters of each of a specified number of words in an array to be right justified with preceding blanks within the same words.

ENTRY POINTS

1. SHIFT

CALLING SEQUENCE

1. CALL SHIFT (NCHAR,NWORD,IRAY)

   a. Inputs

      (1) NCHAR is the number of characters, taken from the left, to be shifted right. If NCHAR is not between 1 and 5, the subroutine takes no action.

      (2) NWORD is the number of words, starting with the first word of the array, to be shifted within.

      (3) IRAY is the array (name).

   b. Outputs

      (1) For each word, 1-NWORD, of array IRAY the leftmost NCHAR characters become the rightmost NCHAR characters with blanks preceding. Of course, the original rightmost character(s) are lost.

SPECIAL NOTES

1. SHIFT assumes a 6-character word and used the FORTRAN ENCODE, but is otherwise machine independent.

SUBROUTINE IDENTIFICATION

Name	SISO (Source Input, Source Output Routine)
Language	1108 Assembler
Date	Sep 73
Programmer	Rogers

FUNCTION

SISO is a set of subroutines which allows a program to read, update and/or write elements in a mass storage program file. SISO is only used in PROGEN which generates a FORTRAN V program element. Calls to these routines may be replaced by either punching cards or writing these card images to another mass storage or tape file. This may force PROGEN to become a two step operation: Program generation then, in another run, compilation and excution.

ENTRY POINTS

1. See Attachment.

CALLING SEQUENCE - See Attachment.

LIBZ.SISO (UNIVAC 1108 Assembler Language Subroutine)

Entry Points -- Routines

SPREP	Call the Pre-Processor routine
SPOST	Call the Post-Processor routine
SIOPN	Open the Source Input file
SICLS	Close the Source Input file
SIGET	Get the next Source Input record
SOOPN	Open the Source Output file
SOCLS	Close the Source Output file
SOPUT	Put a record on the Source Output file

Entry Points -- Data

PARTBL	Parameter table used by the Source Input routine and the Source Output routine.

General Information:

The registers used in these routines (AØ-A3, R1) are not saved or restored by the subroutine. X11 is always the linkage register.

These routines all have FORTRAN compatible calling sequences.

Any errors will cause a message to be printed and the run aborted.

SPREP -- Call the Pre-Processor routine

This routine calls the EXEC 8 routine PREPRO to initialize the Parameter Table, PARTBL. It returns a list of the options specified on the processor call card in fieldata, left adjusted, with six characters per word.

```
CALL SPREP (OPTLST)
```

OPTLST -- An array to receive the option list. It should be filled with zeroes or blanks before the call. The specified options are alphabetical, left to right.

SPOST -- Call the Post-Processor routine

This routine calls the EXEC 8 routine POSTPR to reset the assignment status of program files.

```
CALL SPOST
```

SIOPN -- Open the Source Input file

This routine provides initialization functions for getting Source Input records from the input program file.

```
CALL SIOPN
```

SICLS -- Close the Source Input file

This routine provides termination functions for closing the Source Input file.

SIGET -- Get the next Source Input record

This routine returns the next logical record in the input file to the user's area, along with a flag indicating new or removed cards. When End-of File is detected, the length parameter is set to zero.

```
CALL SIGET (REC,LEN,FLAG)
```

REC.....The receiving area for the record.
LEN.....The length in words of the receiving area.
FLAG....The flag indicating new or deleted images in the first three characters. Possible values are:

"NEW" indicating a new image.
"-xx" where xx is a two-digit decimal number indicating the number of images deleted prior to this image. This number cannot exceed 63. If more than 63 images are deleted, it will only show 63.
"b̸b̸b̸" if neither of the above, the value is spaces.

SOOPN -- Open the Source Output file

This routine provides initialization for output to a program file.

```
CALL SOOPN
```

SOCLS -- Close the Source Output file

This routine provides termination functions for closing the Source Output file.

```
CALL SOCLS
```

SOPUT -- Put a record on the Source Output file

This routine transfers an image from the user's area to the source output area.

```
CALL SOPUT (LEN,REC)
```

LEN.....Length of the output image.
REC.....The image area to be output.

SUBROUTINE IDENTIFICATION

<u>Name</u> STATS (Statistics on Task Responses)

<u>Language</u> FORTRAN V

<u>Date</u> May 74

<u>Programmer</u> Barton/Weissmuller

FUNCTION

STATS is a subroutine which was added to INPSTD in order to provide task response statistics used to predict the running time of OVRLAP.

