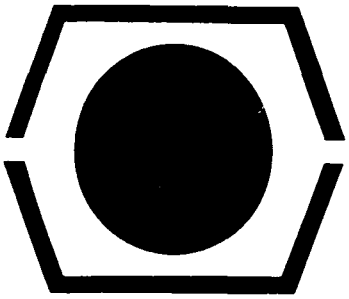

DECUS
PROGRAM LIBRARY

DECUS NO.	8-610
TITLE	INVENT-8
AUTHOR	Charles Moeder
COMPANY	Digital Equipment Corporation Maynard, Massachusetts
DATE	
SOURCE LANGUAGE	SABR

DECUS Program Library Write-up

DECUS NO. 8-610

INVENT-8 A SUMMARY

INVENT-8 IS A SERIES OF GENERAL PURPOSE DATA MANIPULATION AND FILE HANDELING SUBROUTINES. SOME CREATE NEW FILES, AND SOME OPERATE ON ALREADY EXISTING FILES, ADDING AND DELETING INFORMATION. THEY ARE ALL DESIGNED TO OPERATE WITH FORTRAN AND ARE ALL CALLABLE FROM THE FORTRAN LEVEL.

BELOW IS AN OVERVIEW OF THE ROUTINES, FOLLOWED BY A DETAILED DESCRIPTION OF THE CALLS AND THE POSSIBLE ERROR MESSAGES

ISAM. RL

ISAM. RL IS BY FAR THE MOST INTELEGENT AND MOST WIDLY USED OF THE SUBROUTINES INCLUDED HERE. IT OPERATES ONLY ON AN ALREADY EXISTING CORE IMMAGE FILES LOCATED ON 'SYS'. ISAM HAS THE ABILITY OF ADDING AND DELETING INDIVIDUAL RECORDS WITHIN A PRE-DEFINED FILE. IT WILL KEEP TRACK OF MOST OF THE EXTRA SPACE AS IT IS USED AND FREEDED. RECORDS MAY BE ADDED IN THE MIDDLE OF THE FILE, AT THE END, OR AT THE BEGINNING OF THE FILE. SIMIARLY, RECORDS MAY BE DELETED AT THE MIDDLE, THE END, OR THE BEGINNING. ISAM ALSO HAS THE ABILITY OF MODIFYING DATA IN ANY GIVEN RECORD AS WELL AS MOVING FORWARD OR BACKWARD WITHIN THE FILE.

ROISAM. RL

ROISAM WILL OPEN A FILE ON ANY FILE STRUCTURED OS/8 DEVICE. IT ONLY HAS THE CAPABILITY OF READING CORE IMMAGE DATA FROM THAT DEVICE.

WOISAM. RL

WOISAM WILL OPEN AN OUTPUT FILE ON ANY FILE STRUCTURED DEVICE SUPPORTED BY OS/8. IT ONLY HAS THE CAPABILITY OF WRITING CORE IMMAGE DATA ON THE DEVICE.

DUMP. RL

DUMP CAN READ AND WRITE DATA FROM 'SYS' ONLY AND HAS BEEN COMBINED WITH THE FORMATTING ROUTINE ('ZERO') ONLY TO CONSERVE CORE MEMORY. DUMP AND ISAM SHOULD ***NEVER*** BE USED IN THE SAME PROGRAM SINCE THEY BOTH USE THE SAME COMMON MEMORY LOCATIONS AS THE DATA STORAGE AREA.

UNLESS THE PROGRAMMER SPECIFICALLY WANTS THE ROUTINE 'ZERO' INCLUDED, USE ISAM SINCE IT IS FAR, FAR MORE INTELEGENT.

ISAM, ROISAM, WOISAM, AND DUMP ALL ONLY OPERATE ON CORE IMMAGE FILES!!!

MULTI.RL

MULTI IS THE MULTIPLE REEL DECTAPE ASCII FILE DRIVER. IT USES THE DEVICE INDEPENDEDT CHANNEL 4 READ AND WRITE COMMANDS, ONLY AFTER CALLS TO MULTI HAVE BEEN MADE TO PRESET THE DATA PATHS. MULTI IS USED WHENEVER LARGE AMMOUNTS OF DATA ARE BEING MINIPULATED.

SORT

THIS SORT PROGRAM WILL SORT IN ANY SPECIFIED SEQUENCE AN OUTPUT FILE CREATED BY WOISAM. WHEN IT HAS COMPLETED THE SORTING OPERATION, THE RESULT IS ACCEPTABLE TO EITHER ROISAM OR ISAM.

THE SORT PROGRAM IS ACTUALLY A SORT GENERATOR IN THAT IT ASKS A SERIES OF QUESTIONS(WHAT IS THE SORTING SEQUENCE, HOW TO INTERPRETE THE DATA, ETC) AS WELL AS WETHER OR NOT TO CHAIN TO ANOTHER CORE IMMAGE PROGRAM.

BELOW IS A DETAILED LAYOUT OF THE CORE IMMAGE FILE AS WELL AS THE SPECIFIC PARAMETERS ASSOCIATED WITH EACH. AS THE READER WILL NOTE, THERE IS A STRIKING SIMILARITY BETWEEN THE CALLS TO THE VARIOUS SUBROUTINES, AND A LIST FOLLOWS:

OPERATION DESIRED	ROUTINE USED					
	ISAM	ROISAM	WOISAM	DUMP	MULTI INPUT	MULTI OUTPUT
OPEN FILE	INITL	GETOP	PUTOP	FOPN	SPIOP	SPOOP
READ A RECORD	GET	GETX	N/A	GETF	READ	N/A
WRITE A RECORD	PUT	N/A	PUTX	PUTF	N/A	WRITE
BACKSPACE 1 RECORD	BKSPC	N/A	N/A	N/A	N/A	N/A
DELETE 1 RECORD	DELET	N/A	N/A	N/A	N/A	N/A
CLOSE THE FILE	FINSH	N/A	PUTCL	FCLOS	N/A	SPOCL
ALTER A RECORD	RWRTE	N/A	N/A	N/A	N/A	N/A
GO BACK TO THE BEGINNIND DO NOT UPSET THE INPUT STREAM IN ANY WAY	RESET	N/A	N/A	N/A	N/A	N/A
FORMAT A PRE-DEFINED FILE	N/A	N/A	N/A	ZERO	N/A	N/A
PASS TO ANOTHER REEL OF DECTAPE IF NECESSARY	N/A	N/A	N/A	N/A	RDCHK	WRTCHK

THIS IS A DETAILED DESCRIPTION OF THE SOFTWARE SYSTEM INVENT-8

HARDWARE REQUIREMENTS:

THIS SYSTEM WILL RUN ON ANY OS/8 CONFIGURATION. IT USES THE MONITOR FOR A WIDE VARIETY OF SERVICES SUCH AS OPENING, CLOSING FILES, AS WELL AS USING THE VARIOUS DEVICE HANDLERS. THE SYSTEM IS DESIGNED AROUND THE FORTRAN OPERATING SYSTEM, WITH ITS SUPPOUT OF THE TTY, HIGH SPEED PAPER TAPE, CARD READER/LINE PRINTER, AS WELL AS DEVICE INDEPENDENT I/O AT RUN TIME.

THERE ARE A SERIES OF AVAILABLE SUBROUTINES WHICH ARE OUTLINED BELOW. THE GENERAL FILE STRUCTURE IS OUTLINED IN THIS SECTION.

THE FILE IS A CHAINED, CORE IMMAGE FILE. WITHIN EACH BLOCK (A BLOCK IS 256 WORDS LONG, IDENTICAL TO A OS/8 BLOCK), THERE IS A FORDWARD POINTER POINTING TO THE NEXT LOGICAL BLOCK. THIS OFFERS THE ADVANTAGE THAT THE PHYSICAL BLOCKS NEED NOT BE IN THE CORRECT LOGICAL ORDER. RATHER, THE LAST WORD POINTS TO THE RELATIVE POSITION OF THE NEXT PHYSICAL RECORD.

IN LIKE MANNER, THE FIRST WORD IN EACH RECORD POINTS TO THE RELATIVE ADDRESS OF THE BLOCK IN A REVERSE DIRECTION. THIS ALSO MEANS THAT IT IS EASILY POSSIBLE TO GO IN EITHER THE FORDWARD OR REVERSE DIRECTION. THE SUBROUTINES ARE FULLY CAPABLE OF DOING BOTH (ALTHOUGH THE FORDWARD DIRECTION IS CONSIDERABLY FASTER).

THE DATA ITSELF PASSES THROUGH THE AREA RESERVED BY FORTRAN AS COMMON STORAGE. THIS STARTS AT 10200 AND WORKS UP FROM THERE. SINCE THE LARGEST RECORD LENGTH IN INVENT-8 IS 125 (DECIMAL) WORDS, ONLY ONE PAGE OF COMMON NEED BE USED KEEPING CORE REQUIREMENTS DOWN.