ENTRY POINTS

1. STATS
2. STATSS

CALLING SEQUENCE

1. CALL STATS (NZTSK)

   a. Inputs

      (1) NZTSK is a binary integer count of the number of non-zero task responses made by the current case.

   b. Outputs

      (1) The statistics are accumulated for each call.

2. CALL STATSS (AMEAN,SD,MIN,MAX,ISUM,NCASE)

   a. Inputs

      (1) Successive calls to STATS

      (2) AMEAN, SD, MIN, MAX, ISUM, and NCASE are for output only.

   b. Outputs

      (1) AMEAN is the floating point value representing the mean of all values passed to STATS.

      (2) SD is the floating point value of the standard deviation of the values passed.

(3) MIN is a binary integer equal to the lowest NZTSK passed to STATS.

(4) MAX is a binary integer equal to the highest NZTSK passed to STATS.

(5) ISUM is a binary integer summation of all NZTSK values passed to STATS.

(6) NCASE is a binary integer count of the number of times STATS was called.

SPECIAL NOTES

1. STATS is coded in standard FORTRAN

SUBROUTINE IDENTIFICATION

<u>Name</u>	STATUS (Tape File Status)
<u>Language</u>	1100 Assembler
<u>Date</u>	Sep 73
<u>Programmer</u>	Weissmuller

FUNCTION

STATUS is the subroutine used by all the "assign file" subroutines. This is the routine which actually tests to see if a requested tape file is already assigned to the run. See DKSTAT for a similar function on mass storage files.

ENTRY POINTS

1. STATUS

CALLING SEQUENCE

1. CALL STATUS (ID,IREEL1,IREEL2,IFLAG)

   a. Inputs

      (1) ID is a two word array which contains the FORTRAN unit left adjusted in FIELDATA in the first word and blank filled through the second word. The association between the FORTRAN unit and the external filename must be established prior to this call.

      (2) IREEL1,IREEL2, and IFLAG are for output only.

   b. Outputs

      (1) IREEL1 will contain the 6 character FIELDATA reel ID of the first reel of the file if IFLAG = Ø, otherwise IREEL1 will be a binary integer zero.

(2) IREEL2 will contain the 6 character FIELDATA reel ID of the second reel of the file if IFLAG = Ø and there are at least two reels in the file, otherwise IREEL2 will be a binary integer zero.

(3) IFLAG is a binary integer flag word. If IFLAG = Ø, the file was already assigned when the call was made, and the reel IDs above will contain the proper information. If, however, IFLAG = 1, the file in question is not currently assigned and the reel IDs are not set.

SPECIAL NOTES

1. Being part of the file assignment package, this subroutine will either be replaced or deleted.

SUBROUTINE IDENTIFICATION

Name	SUBRTN (Assembler Subroutine Support Package)
Language	1100 Assembler
Date	Oct 73
Programmer	Rogers

FUNCTION

SUBRTN is a large set of assembler subroutines used at this installation. It is included because some programs were adopted from outside the CODAP area and required this package. Primarily these routines are used in CFHIO, SISO, and READPF, all of which are themselves 1100 Assembler programs.

ENTRY POINTS

1. See Attachment

CALLING SEQUENCES - See Attachment for usage.

SPECIAL NOTES

1. The entry point ZERŌ has been modified within CODAP to be ZERØ to avoid an entry point conflict.

LIBZ.SUBRTN (UNIVAC 1108 Assembler Language Subroutines)

Entry Points -- Data Handling

BINFD	Binary to Fieldata (no zero suppression)
BINOCT	Binary to Octal
DATETIME	Current date and time of day
FDBIN	Fieldata to Binary
MOVBIN	Binary to Fieldata (left adjusted)
MOVFD	Move Fieldata character string
MOVFD1	Move Fieldata character string (Sets A3 and X8)
OCTBIN	Octal to Binary
ZERO	Binary to Fieldata with zero suppression

Entry Points -- Free Format Scan

ISCAN	Initialize and scan
RESSCN	Restore scan pointers
SAVSCN	Save scan pointers
SCAN	Scan for one field
SETSCN	Set scan parameters

Entry Points -- Character Handling

GETC	Get a character
PUTC	Put a character
PUTCR	Put a character (Reverse)

Entry Points -- Data

BIN	Binary value of "FIELD"
CARD	Input area for SCAN
DATE	Output from DATETIME
FIELD	Output (FD) from SCAN (left justified, space fill)
INFO	Scan information flags
TIME	Output from DATETIME

General Information:

All partial word parameters to these subroutines require the following forms directive:

```
ARG FORM 12,6,18
```

All registers used by the data handling and scan subroutines will be saved and restored in the subroutine. X11 is always the linkage register.