IN DECIDING ON WHAT RECORD LEGNTH TO USE, TWO ITEMS SHOULD BE KEPT IN MIND:

1) THE FILE FORMATTER ROUTINE ('ZERO') ASSIGNS AN MULTIPLE NUMBER OF RECORDS PER BLOCK. TO KEEP WASTED SPACE TO A MINIMUN, USE A RECORD LENGTH AS NEAR A MULTIPLE OF 254 AS POSSIBLE, AS ANY EXTRA SPACE IN A BLOCK LESS THAN 254, BUT OVER THE MULTIPLE IS WASTED. FOR EXAMPLE, A RECORD LENGTH OF 20 (DECIMAL) WOULD CREATE 12 RECORDS PER BLOCK. WITH THE FWD AND REV POINTERS USING TWO WORDS, 242 OUT OF 256 WORDS WOULD BE USED (WASTING 14 WORDS PER BLOCK). USING A RECORD LENGTH OF 86 WOULD CREATE 2 BLOCKS PER RECORD, AND WASTE 82 WORDS IN EACH AND EVERY RECORD.

2) IF CARE IS USED IN THE DESIGN OF THE FILE, INFORMATION THAT IS COMMON TO MANY RECORDS NEED NOT BE STORED, THUS SAVING MANY COPIES OF THE SAME INFORMATION

THE REST OF THE FILE ORGINIZATION, THEN, IS TO LEFT JUSTIFY THE RECORDS IN THE BLOCK. AS GETS AND PUTS (READ AND WRITE EQUIVILANTS) ARE MADE, THE ROUTINES WILL ACCESS THE MASTER FILE AND GET (OR PUT) THE DATA FROM 10200 TO 10200 + HOWEVER LONG THE RECORDS ARE (IE 10223 USING A RECORD LENGTH OF 20-DECIMAL).

THE FORTRAN PROGRAM MAY EASILY ACCESS THESE LOCATIONS BY MERELY SPECIFYING THEM AS COMMOM. THE FORTRAN COMPILER WILL START WITH THE FIRST VARIABLE SPECIFIED AS COMMON, AND ASSIGN AN ADDRESS OF 10200 AND USE INCREASING ADDRESS FROM THERE USING ONE WORD FOR INTEGER VARIABLES, AND THREE WORDS FOR REAL (FLOATING POINT) VARIABLES.

WITH THIS SCHEME, THERE IS NO REASON FOR THE USERS PROGRAM TO HAVE TO USE THE SABR FEATURE OF THE FORTRAN COMPILER.

THE FIRST RECORD OF A FILE IS DEFINED SLIGHTLY DIFFERANTLY. THE FIRST WORD CONTAINS THE RELATIVE ADDRESS OF THE START OF THE CHAIN OF BLOCKS CONTAINING ANY DATA, THE SECOND WORD IN THE HEADER RECORD CONTAINS THE RELATIVE ADDRESS OF THE FIRST BLOCK IN THE FREE CHAIN (THOSE BLOCKS THAT CONTAIN NO DATA AT ALL). THE THIRD WORD CONTAINS THE NUMBER OF WORDS IN EACH RECORD, AND THE FOURTH WORD CONTAINS THE NUMBER OF RECORDS IN A BLOCK. AS A FILE IS OPENED, THE FILE DRIVER WILL EXTRACT THIS INFORMATION FROM THE HEADER, AND THAN PASS TO THE FIRST DATA BLOCK.

THE ASSUMPTION THROUGHOUT THIS SYSTEM IS THAT AN INTEGER 999 IN THE FIRST POSITION OF THE LAST RECORD WILL INDICATE AN END OF FILE, AND AS CALLS TO GET (SEE BELOW) ARE MADE, THE FIRST ACTION MADE SHOULD BE TO CHECK FOR A 999. IF IT IS FOUND, AN END OF FILE MAY BE ASSUMED.

THESE ARE ALSO PLACED IN THE FIRST RECORD BY THE FORMATTER ('ZERO'), SO THAT A FILE OF CORRECT FORMAT WOULD BE CREATED.

A TYPICAL FILE LAYOUT (WITH 10 BLOCKS) MIGHT BE AS FOLLOWS:

ABS BLOCK NO	REL BLOCK NO	REV. POINTER	LOGICAL RECORD NO. CONTAINED WITHIN THIS BLOCK	FWD POINTER
350	0	SPECIAL:STARTING BLOCK NO,FIRST FREE BLOCK NO, NO WDS PER RECORD,NO RECS PER BLK		
351	1	0 (1)	1,2,3,4,5	4
352	2	4	12,13	7
353	3	0 (3)		5
354	4	1	6,7,8,9,10,11	2
355	5	3		6
356	6	5		10
357	7	2	14,15,16,17	0 (2)
360	10	6		0 (4)

NOTES:

(1) BEGINNING OF DATA STRING
(2) END OF DATA STRING
(3) BEGINNING OF FREE (EMPTY) STRING
(4) END OF FREE STRING

NOTE THAT THE FILE CHAINS FROM RELATIVE BLOCK NUMBER 1 TO RELATIVE BLOCK NO. 4 TO BLOCK NO. 2 TO BLOCK NO. 7. BLOCK NO 7 IS THE LAST BLOCK IN THE DATA STRING. THE FREE STRING STARTS AT RELATIVE BLOCK NO. 3 AND ENDS AT 10. READING THESE INVENT FILES PAST (IN THIS CASE) RECORD NO 17 WOULD CAUSE AN UNEXPECTED EOF ERROR MESSAGE (SEE ERROR LISTING BELOW).

WITH THIS IN MIND, AN OUTLINE FOR THE SUBROUTINES FOLLOWS.

ISAM.RL

THIS SECTION COVERES THE CALLING SEQUENCE TO ISAM.RL AND THE ACTIONS THE DRIVER WILL TAKE.

THIS IS THE MASTER DRIVER FOR GENERAL INPUT/OUTPUT ROUTINES AND OPERATE AS OUTLINED BELOW.

THE ISAM DRIVER WILL ASSUME A DEVICE NAME OF SYS. THIS IS REQUIRED BECAUSE IT USES THE PERMINATELY RESIDENT DEVICE DRIVER, THUS ALLOWING FREE USE OF DEVICE INDEPENDENT I/O AVAILABLE IN FORTRAN. THE ONLY RESTRICTION IS THAT THE CALL TO INITL MUST *****MUST***** BE MADE PRIOR TO ANY USE OF INPUT CHANNEL 4!!!!!!!!!!!!!!

THE ISAM DRIVER WILL KEEP ALL BLOCKS CURRENT AS DATA IS PROCESSED EXCEPT THE HEADER BLOCK WHICH CONTAINS ALL STARTING AND ENDING INFORMATION. IN ORDER TO PROTECT THE INTEGRETY OF THE FILE ON A CRASH OR UPON TYPING A CONTROL C, A TRAP HAS BEEN PLACED IN LOCATION 07600(THE ENTRY TO THE MONITOR).

THIS CAUSES TWO PROBLEMS

1) IF THE SYSTEM IS MANUALLY RESTARTED, LOCATION 07600 MUST BE RESET TO 4207. IF A ^C IS TYPED, THE ISAM DRIVER WILL DO IT ITSELF

2) IF ODT IS USED, THE TWO LOCATIONS THAT PUT A JMP 0165 IN 07600 MUST BE NOP'ED SINCE ODT USES 07600 FOR SWAPPING. THESE TWO CELLS ARE AT INITL+101, AND INITL+102 (A 1312 AND 3722 RESPECTIVLY)

WITH THIS IN MIND, THE FOLLOWING IS THE CALL SEQUENCE TO THE EIGHT ENTRIES.

INITL('FILNAM')

THIS CALL ASSUMES A SINGLE ARGUMENT 'FILMAN' WITH A .DA ASSUMED EXTENSION. THE FILE NAME SHOULD BE ONE TO SIX CHARACTERS WITH THE FIRST BEING AN ALPHA.
SHOULD THE NAME BE LESS THAN THE SIX CHARACTERS, LEAVE BLANKS WITHIN THE SIX CHARACTER QUOTATION MARK.

IT WILL PRESET ALL THE GOOD AND NICE VARIABLES IN THE DRIVER FOR FUTURE ACCESS. IT USES THE SUBROUTINE IOPEN, DOES A FILE LOOKUP, EXTRACTS THE VARIABLES IT NEEDS, AND THEN DISCARDS THE REST OF THE INFORMATION. AT THIS POINT, ALL DEVICE INDEPENDENT I/O HAS NO RESTRICTIONS SINCE ONCE OPENED, THE ISAM FILE IS COMPLETELY SELF CONTAINED (HENCE THE REQUIREMENT SYS: ONLY).

GET

GET HAS NO ARGS AND WILL GET THE NEXT LOGICAL RECORD IN THE FILE. IT WILL AUTOMATICLY PASS FORWARD IN THE FILE AS REQUIRED. IT WILL PLACE THE INFORMATION IN COMMON STORAGE (I.E. IN 10200 AND UP).

PUT

PUT ALSO HAS NO ARGUMENTS AND WILL PLACE THE DATA FROM COMMON STORAGE(SEE ABOVE) DIRECTLY AFTER THE RECORD THAT WAS READ. IT WILL CREATE NEW BLOCKS AS IT REQUIRES THEM SO THAT THE PHYSICAL SEQUENCE OF THE BLOCKS IS COMPLETELY TRANSPARENT TO THE USER AND HIS PROGRAM. AT THE CONCLUSION OF A CALL TO PUT, THE FILE WILL BE ONE RECORD LONGER. PUT, IF IT NEEDS THE ROOM, WILL STEAL THE FIRST BLOCK FROM THE FREE STRING, AND PLACE IT IN THE MIDDLE OF THE CHAINED DATA STRING, UPDATING THE POINTERS AS IT DOES IT

BKSPC

BACKSPACE HAS NO ARGS AND WILL MOVE THE FILE ONE LOGICAL RECORD BACKWARDS. SHOULD THE BEGINNING OF THE FILE BE ENCOUNTERED, THE ROUTINE WILL GET THE FIRST LOGICAL RECORD.

DELET

DELETE WILL DO JUST THAT - DELETE THE RECORD THAT WAS JUST READ. IT WILL THEN GET THE NEXT LOGICAL RECORD. IT, ALSO, HAS NO ARGUMENTS. IT WILL ALSO CHECK FOR BLOCKS THAT ARE MADE EMPTY PLACING THEM ON THE FREE STRING IF THEY ARE. HENCE, THE FREE STRING IS A VERY DYNAMIC LIST, OPERATING IN A LAST IN FIRST OUT MODE.

FINSH

FINISH WILL CLOSE THE FILE TO ALL ADDITIONAL I/O. IT WILL UPDATE THE HEADER RECORD CONTAINING THE START AND FREE LIST RECORD LOCATION. IT ALSO CLEARS THE TRAP TO THE CRASH ROUTINE, SO THAT ALL FUTURE CALLS TO THE MONITOR WILL BE HANDLED PROPERLY.

RWRTE

REWRITE WILL REPLACE THE RECORD THAT WAS JUST READ WITH THE INFORMATION OBTAINED IN THE COMMON STORAGE AREA. NO RECORDS ARE ADDED OR DELETED. REWRITE IS EQUIVILENT TO A CALL TO DELET, FOLLOWED BY A CALL TO BKSPC, AND A CALL TO PUT EXCEPT THAT IT IS MUCH FASTER SINCE ALL BOOKEEPING NEED NOT BE TESTED. AFTER A CALL TO RWRTE, THE FILE WILL HAVE THE SAME LEGNTH AS BEFORE THE CALL.

RESET

RESET WILL TAKE THE FILE BACK TO THE BEGINNING, RESET THE VARIOUS PARAMETERS, AND THEN GET THE FIRST LOGICAL RECORD. IT IS EQUIVILENT TO A CALL TO FINSH, FOLLOWED TO A CALL TO INITL EXCEPT THAT IT IS MUCH FASTER, AND THAT IT DOES NOT UPSET DEVICE INDEPENDENT INPUT (CHANNEL 4).

ISAM.RL ERROR MESSAGES

ISM1 ERROR AT LOC XXXXX

THE ADDRESS IS CALL TO GET.

AN UNEXPECTED END OF FILE WAS ENCOUNTERED. THE ERROR IS FATAL, BUT THE FILE IS IN TACT

ISM2 ERROR AT LOC XXXXX

THE ADDRESS IS A CALL TO INITL.

AN ATTEMPT TO OPEN TWO FILES HAS BEEN MADE, OR THERE HAS BEEN NO CALL TO INITL PRIOR TO A CALL TO GET, PUT, ETC. THE ERROR IS FATAL, BUT THE FILE IS IN TACT.

ISM3 ERROR AT LOC XXXXX
THE ADDRESS IS OF A CALL TO PUT.
AN ATTEMPT TO PUT DATA ON THE FILE HAS BEEN MADE WHEN THERE IS NO ROOM LEFT. THE CALL IS CANCELED. THE ERROR IS FATAL, BUT THE FILE IS IN TACT.

ISM4 ERROR AT LOC XXXXX
THE ADDRESS IS FROM THE INTERNAL CALL.
THIS COVERS A WIDE VARIETY IF ERRORS, INCLUDING:
1. BAD LINKAGE (FORDWARD AND REVERSE POINTERS WITHIN TWO BLOCKS DON'T MATCH).
2. AN ATTEMPT TO READ OR WRITE A BLOCK OUTSIDE THE FILE BOUNDRIES WAS MADE.
3. THERE WAS A MASS STORAGE I/O ERROR (FROM SYS:).
4. THERE WAS A NON ZERO REVERSE POINTER ON THE FIRST BLOCK OF THE FREE CHAIN.
5. THERE WAS A BAD LINKAGE IN THE FREE CHAIN.
THE ERROR IS FATAL, BUT THE HEADER RECORD HAS NOT********** BEEN REWRITTEN.

THE ACTUAL ADDRESS IS AS FOLLOWS:

1) XXXXX=YYY10 - POINTERS DONT MATCH. A CALL TO GET
2) XXXXX=YYY37 - POINTERS DONT MATCH. A CALL TO BKSPC OR RWRTE.
3) XXXXX=YYY30 - THE BLOCK TO BE READ OR WRITTED WAS BELOW THE AREA OF THE FILE OPENED.
4) XXXXX=YYY35 - THE BLOCK TO BE READ OR WRITTEN WAS ABOVE THE AREA OF THE FILE OPENED.
5) XXXXX=YYY47 - THE OS/8 DEVICE HANDLER RETURNED AN ERROR.
6) XXXXX=YYZ22 - THE FILE HAD A NON-ZERO POINTER IN THE FIRST BLOCK OF THE FREE STRING.
7) XXXXX=YYW22 - THE FREE LIST HAD BAD POINTERS.
8) XXXXX=YYY41 - THERE WERE BAD LINKAGES IN THE FILE ON A CALL TO PUT.
9) XXXXX=YYY57 - BAD LINKAGES ON A CALL TO DELET.
10) XXXXX=YYY14 - BAD LINKAGES ON A CALL TO DELET.
11) XXXXX=YYY32 - DITTO
12) XXXXX=YYY07 - BAD LINKAGES ON A CALL TO RWRTE.

WHERE YYY IN MEANINGLESS DUE TO THE RELOCATIBILITY OF THE LIBRARY (THE USER HAS NO CONTROL OVER WHERE THE ACTUAL LOADING WILL OCCUR), AND WHERE Z IS AN ODD NUMBER AND W IS AN EVEN NUMBER).

ALSO-----
SINCE ISAM USES IOPEN, THERE IS THE POSIBILITY OF "IOER ERROR AT LOC XXXXX". THIS USUALLY MEANS THAT THE FILE WAS NOT FOUND OR NO /I SWITCH WAS SPECIFIED AT LOADER TIME.

DUMP.RL

THIS SECTION COVERS THE CALLING SEQUENCE TO DUMP.RL, AND THE ACTIONS THE ROUTINE WILL TAKE

THIS IS THE DUMPING AND RELOADING ROUTINE AND HANDLES THE GETS AND PUTS IN A DIFFERENT MANER THAN ISAM.RL

FOPN(FILNAM)

THIS ROUTINE WILL OPEN A FILE ON THE SYSTEMS DEVICE IN THE SAME MANNER AS THE INITL(FILMAN) OUTLINED ABOVE. IT, ALSO, HAS THE SAME RESTRICTIONS, NAMELY THAT IT MUST BE CALLED BEFORE ANY OTHER USE OF DEVICE INDEPENDENT INPUT (CHANNEL 4).

GETF

THIS ROUTINE ACTS IDENTICALLY TO GET OUTLINED IN ISAM. IT IS INCLUDED IN THIS SERIES OF SUBROUTINES FOR THE SAKE OF CONSERVING CORE.

PUTF

THIS ROUTINE WILL PLACE DATA DIRECTLY AFTER THE DATA JUST READ. IT DIFFERS FROM PUT IN THAT IT WILL HOLD ANY SWAPPED OUT DATA IN CORE, RATHER THAN REWRITING THE DATA BACK ON THE SYSTEMS DEVICE. THIS OFFERS THE DISTINCT ADVANTAGE IN THAT THE DATA WILL END UP NOT ONLY IN CORRECT LOGICAL SEQUENCE, BUT ALSO IN CORRECT PHYSICAL SEQUENCE.