The character handling tables (GETC, PUTC, PUTCR) require initial register settings.

These subroutines are NOT compatible with FORTRAN's calling sequence.

BINFD -- Binary to Fieldata

This routine converts a signed binary integer value in memory to a Fieldata character string. The result will have leading zeros and the sign will be placed in the leftmost character position if the binary value is negative. Positive numbers have a zero in the leftmost position.

If the receiving storage area is too small to accomodate the value and the sign, it will be truncated with no error indication given.

```
LMJ     X11,BINFD
ARG     SC,Ø,AREA
ARG     NC,X,BINVAL
```

SC.......Starting character position of the receiving area
AREA.....Base address of the FD area
NC.......Number of characters in the receiving area
X........Cptional X register if BINVAL is indexed
BINVAL...Location of the binary value

Example:

```
LMJ     X11,BINFD
ARG     62,Ø,LINE1
ARG     9,Ø,NUM          NUM = 34624
```

Characters 62-7Ø of LINE1 will contain ØØØ34624. If NUM contained -34624, characters 62-7Ø would contain -ØØ34624.

BINOCT -- Binary to Octal

This routine converts the contents of one word of storage to its 1-12 digit octal equ -ent in Fieldata.

If the receiving storage area is less than 12 characters the value will be truncated to fit.

```
        LMJ     X11, BINOCT
        ARG     SC,Ø,AREA
        ARG     NC,X,BINVAL
```

SC.......Starting character position of the receiving area
AREA.....Base address of the FD area
NC.......Number of characters in the receiving area
X........Optional X register if BINVAL is indexed
BINVAL...Location of the binary value

Example:

```
        LMJ     X11,BINOCT
        ARG     43,Ø,HDR
        ARG     6,Ø,CTLWRD              CTLWRD = ØØØØØØ1Ø4Ø221
```

Characters 43-48 of HDR will contail Ø4Ø221. If CTLWRD contained -1, characters 43-48 would contain 777776.

DATETIME -- Current date and time of day

This routine picks up the current date and time of day from EXEC 8 and converts them to a form suitable for printing.

```
LMJ    X11,DATETIME
```

The results are as follows with trailing spaces in DATE+2 and TIME+1

DATE	YYMMDD
DATE+1,DATE+2	DD MMM YY
TIME,TIME+1	HH:MM:SS

FDBIN -- Fieldata to Binary

This routine converts a Fieldata character string to a signed 'inary value. The character string is always scanned, leading spaces are ignored and FDBIN stops scanning on the end of the field or a trailing space.

Two modes of operation are possible, (1) nonediting and (2) editing. Only in editing mode will a numeric test be performed on the nonblank characters.

```
LMJ    X11,FDBIN       NONEDITING MODE
ARG    SC,Ø,AREA
ARG    NC,X,BINVAL
```

SC.......Starting character of the input character string
AREA.....Base address of the FD area
NC.......Number of characters in the input string
X........Optional X register if BINVAL is indexed
BINVAL...Resulting signed binary value

```
LMJ    X11,FDBIN       EDITING MODE
ARG    SC,E,AREA
ARG    NC,X,BINVAL
+      ERRXIT
```

E........Any nonzero value
ERRXIT...Alternate return point if the editing results in an error condition. Under this circumstance, BINVAL will contain one of the following values:

Ø = FDVAL was all spaces or only a sign
1 = At least one nonnumeric character was found
2 = The sign does no. precede the value
3 = The number is more than 10 digits in length

Example:

```
LMJ    X11,FDBIN
ARG    1,Ø,FIELD
ARG    12,Ø,BIN             FIELD = "   - 6   "
```

Location BIN will contain 777777777771$_8$. If FIELD contained "   2817", BIN would contain ØØØØØØØØ54Ø1$_8$.

MOVBIN -- Binary to Fieldata (Left Adjusted)

This routine converts a signed binary value to Fieldata and stores all significant digits (plus the minus sign for negative values) left to right beginning with the starting character specified in the calling sequence. The length of the character string is returned to the calling program.