SINCE IT DOES HOLD THE DATA IN CORE RATHER THAN ON THE MASS STORAGE DEVICE, PUTF CALLS MUST ***MUST*** BE MADE ONE AFTER ANOTHER. A CALL TO GETF AFTER A CALL TO PUTF WILL YIELD STRANGE RESULTS INDEED!!!. IF IT IS NECESSARY TO CALL GETF AFTER PUTF, CALL FCLOS, THEN FOPN FIRST.

FCLOS

FCLOSE WILL CLOSE THE FILE TO ALL ADDITIONAL I/O. IT WILL CHECK TO SEE IF ANY INFORMAION HAS BEEN HELD IN CORE, WRITING IT OUT ON THE FILE IF THERE IS A POSITINE ANSWER. IT WILL THEN UPDATE THE HEADER RECORD AND RETURN TO THE CALLING PROGRAM.

ZERO('FILNAM',FILENGTH)

A CALL TO ZERO WILL COMPLETLY ZERO AND FORMAT THE FILE SPECIFIED TO THE FILE LENGTH SPECIFIED. CARE MUST BE USED WITH THIS SUBROUTINE SINCE THERE ARE NO QUESTIONS ASKED AS TO WETHER OR NOT THAT IS WHAT YOU REALLY WANTED TO DO. NO DATA IS TRANSMITTED OR RECIEVED.

THIS ROUTINE WILL ALSO PLACE INTEGER 999'S (OR OCTAL 1747'S) IN THE LAST HALF OF THE FIRST BLOCK. THIS IS REQUIRED SO THAT THE USERS PROGRAM MAY DETECT AN END OF FILE BY MERELY TESTING AN INTEGER VARIABLE FOR DECIMAL 999 (ASSUMING THAT FIELD IS NEVER 999 EXCEPT FOR AN EOF).

DUMP.RL ERROR MESSAGES

ZR01 ERROR AT LOC XXXXX

THE ADDRESS SPECIFIED IS THE CALL TO ZERO.

THE PROGRAM ATTEMPTED TO FORMAT A FILE OF LENGTH 0, A NEGATIVE LENGTH, OR A LENGTH OF GREATER THAN 125 (DECIMAL). THE ERROR IS FATAL, BUT THE FILE HAS NOT BEEN ALTERED.

ZR05 ERROR AT LOC XXXXX

THE ADDRESS SPECIFIED IS MEANINGLESS.

AN ATTEMPT TO PUTF DATA WITH A FILE THAT IS FULL WAS ATTEMPTED. THE FILE IS INTACT, BUT THE CALL IS CANCELED. THE ERROR IS FATAL.

ZR06 ERROR AT LOC XXXXX

THE ADDRESS SPECIFIED IS THE CALL TO PUTF

AN ATTEMPT TO ENTER DATA IN THE MIDDLE OF THE FILE FILE, OR AN ATTEMPT TO TO SWAP DATA TWICE INTO THE BUFFER AREA HAS BEEN MADE.

THE ERROR IS FATAL, BUT THE FILE IS IN TACT.

ZR07 ERROR AT LOC XXXXX

THIS CAN BE ANY ONE OF VARIOUS ERRORS SUCH AS

1. BAD LINKAGES (POINTERS)
2. AN ATTEMPT TO READ OR WRITE A BLOCK THAT IS OUTSIDE THE FILE BOUNDRIES.
3. AN I/O ERROR FROM THE SYS: DEVICE HANDLER
4. A NON ZERO REVERSE POINTER ON THE FIRST BLOCK ON THE FREE CHAIN.

THE ADDRESS SPECIFIED IS THAT OF THE INTERNAL CALL.

NOTE:****** THE HEADER RECORD HAS NOT BEEN RESTORED!!!!!!!!!!

ZR08 ERROR AT LOC XXXXX

THE ADDRESS IS THAT OF THE CALL TO GETF.

AN UNEXPECTED END OF FILE HAS BEEN ENCOUNTERED. THE ERROR IS FATAL, BUT THE FILE IS IN TACT.

ZR09 ERROR AT LOC XXXXX

THE ADDRESS IS THAT OF FOPN.

AN ATTEMPT TO GETF OR PUTF HAS BEEN MADE WITH NO PRIOR CALL TO FOPN. THE ERROR IS FATAL, BUT THE FILE IS IN TACT.

THE ACTUAL ADDRESS OF ZR07 IS AS FOLLOWS:

1) XXXXX=YYY47 - BAD LINKAGES FROM A CALL TO GETF
2) XXXXX=YYY73 - BLOCK NUMBER IS TOO SMALL(AS IN ISM4#3 ABOVE).
3) XXXXX=YYY00 - BLOCK NUMBER IS TOO LARGE (AS IN ISM4#4 ABOVE
4) XXXXX=YYY12 - THE OS/8 DEVICE HANDLER RETURNED AN ERROR.
5) XXXXX=YYY07 - A CALL TO PUTF WITH NO ROOM LEFT WAS MADE (THE ERROR ZR05 FAILED TO CATCH THE CONDITION EARLIER).
6) XXXXX=YYY21 - A NON ZERO REVERSE POINTER WAS FOUND IN THE FIRST BLOCK IN THE FREE STRING.
7) XXXXX=YYY37 - A CALL TO PUTF WAS MADE WHEN THE USER WAS NOT AT THE END OF THE VALID DATA.
8) XXXXX=YYY51 - THERE WERE BAD LINKAGES IN THE FREE STRING.

WHERE YYY IS MEANINGLESS.

NOTE: DUMP.RL ALSO USES IOPEN AND HENCE MAY PRODUCE "IOER ERROR AT LOC XXXXX". SEE ISAM FOR THE PROBABLE REASONS.

MULTI.RL

THIS SECTION COVERS THE CALLING SEQUENCE TO MULTI.RL, AND THE ACTIONS THE ROUTINE WILL TAKE

THIS IS THE MULTI-REEL DECTAPE DRIVER, AND IS USED WHEN LARGE DATA FILES ARE REQUIRED.

SPOOP(FILNAM)

THIS SUBROUTINE WILL OPEN A FILE(ASCII) ON DECTAPE #1 WITH THE NAME OBTAINED FROM THE CALLING PROGRAM. IT WILL REJECT THE REEL IF THERE ARE LESS THAN 650 (DECIMAL) BLOCKS AVALIBLE. IT WILL TYPE OUT "REEL NOT EMPTY-PLEASE RELOAD". IT WILL THEN TRY TO OPEN IT AGAIN WITH THE SAME NAME AND CHECK AGAIN. THIS FEATURE MAY BE OVERIDEN BY NOT TYPING A CR, BUT RATHER BY TYPING A "A" AND THEN TYPING A CR (THE A FOR ACCEPT). THE ROUTINE WILL THEN ACCEPT THE REEL EVEN THOUGH IT IS NOT EMPTY.

WRTCHK

WRITECHECK IS CALLED AFTER EACH AND EVERY WRITE (4,X) STATEMENT. THE ROUTINE WILL CHECK THE AVAILABLE SPACE ON THE REEL CURRENTLY IN USE. IF THERE ARE LESS THAN 2 BLOCKS REMAINING, IT WILL CLOSE THE CURRENT REEL,OPEN ANOTHER FILE ON DECTAPE 2 WITH A NAME THE SAME AS THE FIRST EXCEPT THAT AN ARITHMETIC 1 WILL BE ADDED TO THE SIXTH CHARACTER.
FOR EXAMPLE, IF THE FIRST NANE WAS "AAAAA1", THE SECOND NAME WILL BE "AAAAA2". HENCE IT WORKS ONLY IF THE NAME IS FIVE ALPHA CHARACTERS AND ONE DIGIT FOR A TOTAL LENGTH OF SIX CHARACTERS.

THE OUTPUT FILE WILL THEN ALTERNATE BETWEEN DECTAPE 1 AND DECPAPE 2 WITH INCREASING NAMES (IE "AAAAA3", "AAAAA4" ETC.)

WHEN CLOSING AN INTERMEDIATE REEL OF TAPE, THE ROUTINE WILL ALSO ADD TWO RECORDS. THE FIRST RECORD IS "999" (THREE CHARACTERS - THREE 9'S). THE SECOND RECORD WILL CONTAIN A "MORE", INDICATING TO THE INPUT ROUTINES (AND ANY HUMAN READER) THAT THERE IS MORE TO THE FILE.

SPOCL

THIS ROUTINE WILL CLOSE THE OUTPUT FILE TO ADDITIONAL DATA. IT WILL ALSO RESET THE FIRST OUTPUT DEVICE TO DTA1 (AS REQUIRED IN SPOOP ABOVE), AS WELL AS ADDING TWO RECORDS TO THE END OF THE CURRENT FILE (A "999" AS ABOVE, BUT ALSO A "END ") TO INDICATE TO THE INPUT ROUTINES THAT "THATS ALL FOLKS" (AN EOF)

SPIOP(FILNAM)

SPECIAL INPUT OPEN WILL OPEN A FILE ON DECTAPE 5 WITH A NAME PROVIDED BY THE CALLING PROGRAM. THE USER MAY THEN USE THE READ (4,X) STATEMENTS AS REQUIRED.

RDCHK(SWTCH)

READ CHECK WILL EXAMINE THE NEXT RECORD IN THE INPUT STREAM FOR A "MO" OR NOT. THEREFORE IT IS UP TO THE USER IN HIS READ STATEMENTS TO CHECK FOR A "999" IN THE FIRST THREE COLUMNS OF DATA READ. IF HE FINDS THEM, HE SHOULD CALL RDCHK. RDCHK WILL

THEN LOOK AT THE NEXT RECORD (AS MENTIONED ABOVE) FOR A "MO". IF IT FINDS THEM, IT WILL AUTOMATICLY OPEN A NEW FILE ON DECTAPE 6 WITH THE SAME INCREMENTING SEQUENCE AS WRTCH ABOVE (IE "AAAAA1", "AAAAA2", "AAAAA3", ETC.) ALTERNATING BETWEEN DTA5 AND DTA6.

AT THE SAME TIME IT WILL RETURN AN ARGUMENT SWITCH TO INDICATE WETHER OR NOT A TRUE END OF FILE WAS ENCOUNTERED (SW=0 MEANS "MO" WAS NOT FOUND, AND SW=1 MEANS "MO" WAS FOUND). IT IS UP TO THE USER TO ACT ACCORDINGLY. THEREFORE, THE INPUT STREAM WILL ALTERNATE BETWEEN DTA5 AND DTA6, EVEN THOUGH THE ACTUAL DEVICE SWAPPING IS INVISABLE TO THE PROGRAM.

THERE ARE NO ERROR MESSAGES ASSOCIATED WITH MULTI.RL, ALTHOUGH "IOER AT LOC XXXXX" MAY BE PRODUCED BY MULTI'S CALLS TO IOPEN, OOPEN, AND OCLOS.

ROISAM.RL

ROISAM.RL HAS TWO ENTRIES AND DIFFERS FROM THE PREVIOUS SUBROUTINES IN THAT IT WILL OPERATE FROM ANY MASS STORAGE DEVICE SUPPORTED BY THE OS/8 SYSTEM (THIS IS ALSO THE CASE WITH WOISAM. THERE IS, HOWEVER, ONE RESTRICTION ON THIS FILE DRIVER: IF DEVICE INDIPENDENT INPUT IS USED, THE DEVICE SPECIFIED MUST ***MUST*** HAVE A CORESIDENT HANDLER.

THEREFORE THIS IS AIMED AT DECTAPE AND MULTIPLE RK8 DRIVES. FOR EXAMPLE, ON A DISK BASED SYSTEM, ALL DECTAPE DRIVES HAVE CORESIDENT HANDLERS (EXCEPT THE TD8E ACCUMULATOR DECTAPE). ON AN RK8 SYSTEM, DRIVES 2,3, AND 4 HAVE CORESIDENT DRIVERS. ON A TC08 BASED SYSTEM, DECTAPE DRIVES 1-7 HAVE CORESIDENT HANDLERS.

WITH THIS IN MIND, THE CALLING SEQUENCE TO ROISAM, AND WOISAM FOLLOW

GETOP('DEV','FILNAM')

THIS WILL OPEN AN READ ONLY ISAM FILE ON THE DEVICE SPECIFIED. IT USES 'IOPEN', EXTRACTS THE INFORMATION IT NEEDS, AND THEN THE ONLY RESTRICTION OF INPUT DEVICE CODE 4 IS THAT ANY OTHER OPENED INPUT FILES MUST ***MUST*** USE THE SAME OS/8 DEVICE HANDLER.

GETX

GETX WILL GET. THE NEXT LOGICAL RECORD FROM THE INPUT STREAM. IT OPERATES IN THE EXACT SAME WAY AS GET EXCEPT THAT IT WILL PLACE THE DATA IN 10400 ON UP (COMMON, BUT OUT OF THE WAY OF THE AREA USED BY GET AND PUT OF ISAM.RL.

IF ISAM AND ROISAM (AND OR WOISAM) ARE USED IN THE SAME PROGRAM SPACE MAY BE RESERVED BY THE FOLLOWING CODING

```
COMMON I1, I2, I3, IDUM, J1, J2, J3
DIMENSION IDUM(125)
```

ROISAM.RL ERROR MESSAGES

GETX ERROR AT LOC XXXXX

THIS IS THE ONLY ERROR MESSAGE GIVEN BY ROISAM. THE MEANING DEPENDS ON XXXXX

1) XXXXX=YYY36 MEANS THAT AN UNEXPECTED EOF WAS ENCOUNTERED
2) XXXXX=YYY66 MEANS THAT THE FILE WAS NEVER OPENED
3) XXXXX=YYY74 MEANS THAT THERE WAS AN ERROR IN THE OS/8 DEVICE HANDLER RETURN
4) XXXXX=YYY44 MEANS THAT THE POINTERS DIDN'T MATCH

WHERE XXXXX IS THE ADDRESS PRODUCED BY RUN TIME FORTRAN, AND YYY IS MEANINGLESS DUE TO THE RELOCATIBILITY OF THE LOADER (I.E THE USER HAS NO CONTROL OVER WHERE THE SPECIFIC ROUTINE WILL BE LOADED).

WOISAM.RL

THE SAME RESTRICTIONS THAT APPLY FOR READ ONLY ISAM ALSO APPLY FOR WRITE ONLY ISAM (WOISAM.RL). ONCE OPENED, THERE IS NO PROBLEM WITH OUTPUT CHANNEL 4 EXCEPT THAT THE THE OS/8 DRIVERS MUST BE CORESIDENT.

****

NOTE: OS/8 WILL ACCEPT ONE AND ONLY ONE ACTIVE TENTATIVE FILE (OPENED BUT NOT YET CLOSED) PER DEVICE AND INVENT MUST ADHERE TO THIS SYSTEM PARAMETER.

****

WITH THIS IN MIND, THE CALLING SEQUENCE TO WOISAM FOLLOWS:

PUTOP('DEV','FILNAM',FILENGTH)

PUTOP WILL OPEN A TENTATIVE OUTPUT FILE ON THE DEVICE SPECIFIED, WITH THE NAME SPECIFIED, AND WITH THE LENGTH SPECIFIED (REMEMBER THAT THE LENGTH MUST BE GREATER THAN 0 AND LESS THAN 126 DECIMAL). NOTE ALSO THAT THE LENGTH IS NOT IN QUOTES.

PUTX

PUTX WILL TAKE THE DATA FROM 10400 ON UP AND WRITE IT ON THE OUTPUT DEVICE.

PUTCL

PUTCL WILL CLOSE THE FILE AFTER WRITING A SERIES OF EOF'S (999'S).

WOISAM.RL ERROR MESSAGES

PUTX ERROR AT LOC XXXXX.

THIS IS THE ONLY ERROR MESSAGE GIVEN BY WOISAM. THE MEANING DEPENDS ON XXXXX.

1) XXXXX=YYY74 MEANS THAT THE FILE WOULD NOT CLOSE PROPERLY. A WRONG DEVICE (OOPEN CALLED ON A NON CORESIDENT DEVICE) OR OTHER PROBLEM.
2) XXXXX=YYY51 MEANS THAT THE /O SWITCH AT LOADER TIME WAS NOT SPECIFIED.
3) XXXXX=YYY16 MEANS THAT THE FILE RECORD LEGNTH WAS TOO SMALL OR 0.
4) XXXXX=YYY21 MEANS THAT THE FILE RECORD LENGTH WAS TOO LARGE.
5) XXXXX=YYY65 MEANS THAT THERE WAS NOT ENOUGH ROOM ON THE MASS STORAGE DEVICE SPECIFIED IN PUTOP.
6) XXXXX=YYY44 MEANS THAT THERE WAS A HIGH SPEED I/O ERROR (OS/8 DEVICE HANDLER RETURNED AN ERROR).
7) XXXXX=YYY36 MEANS THAT THE FILE WAS NEVER OPENED.

ROISAM AND WOISAM BOTH USE THE FORTRAN LIBRARY SUBROUTINE IOPEN AND HENCE MAY PRODUCE "IOER ERROR AT LOC XXXXX". SEE THE OS/8 USERS MANUAL FOR THE PROPER EXPLINATION.

GTDATE.RL

TWO OTHER ROUTINES HAVE BEEN ADDED TO THE LIBRARY FOR THE SAKE OF SIMPLICITY:GTDATE AND NCOMP.