MOVBIN uses subroutine ZERO to make the initial conversion prior to storing the character string.

```
LMJ    X11,MOVBIN
ARG    SC,Ø,AREA
ARG    NC,X,BINVAL
```

SC.......Starting character position of the receiving area
AREA.....Base address of the FD area
NC.......Number of characters stored by MOVBIN
X........Optional X register if BINVAL is indexed
BINVAL...Location of the binarv value

Example:

```
LMJ    X11,MOVBIN
ARG    75,Ø,MSG
ARG    Ø,Ø,INCT           INCT = ØØØØØØØ3ØØ71₈
```

Characters 75-79 of MSG will contail "12345" and the number 5 will be stored in T1 of the second parameter. If INCT had contained 7777777477Ø6$_8$, characters 75-8Ø of MSG would contain -12345 and the number 6 would have been stored in T1 of the second parameter.

MOVFD & MOVFD1 -- Move Fieldata character string

This routine will move a Fieldata character string of up to 4Ø95 characters in length from one area of memorv to another.

MOVFD and MOVFD1 use both the GETC and the PUTC routines to move data.

MOVFD1 is identical to MOVFD with the exception that registers A3 and X8 will not be restored to their original contents. Thev will be set to the word and byte position of the first charac,er following the receiving field.

```
    LMJ     X11,MOVFD (or MOVFD1)
    ARG     NC,FBYTE,FWORD
    ARG     NB,TBYTE,TWORD
```

NC.......Number of characters to be moved
FBYTE....1st bvte position (Ø-5) within FWORD
FWORD....Starting address of the sending (from) field
NB.......Number of blanks (spaces) to move following the character string
TBYTE....1st byte position (Ø-5) within TWORD
TWORD....Starting address of the receiving (to) field

Example:

```
    LMJ     X11,MOVFD
    ARG     32,Ø,MSG
    ARG     Ø,4,LINE+3
```

Characters 1-32 of MSG will be moved to characters 18-49 of LINE. If T1 of the second parameter had been 5, then characters 5Ø-54 would have been set to spaces after the move.

```
    LMJ     X11,MOVFD1
    ARG     17,Ø,M1
    ARG     1,Ø,CTMSG
```

Characters 1-17 of M1 will be moved to characters 1-17 of CTMSG followed bv one space. A3 and X8 will be pointing to character position 19 of CTMSG. In this case, the calling program could con·inue moving data via the PUTC routine.

OCTBIN -- Octal to Binary

This routine will convert a Fieldata (FD) character string to a binarv value. Each digit will be edited for validity (Ø-7). Any out of range character will terminate the scan and the binary value collected to that point will be saved.

```
        LMJ    X11,OCTBIN
        ARG    SC,Ø,FDVAL
        ARG    NC,Ø,BINVAL
```

SC.......Starting character within FDVAL
FDVAL....Fieldata value starting address
NC.......Length of FDVAL
BINVAL...Result of conversion

Example:

```
        LMJ    X11,OCTBIN
        ARG    5,Ø,LINE1
        ARG    6,Ø,NUM
```

If LINE1 contains 'b̸b̸b̸b̸73125b̸b̸b̸', NUM = Ø73125. The blank after the '5' is not in range, so the scan terminates.

ZERO -- Binary to Fieldata with Zero suppression

This routine converts a signed binary integer value in memory to a Fieldata character string. The result will be right adjusted in the receiving field with all leading zeros removed. The units position of the number is always printed. The minus sign on negative numbers will be "floated" to the first position in front of the most significant digit.

If the receiving storage area is too small to accomodate the value and the sign, it will be truncated with no error indication given.

```
LMJ    X11,ZERO
ARG    SC,Ø,AREA
ARG    NC,X,BINVAL
```

SC.......Starting character position of the receiving area
AREA.....Base address of the FD area
NC.......Number of characters in the receiving area
X........Optional X register if BINVAL is indexed
BINVAL...Location of the binary value

Example:

```
LMJ    X11,ZERO
ARG    62,Ø,LINE1
ARG    9,Ø,NUM                NUM = binary 34624
```

Characters 62-7Ø of LINE1 will contain b̸b̸b̸b̸34624. If NUM contained a negative 34624, characters 62-7Ø of LINE1 would contain b̸b̸b̸-34624.

ISCAN -- Initialize and Scan

This routine initializes the pointers for SCAN and performs the first scan on contents of CARD. As each new image is placed into CARD, the scan pointers must be reset to the beginning via ISCAN.