THE CALLING SEQUENCE IS AS FOLLOWS:

GTDATE:

GDATE(IM,ID,IY)

THIS ROUTINE WILL GET THE DATE FROM MONITOR AND RETURN IT IN THE FORM OF THREE INTEGER VARIABLES, IM, ID, AND IY FOR MONTH (0<IM<13), DAY (0<ID<32), AND YEAR (69<IY<79). THIS IS THE SAME FORMAT THAT THE DATE IS KEPT IN THE MONITOR EXCEPT THAT THE YEAR IN MONITOR IS BETWEEN 0 AND 7.

IF THE DATE HAS NOT BEEN SET SINCE BOOTSTRAPPING THE SYSTEM, THE ROUTINE WILL TYPE "ENTER DATE AND RESTART", AND EXIT TO THE MONITOR RATHER THAN RETURN TO THE CALLING PROGRAM.

CKTST(INTGR)

OFTEN IT IS USEFUL TO TEST FOR AN ALPHA Y (FOR YES) N (FOR NO) AND O (FOR OMIT). THIS CALL WILL DO JUST THAT, AND RETURN INTGR =0 FOR A YES; +1 FOR A NO; AND =-1 FOR AN OMIT. NOTE THAT THE TEST CHARACTER SHOULD HAVE BEEN READ UNDER AN A1 FORMAT. IF THE ROUTINE DOES NOT FIND ONE OF THE THREE, IT WILL TYPE "Y,N, OR O ONLY" AND WAIT FOR ANOTHER ALPHA. IT WILL THEN CHECK AGAIN.

GTDATE.RL ERROR MESSAGES

THERE ARE NO ERROR MESSAGES ASSOCIATED WITH GTDATE.RL

NCOMP:

NCOMP(A1,A2,A3,A4,A5)

THIS ROUTINE WILL COMPARE TWO ALPHANUMERIC FIELDS, AND RETURN A -1 FOR A1<A4; 0 FOR A1=A4; AND +1 FOR A1>A4. IT, THEREFORE IS A FUNCTION AND IS USED:

```
        IF (NCOMP(X1,1,5,Y1,4) 10,20,30
10      ACTION TO BE TAKEN IF X1 COLATES ABOVE Y1
20      ACTION TO BE TAKEN IF X1 AND Y1 COLATE EQUAL
30      ACTION TO BE TAKEN IF X1 COLATES BELOW Y1
```

AND WHERE THE 1 MEANS THAT THE FIRST ELEMENT OF THE SUBSCRIPTED REAL VARIABLE OF X1 IS TO BE USED FIRST, AND THE SECOND ELEMENT OF THE SUBSCRIPTED ARAY Y1 IS TO BE USED. (REMEMBER THAT REAL VARIABLES ARE THREE WORDS EACH). THE 5 MEANS THAT FIVE 12 BIT WORDS (OR 10 PACKED ASCII CHARACTERS) COMPARISONS ARE TO BE MADE.

FOR EXAMPLE:

```
        DIMENSION X1(10),Y1(30)
        IF (NCOMP(X1,1,5,Y1,4)) 10,20,30
```

WOULD COMPARE:

X1(1) TO Y1(2)

X1(2) TO Y1(3) (THE FIRST TWO 12 BIT WORDS ONLY)

AND TAKE A BRANCH TO 10 IF THEY COLATE LOW, TO 20 IF THEY COLATE EQUAL, AND TO 30 IF THEY COLATE HIGH AND SO FORTH. NOTE THAT INTEGER VARIABLES MAY BE USED IF THEY ARE CONSIDERED TO BE ONE 12 BIT WORD EACH.

MOVE(X1,1,9,Y1,15)
        THE ARGUMENTS TO MOVE ARE IDENTICAL TO NCOMP.
THIS IS A SUBROUTINE, HOWEVER. I.E. :
        CALL MOVE(X1,1,9,Y1,15)
        WILL MOVE X1(1) TO Y1(5), X1(2) TO Y1(6), AND X1(3) TO
Y1(7).
        ****** BE SURE THERE IS A Y1(7) IN YOUR DIMENSION
STATEMENT OR IT WILL MOVE DATA TO WHATEVER HAPPENS TO BE
IN THE SUCEEDING MEMORY LOCATIONS.
        CALL MOVE(X1,1,6,Y1,3) IS IDENTICAL TO:
        DO 10 I=1,3
        J=I+4
10   Y1(J)=X1(I)

        EXCEPT THAT IT IS MUCH FASTER.

NCOMP.RL ERROR MESSAGES
        THERE ARE NO ERROR MESSAGES ASSOCIATED WITH NCOMP.RL

LIBRARY NOTES - - - - - - -

THE CURRENT STRUCTURE OF LIBSET AND THE /L OPTION TO THE LINKING LOADER MAKE IT DIFFICULT TO HAVE VERY LARGE LIBRARY DIRECTORIES. FOR THIS REASON, THESE ROUTINES HAVE BEEN ADDED AND TWO LIBRARIES CREATED: LIB81.RL AND LIB82.RL. THE ONLY DIFFERANCE BETWEEN THESE AND LIB8.RL IS THAT IT TOOK TWO DIRECTORIES TO HOLD ALL THE ENTRY POINTS. THE LONGET ROUTINES ARE IN THE FIRST (LIB81.RL) AND THE OVERFLOW IN THE SECOND (LIB82.RL).

IN COMPLING A FORTRAN PROGRAM USE THE FOLLOWING SEQUENCE:

```
.R FORT
DTA3:PROG/I/O/L       (USE THE /I AND /O OPTIONS AS REQUIRED
*LIB81/L              (THE /L OPTION HERE MEANS THE INPUT IS A LIBRARY
*LIB82/L            (/L AGAIN FOR MORE LIBRARY INPUT
*/M$                (THE $ IS THE ECHO OF ALT MODE AND THE /M IS
                    (TO PRODUCE A MAP BEFORE CREATING A CORE IMMAGE
                    (WITH A CORE CONTROL BLOCK.

THE LOADER WILL THEN OUTPUT:
MAIN   12345
ABCDEF 23456
BBBBBB 11221
     .
     .
     .
     .
     .
XYZABD 23456
0000            (THIS IS THE NUMBER OF FREE PAGES IN FIELD 0
0003            (AND THE NUMBER OF FREE PAGES IN FIELD 1, ETC.
```

FOR THOSE INTERESTED, THE SUBROUTINES IN LIB81 ARE:

ISAM.RL
DUMP.RL
IOH.RL
FLOAT.RL
WOISAM.RL
ROISAM.RL
POWERS.RL
UTILTY.RL
INTEGR.RL
TRIG.RL
ATAN.RL

AND THE SUBROUTINES IN LIB82 ARE:

MULTI.RL
IPOWRS.RL
RWTAPE.RL
IOPEN.RL
GTDATE.RL
SQRT.RL
NCOMP.RL

SORT.SV

THE SORT PROGRAM IS IN ITSELF NOT A SORT BUT A SORT GENERATOR. RUNNING THE SORT PROGRAN WILL ACTUALLY PRODUCE A SERIES OF QUESTIONS. THE FIRST QUESTION IS WHAT TYPE OF FORMAT THE DATA IS TO BE INTERPRETED AS - NAMELY I FORMAT (A 12 BIT WORD IN FIXED POINT FORMAT), F FORMAT (A 36 BIT WORD IN FLOATING POINT FORMAT), A2 FORMAT (A 12 BIT WORD IN STRIPPED ASCII (2 6-BIT CHARACTERS) FORMAT), OR A6 FORMAT (A 36 BIT WORD IN STRIPPED ASCII (6 6-BIT CHARACTERS) FORMAT).

THE SECOND QUESTION IS WHERE IN THE RECORD THE HIGH ORDER WORD(S) STARTS. ANY NUMBER BERWEEN 1 AND 125 (DECIMAL) IS OK. THE PROGRAM THEN ASKS IF THE LAST TWO ENTRIES (FORMAT AND LOCATION) ARE CORRECT. ANSWER Y OR N. "N" WILL ALLOW RE-ENTRY OF THE DATA, WHILE "Y" WILL CAUSE THE PROGRAM TO PROCEED TO THE NEXT QUESTION.

THEN THE PROGRAM WILL ASK IF THERE ARE ANY MORE SORT KEYS. ANSWER Y TO INPUT THE SECONDARY SORTING KEY (THE KEY THAT WILL BE TESTED IF THE FIRST KEYS WERE EQUAL). THIS PROCEDURE WILL CONTINUE UNTIL 1) A "N" IS THE ANSWER TO THE "MORE VARS?" QUESTION, OR
2) SIXTEEN SEPERATE SORTING FIELDS ARE ENTERED.
THESE QUESTIONS DEFINE THE DATA LAYOUT TO THE SORTING SECTION OF THE GENERATOR.