ISCAN will initialize and perform the first scan via a call to SCAN before returning. Therefore, the scan data fields of PTN, CARD, FIELD and INFO apply to both ISCAN and SCAN.

A call to the SETSCN routine may be made prior to a call to ISCAN.

```
    LMJ     X11,ISCAN
    +       LENGTH
```

LENGTH.....Location of the length of the current image

Example:

```
    LA      AØ,(EOF,CARD)
    ER      READ$
    SA,H2   AØ,LEN

    ..
    LMJ     X11,ISCAN
    +       LEN
```

The read symbiont will return the length of the card (in words) via AØ. It is in turn passed to the ISCAN routine.

RESSCN -- Restore scan pointers

This routine restores the word and byte position saved by SAVSCN.

```
        LMJ    X11,RESSCN
```

After calling RESSCN, a call to SETSCN is required.

SAVSCN -- Save scan pointers

This routine saves the current word and byte position of the SCAN subroutine.

```
        LMJ    X11,SAVSCN
```

SCAN -- Scan the input image for one field

This routine performs a free format scan of the image in CARD beginning with the position of the last call to ISCAN or SCAN.

The scan pointers must be initialized by a call to ISCAN, then, all subsequent calls to SCAN will leave the pointers set for the next entry.

```
LMJ    X11,SCAN
```

The following storage areas will contain the results of the scan operation:

```
FIELD....Fieldata value left adjusted with trailing spaces
BIN......Binary value of FIELD if it is numeric
INFO.....Scan information flags

         T1....Length of the FD value in FIELD
         S3....Type of field
               1 = Alphanumeric
               2 = Numeric
               3 = Literal
               4 = Blank (null) field
         S4....EOC (End of Card flag)
               Ø = Not end of card
               1 = End of card
         S5....Subscript of PUNCT in punctuation list
         S6....PUNCT (Punctuation mark which terminated the scan)
```

Example:

Assume that the image in CARD is "CARD (123) ....

```
1st scan (ISCAN):  FIELD = "CARD"
                   BIN = Ø
                   INFO = ØØØ4 Ø1 ØØ Ø4 51

2nd scan (SCAN):   FIELD = "123"
                   BIN = 123
                   INFO = ØØØ3 Ø2 Ø1 Ø5 4Ø
```

SETSCN -- Set Scan Parameters

This routine sets the free format scan parameters STARTC, PCTL and NPCT for ISCAN and SCAN. SETSCN must be called prior to a call to ISCAN.

The two items set by SETSCN are the starting position and the punctuation list. They are the first and second parameters for SETSCN, respectively.

```
    LMJ   X11,SETSCN
    ARG   BYTE,Ø,WORD  or + Ø
    ARG   NPCT,Ø,PLIST or + Ø
```

BYTE.....1st Byte with word (Ø-5)
WORD.....1st Word of the scan area (within CARD)
NPCT.....Number of punctuation marks in PLIST
PLIST....Punctuation mark list address

```
        If 1st parameter = + Ø SETSCN uses Ø,Ø,CARD
        If 2nd parameter = + Ø SETSCN uses 9,Ø,PLIST

        PLIST = ','            NPCT = 1
                '='                   2
                '/'                   3
                '('                   4
                ')'                   5
                '*'                   6
                '-'                   7
                '+'                   8
                '.'                   9
```

Note: The punctuation mark list contains one punctuation mark per word (S6).

Example:

```
       LMJ   X11,SETSCN
       +     Ø
       ARG   2,Ø,NEWPCT
       ..

NEWPCT +     ','
       +     '='
```

The starting character position for scanning the image in CARD will remain at its normal position of character 1 and the punctuation mark list will be changed to comma and equal.

It should be noted here that a space is always a scan terminating character and that literals bounded by quotes are always defined. The character octal 76 may not be used since it is the end of card flag.

GETC, PUTC, and PUTCR -- Character handling tables

These routines retrieve (GETC) or store (PUTC and PUTCR) character strings from and into memory. When moving over a word boundary, X6 is destroyed by the update routine.

```
GETC.....LX       X7,(1,WORD)
         LA       A2,(1,BYTE)

         ..
         EX       GETC,*A2             A4 contains the next character

PUTC.....LX       X8,(1,WORD)
         LA       A3,(1,BYTE)

         ..
         EX       PUTC,*A3             S6 of A4 is stored

PUTCR....LX       X8,(-1,WORD)
         LA       A3,(1,5-BYTE)

         ..
         EX       PUTCR,*A3            S6 of A6 is stored
```

WORD.....Beginning word address of he character string
BYTE.....Beginning byte within WORD (Ø-5)

Example:

```
         LX       X7,(1,MSG)
         LX       X8,(1,LINE+2)
         LA       A2,(1,Ø)
         LA       A3,(1,3)
         LR,U     R1,14
         EX       GETC.*A2
         EX       PUTC,*A3
         JGD      R1,$-2
```

The 15 characters beginning with character 1 of MSG will be moved to the 15 characters beginning with character 16 of LINE.

SUBROUTINE IDENTIFICATION

Name	TABL3 (Table 3 of FORTRAN V Sort Package)
Language	1100 Assembler
Date	---
Programmer	UNIVAC

FUNCTION

TABL3 is a UNIVAC provided subroutine which is included for reference only. It is part of the FSORT - LINK package which a a FORTRAN V interface to the UNIVAC utility Sort Package. This routine provides 16000 words of core and 200,000 words of mass storage for sorting. TABL3 is the largest non-tape space reservation subroutine of the TABL1 - TABL6 series and is the only one used in the CODAP system.

ENTRY POINTS

1. SOPEN3

CALLING SEQUENCE

1. CALL SOPEN3 ($RET,$ESORT,MAXLRL,MAXLEN,ITABL)

   a. Inputs

      (1) RET is the FORTRAN statement number to return to upon completing this call.

      (2) ESORT is the FORTRAN statement number to return to after SSORT of LINK is called and all sorting is complete.

      (3) MAXLRL is a binary integer denoting the largest record length in words to be passed via SRREL to the sort routine.

      (4) MAXLEN is a binary integer denoting the maximum number of words required to hold all sort fields (normally 1 or 2).

(5) ITABL is an array containing the control information for the sort. Its contents are set by a data statement and necessarv changes are made by calling FSORT or FSORT2. Each sort field is defined by six entries in the array and an integer value of 99999 signals an end to the sort fields. All values are binary integer and denote the following attributes of the sort:

Entry 1: Word # of sort key (=1 or 2)

Entry 2: Bit position of sort key (Left to Right, =1)

Entry 3: Number of bits in sort key (=36=full word)

Entry 4: Type of sort: (usually Ø or 1)

Ø = alphanumeric

1 = binary 1108 format

2 = signed decimal

Entry 5: Order of sort

Ø = ascending

1 = descending

Entry 6: Significance number

1 = Major sort field

2 = Minor sort field

b. Outputs

(1) Area in core and mass storage are reserved for the sort.

(2) Return address from SSORT is established.

(3) The entry points in LINK may be called.

SPECIAL NOTES

1. This is a highly specialized subroutine serving to allocate space for the sort package. This routine will undoubtedly be replaced or deleted at any non-UNIVAC installation.

SUBROUTINE IDENTIFICATION

Name TMTCSC (Transmit Array)

Language FORTRAN V

Date May 73

Programmer CSC/Weissmuller

FUNCTION

TMTCSC causes a specified number of words from one array to be copied into a second array.

ENTRY POINTS

1. TMTCSC

CALLING SEQUENCE

1. CALL TMTCSC (NWORDS,ITO,IFROM)

   a. Inputs

      (1) NWORDS is the number of words, starting with the first word in each array, to be transmitted.

      (2) ITO is the receiving array.

      (3) IFROM is the sending array.

   b. Outputs

      (1) The first NWORDS words of ITO are made identical with the first NWORDS words of IFROM.

SPECIAL NOTES

1. This subroutine is an elementary FORTRAN DO loop which replaces an IBM 7040 Assembly Language subroutine widely used in CODAP.

2. By using a subscript other than (implied) 1 in the ITO or IFROM argument of the subroutine call, any block of words may receive, or be sent as, the NWORDS words of data. In particular, different sections of the same

ITO array may be used to receive different types of information. For example, in CODAP this subroutine is used to gather several different arrays and, in a series of calls, place them into a single array.

SUBROUTINE IDENTIFICATION

Name	TREAD (Tape Read of FORTRAN Print)
Language	1100 Assembler
Date	Oct 73
Programmer	Weissmuller

FUNCTION

TREAD will read a FORTRAN written print file from a tape. This subroutine is used primarily for reading, copying, or printing the report file. See DREAD for a similar function on mass storage files.