UPON THE COMPLETION OF THESE QUESTIONS, THE GENERATOR WILL ASK WHAT DEVICE THE DATA RESIDES. ANSWER AND OS/8 DEVICE NAME. THEN IT WILL ASK WHAT THE FILE IS NAMED. ANSWER THE OS/8 NAME (THE .DA EXTENAION SHOULD BE OMITTED, SINCE THE GENERATOR SUPPLIES THE .DA BY ITSELF). THEN IT WILL ASK "ALL OK?". ANSWER Y OR N AS REQUIRED.

THE PROGRAM WILL THEN ASK IF THE GENERATED SORT IS CHAIN TO ANOTHER PROGRAM. ANSWER Y OR N. IF THE ANSWER WAS "Y", IT WILL ASK THE NAME OF THE PROGRAM TO BE CHAINED TO. NOTE THAT THE CHAINED TO PROGRAM MUST BE IN SAVED OR CORE IMMAGE FORMAT. ANSWER AS REQUIRED.

THE LAST QUESTION ASKED IS WETHER OR NOT THE GENERATED SORT IS TO BE SAVED. A "N" ANSWER WILL START THE SORT IMMEDIATLEY, WHILE A "Y" ANSWER WILL DO SOME INTERNAL HOUSE KEEPING AND THEN RETURN TO THE OS/8 MONITOR WITH A DOT (.).

IF THE GENERATED SORT IS TO BE SAVED, TYPE "SAVE SYS SORTXX" WHERE SORTXX IS THE OS/8 NAME OF THE GENERATED PROGRAM.

A SAMPLE OF THE SORT GENERATION DIALOGUE FOLLOWS:
TO SORT THE SECOND EXAMPLE PROGRAM FIRST BY TYPE, AND THEN BY DATE, THE FOLLOWING DIALOGUE WOULD BE REQUIRED.

```
.RUN SYS SORT

INVENT-8 SORTING PROGRAM
PLEASE INPUT SORT KEYS

WHAT FORMAT? I
STARTING AT WORD NO.  0

ALL OK? Y
MORE VARS? Y
WHAT FORMAT? I
STARTING AT WORD NO.  7
```

```
ALL OK? Y
MORE VARS? Y
WHAT FORMAT? I
STARTING AT WORD NO.  5

ALL OK? Y
MORE VARS? Y
WHAT FORMAT? I
STARTING AT WORD NO.  6

ALL OK? Y
MORE VARS? N
WHAT DEVICE DTA5
WHAT FILE PARTS
ALL OK? Y
CHAIN TO ANOTHER PROGRAN? Y

CHAIN TO WHAT PROGRAM? PRINT
ALL OK? Y

SAVE IT? Y

.SAVE SYS SORTXX
```

SEVERAL ITEMS BECOME APARENT FROM THIS EXAMPLE

1) ALWAWS START THE SORT GENERATOR WITH THE RUN COMMAND SINCE THE R COMMAND DOES NOT BRING THE CORE CONTROL BLOCK (THAT PIECE OF INFORMATION THAT TELLS THE MONITOR WHAT CORE LOCATIONS TO SAVE IN THE "SAVE SYS SORTXX" COMMAND).
2) THE DEVICE THAT CONTAINS THE DATA TO BE SORTED MUST BE A FILE STRUCTURED (READABLE AND WRITABLE) DEVICE.
3) THE SORT WILL SORT THE DATA UPON ITSELF AND IT REQUIRES NO TEMPORARY STORAGE. THE SORTED DATA (UPON THE COMPLETION OF THE SORTING PROCESS) HAS THE SAME NAME AS WHEN IT WAS STARTED.
4) THE SORT PROGRAM WILL OPERATE ***ONLY*** ON WOISAM GENERATED DATA FILES. ANY OTHER INPUT FILES WILL PRODUCE ENTIRELY UNPREDICTABE (AND PERHAPS DESASTEROUS) RESULTS.
5) WHEN SORTING A DATE, SORT ON YEAR FIRST, THEN MONTH, FOLLOWED BY THE DAY.
6) IF THE FOURTH SORT KEY WAS DESIRED, NAMELY THE VARIABLE CN THE FORMAT WOULD BE "F", STARTING AT WORD NO. 2!
7) IF THE FIFTH SORT KEY WAS DESIRED, NAMELY THE VARIABLE D1 THE FORMAT WOULD BE "A6", STARTING AT WORD NO. 8!

GOOD LUCK ON YOUR ENDEAVOR TO USE THIS SOFTWARE SYSTEM!!!
ANY COMMENTS AND/ OR SUGGESTIONS ABOUT THE SYSTEM SHOULD BE REFFERED TO THE AUTHOR THROUGH DECUS (PLEASE)