ENTRY POINTS

1. TREAD
2. REWIND

CALLING SEQUENCES

1. CALL TREAD (NWORDS,LINE,JSEQ,$NEW,$EOF)

   a. Inputs

      (1) NWORDS is for output only.

      (2) LINE is a 22 word array for output only.

      (3) JSEQ is for output only.

      (4) NEW is the FORTRAN statement number to jump to if a new report sentinel is read.

      (5) EOF is the FORTRAN statement number to jump to if an end of file is encountered in an attempt to read the next line.

   b. Outputs

      (1) NWORDS is a binary integer which is the number of words in the print image array LINE.

(2) LINE contains the next print line read. It has a maximum length of 22 words and may be printed thusly:

```
    WRITE(6,100) (LINE(I),I=1,NWORDS)
100 FORMAT(22A6)
```

(3) JSEQ is a binary integer which is set equal to the index number of the current Report being read. This value is set only when the $NEW return is used.

2. CALL REWIND

a. Inputs - None

. Outputs

(1) The print image file read by TREAD is rewound.

SPECIAL NOTES

1. This is a very machine dependent subroutine and should either be replaced or eliminated. The primary function of this subroutine is to read print images from tape and in particular, print images in Report File Format. [Report File Format simply means that individual reports are preceded by a sentinel of the form 'BEGIN REPORT',N where N is a binary integer written out in 1A6 format.]

2. Both TREAD and DREAD assume FORTRAN UNIT 28 is the print image file. This association must be established prior to calling either subroutine.

SUBROUTINE IDENTIFICATION

Name	ZERBLK (Change Zeroes to Blanks)
Language	FORTRAN V
Date	Mar 74
Programmer	Weissmuller

FUNCTION

ZERBLK changes certain leading FIELDATA zeroes on INPSTD Task/Duty Title Cards to blanks.

ENTRY POINTS

1. ZERBLK

CALLING SEQUENCE

1. CALL ZERBLK (ID,IDUT)

   a. Inputs

      (1) ID is an 11-word array containing a Task/Duty Title Card image.

      (2) IDUT is for output only.

   b. Outputs

      (1) IDUT returns the duty character, and

      (2) Any leading zeroes in the task number are changed to blanks, OR

      (3) If the task number field contains invalid characters the program aborts after a diagnostic.

SPECIAL NOTES

1. The subroutine is specially written for use by INPSTD.
2. ZERBLK employs subroutines BCDBIN and ERTRAN, and the FORTRAN ENCODE, DECODE, and "R" format specification.

SUBROUTINE IDENTIFICATION

Name ZERO (Store Floating Point Zeroes into Array)

Language FORTRAN V

Date May 73

Programmer CSC/Weissmuller

FUNCTION

ZERO fills a specified number of words in an array with floating point zeroes.

ENTRY POINTS

1. ZERO

CALLING SEQUENCE

1. CALL ZERO (ARRAY,NWORDS)

   a. Inputs

      (1) ARRAY is an array for floating point values.

      (2) NWORDS is the number of words, starting with the first word in the array to be zero filled.

   b. Outputs

      (1) The first NWORDS words of ARRAY are set equal to zero.

SPECIAL NOTES

1. This subroutine is an elementary FORTRAN DO loop which replaces an IBM 7040 Assembly Language subroutine.

2. By using a subscript other than (implied) 1 in the ARRAY argument of the subroutine call, any block of words may be set to zero.

SUBROUTINE IDENTIFICATION

Name	ZEROI (Store Integer Zeroes into Array)
Language	FORTRAN V
Date	May 73
Programmer	CSC/Weissmuller

FUNCTION

ZEROI fills a specified number of words in an array with integer zeroes.

ENTRY POINTS

1. ZEROI

CALLING SEQUENCE

1. CALL ZEROI (IRRAY,NWORDS)

   a. Inputs

      (1) IRRAY is an array for integer values.

      (2) NWORDS is the number of words, starting with the first word in the array to be zero filled.

   b. Outputs

      (1) The first NWORDS words of IRRAY are set equal to zero.

SPECIAL NOTES

1. This subroutine is an elementary FORTRAN DO loop which replaces an IBM 7040 Assembly Language subroutine.

2. By using a subscript other than (implied) 1 in the IRRAY argument of the subroutine call, any block of words may be set to zero.