TWO SAMPLE PROGRAMS FOLLOW. THE FIRST USES MULTI AND DUMP, WHILE THE SECOND USES ROISAM, WOISAM, ISAM, AND DEVICE INDEPENDENT INPUT/OUTPUT (READ (4,N) LIST AND WRITE (4,M) LIST) AS ILLISTRATIONS.

```
C ALL FILES ARE KEYED AND SORTED ON X1 Y1 AND Z1
C
C THIS PROGRAM WILL MERGE DBASE, INASCI, AND INPUT AS FOLLOWS
C
C IF ON INASCI AND DBASE - DELETE FROM DBASE
C IF ON INASCI AND INPUT - ADD TO DBASE
C IF ON INASCI ONLY - ADD TO OUTASC
C IF ON INPUT AND DBASE - DO NOTHING
C IF ON INPUT ONLY - ADD TO OUTPUT
C IF ON DBASE ONLY - CHANGE TO TYPE 2
C IF ON ALL THREE - CHANGE TYPE TO 2
C
C
C
C THE FOLLOWING EXAMPLE USES ISAM, ROISAM, AND WOISAM
C IT ALSO USES READ (4,XXX) AND WRITE (4,XXX)
C
C
C
      COMMON ITYPE,X1,X2,IDUM,JTYPE,Y1,Y2
      DIMENSION IDUM(121)
C FORTRAN WILL ASSIGN 121 LOCATIONS IN COMMON FOR IDUM, FORCING
C     JTYPE TO 10400 WHICH IS WHERE WE WANT IT.
      CALL INITL('DBASE')
      CALL GETOP('DTA5','INPUT')
      CALL PUTOP('DTA6','OUTPUT',7)
C MUST CALL IOPEN AFTER GETOP
      CALL IOPEN('DTA1','INASCI')
C MUST CALL OOPEN AFTER PUTOP
      CALL OOPEN('DTA2','OUTASC')
C INASCI AND OUTASC NEED NOT BE ON THE SAME DEVICE TYPE, BUT
C     INASCI AND GETOP MUST HAVE CORESIDENT HANDLERS
      READ (4,1000) KTYPE,Z1,Z2
1000  FORMAT(I4,2F6.1)
C NOW ALL THE FILES HAVE THE SAME STATUS( THE FIRST RECORD IS
C     IN CORE.
10    IF (X1-Y1) 20,60,90
20    IF (Z1-X1) 30,40,50
C ON INASCI ONLY
30    WRITE (4,1000) KTYPE,Z1,Z2
      READ (4,1000) KTYPE,Z1,Z2
C IS THERE AN EOF ON INASCI?
      IF (KTYPE-999) 10,200,200
C ON INASCI AND DBASE
40    CALL DELET
C DELET WILL ALSO GET THE NEXT SEQUENTIAL RECORD
C CHECK FOR AN END OF FILE
      IF (ITYPE-999) 45,300,300
45    READ (1L4,1000) KTYPE,Z1,Z2
C CHECK FOR AN EOF
      IF (KTYPE-999) 10,200,200
C ON DBASE ONLY
```

```
50        ITYPE=2
          CALL RWRTE
C RWRTE WILL NOT GET THE NEXT RECORD
          CALL GET
          IF (ITYPE-999) 10,300,300
60        IF (X1-Z1) 70,80,30
C ON DBASE AND INPUT - DO NOTHING
70        CALL GET
          IF (ITYPE-999) 75,300,300
75        CALL GETX
          IF (JTYPE-999) 10,400,400
C ON ALL THREE
80        ITYPE=2
          CALL RWRTE
          CALL GET
          IF (ITYPE-999) 82,300,300
82        CALL GETX
          IF (JTYPE-999) 84,400,400
84        READ (4,1000) KTYPE,Z1,Z2
          IF (KTYPE-999) 10,200,200
90        IF (Z1-Y1) 30,100,110
C ON INPUT AND INASCI-ADD TO DBASE BUT KEEP IT IN SEQUENCE
100       CALL BKSPC
          ITYPE=2
          X1=Z1
          X2=Z2
          CALL PUT
C PUT WILL NOT GET THE NEXT RECORD
          CALL GET
          IF (ITYPE-999) 102,300,300
102       CALL GETX
          IF (JTYPE-999) 104,400,400
104       READ (4,1000) KTYPE,Z1,Z2
          IF (KTYPE-999) 10,200,200
C ON INPUT ONLY
110       CALL PUTX
          CALL GETX
          IF (JTYPE-999) 10,400,400
C
C
C
C MUST HANDLE EOS'S
C EOF ON INASCI
200       IF (ITYPE-999) 210,260,260
210       IF (JTYPE-999) 220,230,230
220       IF (X1-Y1) 230,240,250
C ON DBASE ONLY
230       ITYPE=2
          CALL RWRTE
          CALL GET
          GO TO 200
```

```
C ON DBASE AND INPUT
240      CALL GETX
         READ (4,1000) KTYPE,Z1,Z2
         GO TO 200
C ON INPUT ONLY
250      CALL PUTX
         CALL GETX
         GO TO 200
260      IF (JTYPE-999) 250,900,900
C
C
C BACKSPACE DBASE SINCE MAY WANT TO ADD TO IT
300      CALL BKSPC
305      IF (JTYPE-999) 310,360,360
310      IF (KTYPE-999) 320,330,330
320      IF (Y1-Z1) 330,340,350
C ON INPUT ONLY
330      CALL PUTX
         CALL GETX
         GO TO 305
C ON INPUT AND INASCI
340      ITYPE=JTYPE
         X1=Y1
         X2=Y2
         CALL PUT
         GO TO 305
C ON INASCI ONLY
350      WRITE (4,1000) KTYPE,Z1,Z2
         READ (4,1000) KTYPE,Z1,Z2
         GO TO 305
C AN EOF ON INPUT - SO ON INASCI ONLY
360      IF (KTYPE-999) 350,900,900
C
C
C
C EOF ON INASCI
400      IF (ITYPE-999) 410,460,460
410      IF (JTYPE-999) 420,430,430
420      IF (X1-Y1) 430,440,450
C ON DBASE ONLY
430      ITYPE=2
         CALL RWRTE
         CALL GET
         GO TO 400
C ON DBASE AND INPUT
440      CALL GETX
         CALL GET
         GO TO 400
C ON INPUT ONLY
450      CALL PUTX
         CALL GETX
         GO TO 400
```

```
C ON INPUT ONLY (MAYBE)
460      IF (JTYPE-999) 450,900,900
C
C
C
C THATS ALL FOLKS
C CLOSE ALL THE FILES
C ALSO CLOSE DBASE
900      CALL FINSH
         CALL PUTCL
C PUT SOME EOF'S ON OUTASC
         WRITE (4,1000) 999,0.,0.
         WRITE (4,1000) 999,0.,0.
         WRITE (4,1000) 999,0.,0.
         CALL OCLOS
         WRITE (1,1001)
1001     FORMAT('JOB XXX COMPLETE')
         CALL EXIT
         END
```

```
C THE RECORD FORMAT IS AS FOLLOWS
C THE ORDER HEADER IS TYPE 1
C THE PART HEADER IS TYPE 2
C THE DELIVERY HEADER IS TYPE 3
C VAR    FORMAT   NO WDS    P. O.     P. N.     DELIV
C 1      I1       1         =1        =2         =3
C 2      A2       1         PO1       PN1       CODE
C 3      F7.0     3         PO2       PN2       QUAN
C 4,5,6  3I2      3         DATE      DATE      DATE
C 7,8    2A6      6         VNAME     PDESC
C 9      F9.2     3         VCODE     PRICE     RCQUAN
C 10     I3       1         ACC       PN3       RCDATE1
C 11     2I2      2                             RCDATE2,3
C
C
         COMMON IM,IC,CN,ID1,ID2,ID3,D1,D2,VC,IP,IXD2,IXD3
         CALL GDATE(IMO,IDAY,IYR)
1099     FORMAT('UTILITY ROUTINE')
1009     FORMAT('WHAT COMMAND ',A6)
         WRITE (1,1099)
         READ (1,1009) X
5        IF (X-'DUMP  ') 200,8,200
C INITIALIZE THE FILE
8        CALL FOPN('DBASE')
         CALL SPOOP('MASTR1')
10       CALL GETF
         IF (IM-1) 10,20,30
20       WRITE (4,1001) IM,IC,CN,ID1,ID2,ID3,D1,D2,VC,IP,IXD2,IXD3
         CALL WRTCH
C CHECK TO SEE IF END OF REEL
         GO TO 10
30       IF (IM-3) 40,50,60
40       WRITE (4,1002) IM,IC,CN,ID1,ID2,ID3,D1,D2,VC,IP,IXD2,IXD3
         CALL WRTCH
         GO TO 10
50       WRITE (4,1003) IM,IC,CN,ID1,ID2,ID3,D1,D2,VC,IP,IXD2,IXD3
         CALL WRTCH
         GO TO 10
60       CALL SPOCL
         CALL FCLOS
70       READ (1,1004) X
1004     FORMAT('COMPRESS ALSO(YES OR NO)? ',A6)
         IF (X-'YES   ') 80,90,80
80       IF (X-'NO    ') 70,900,70
90       CALL SPIOP('MASTR1')
C LETS CALL SPIOP JUST TO BE SURE IT WON'T FAIL
         CALL ZERO('DBASE',20)
```

```
        CALL FOPN('DBASE')
        CALL SPIOP('MASTR1')
100     READ (4,1005) IM,
        IF (IM-1) 100,110,120
110     READ (4,1006) IC,CN,ID1,ID2,ID3,D1,D2,VC,IP,IXD2,IXD3
        GO TO 145
120     IF (IM-3) 130,140,150
130     READ (4,1007) IC,CN,ID1,ID2,ID3,D1,D2,VC,IP,IXD2,IXD3
        GO TO 145
140     READ (4,1008) IC,CN,ID1,ID2,ID3,D1,D2,VC,IP,IXD2,IXD3
145     CALL PUTF
        GO TO 100
150     CALL RDCHK(I)
        IF (I) 160,160,100
160     CALL FCLOS
900     WRITE (1,1013)
1013    FORMAT('DONE')
        CALL EXIT
1001    FORMAT(I4,2X,A2,F7.0,3I3,2X,2A6,2X,F7.0,3I5)
1002    FORMAT(I4,2X,I2,F7.0,3I3,2X,2A6,2X,F7.2,3I5)
1003    FORMAT(I4,2X,A2,F7.0,3I3,2X,2F6.0,2X,F7.0,3I5)
1005    FORMAT(I4)
1006    FORMAT(2X,A2,F7.0,3I3,2X,2A6,2X,F7.0,3I5)
1007    FORMAT(2X,I2,F7.0,3I3,2X,2A6,2X,F7.2,3I5)
1008    FORMAT(2X,A2,F7.0,3I3,2X,2F6.0,2X,F7.0,3I5)
200     IF (X-'RELOAD') 210,90,210
210     IF (X-'ZERO  ') 220,230,220
1010    FORMAT('DUMP,RELOAD,OR ZERO ONLY ',A6)
220     READ (1,1010) X
        GO TO 5
230     READ (1,1011) X
1011    FORMAT('ARE YOU SURE? ',A6)
        IF (X-'YES   ') 901,240,901
240     CALL ZERO('DBASE',20)
        GO TO 900
901     WRITE (1,1012)
1012    FORMAT('ERROR')
        CALL EXIT
        END
```

```
LIB81 . RL   46 8/2/72
LIB82 . RL    9 8/2/72
SORT  . SV   12 9/6/72
ISAM  . 02   39 8/2/72
ISAM  . RL    8 8/2/72
DUMP  . 02   31 8/2/72
DUMP  . RL    7 8/2/72
IOH   . RL    7 8/2/72
FLOAT . RL    5 8/2/72
WOISAM. 01   13 8/2/72
WOISAM. RL    4 8/2/72
ROISAM. 02   10 8/2/72
ROISAM. RL    3 2/16/73
POWERS. RL    3 8/2/72
UTILTY. RL    3 8/2/72
INTEGR. RL    2 8/2/72
TRIG  . RL    2 8/2/72
ATAN  . RL    2 8/2/72
MULTI . 03   13 8/2/72
MULTI . RL    3 8/2/72
IPOWRS. RL    2 8/2/72
RWTAPE. RL    2 8/2/72
IOPEN . RL    2 8/2/72
GTDATE. 02    5 8/2/72
GTDATE. RL    2 8/2/72
SQRT  . RL    2 8/2/72
NCOMP . 01    4 8/2/72
NCOMP . RL    2 8/2/72
SORT  . 02   61 9/6/72
BLURB . 04  108 8/3/72
SORT  . BN   11 9/6/72
<EMPTY>     258
  258 FREE BLOCKS
```