JOVIAL PROCEDURE LIBRARY

FOCCPAC TECH NOTE 3

APRIL 1968

12S365

CONTENTS

SECTION I

INTRODUCTION

A. INTRODUCTION

This document represents the first attempt to assemble, in a single volume, all the various descriptions (written or verbal) of the routines on the JOVIAL Procedure Library at FOCCPAC.

Special importance has been given to a standard format, indexing by function, simplicity in presentation and accuracy of information.

No pretext is made that this document is as complete as would be desired. As a result, it is requested that all users report discrepancies and/or improvement suggestions to the author, C31A, LTJG MOLTZAU.

It is also requested that new procedures, which may be of use by other programmers, be documented and given to the author so that they may be evaluated for inclusion on future libraries.

B. ORGANIZATION

1. Manual Format

An attempt has been made to standardize the contents of this manual. This makes it possible for the user to know what he can expect to find in each procedure write up and where.

It is suggested that the user place this manual in a multi-ring binder. This will greatly facilitate adding additional material and changes at a future date.

A change sheet will be issued with all changes listing the current changes and all previous changes. Each change will consist of: A change number, an effective change date, the section number under which the change occurs and the number of pages involved in a change. Space is also provided for a signature of the individual who makes the change and the date. Since each change sheet will be cumulative; i.e. listing all previous changes; the previous change sheet may be discarded. This gives the user the ability of being able to continuously verify that the appropriate changes have been made.

There are four basic divisions to the manual: The Quick Reference Index, the Introduction, the Library, and the Appendices.

a. Quick Reference Index

The quick reference index was designed to aid the user in several ways. It acts as a summary for the actual procedure write-up. It will be particularly useful to the individual who has used a particular procedure before, and is interested primarily in: how to call it, how much core it will occupy, and what other procedures it may call. It will also serve as an aide to the individual who has a procedure name and wants to look it up in detail. The index will indicate the section under which further documentation will be found.

Abbreviations used will be found listed at the end of the index. With the exception of Hollerith and STC items, only the item types are given; e.g., floating point, arithmetic, etc. In the case of the two exceptions the number of bytes that the declared item contains in the procedure is also given. These item types are indicated so that the user will know how they are declared in the procedure. The UNKNOWN listed for the accuracy of many procedures is not to imply that the procedures are of questionable accuracy, but rather that they have not been tested to determine the exact limit of their accuracy under a given set of conditions for documentation purposes. In general, the accuracy of all the procedures should be at least eight or nine decimal places.

The parameters in the call format will not necessarily match those in the procedure write up. They are, however, the identical parameter as declared in the procedure.

To aid the user in finding the desired procedure, the upper right hand corner contains the range of the procedures covered on that page.

b. Introduction

This section gives an explanation of the organization of the manual itself and the library tape. Discussions on procedures in general and how to make efficient use of this manual are also required.

c. Library

The library section is currently broken down into eight sections with room for another two should they become needed. These eight sections break down all the procedures into basic subject categories as follows:

1) Arithmetic Procedures
2) Trigonometric Procedures
3) Conversion Procedures
4) File Processing System Procedures
5) Program Maintenance and Utility Procedures
6) Navigational Procedures
7) Plotter Procedure
8) Search and Sort Procedures

The procedures are listed alphabetically by name in each of the sections. The pages are not numbered to facilitate future changes in documentation on a page by page basis.

Each page currently has a date in the bottom right hand corner indicating when it was originally written. Any pages that are changed will be so indicated by a change number and date.

Each section of procedures will have a brief introduction describing the characteristics of that particular group of procedures that are in common. This prevents consistent repetion from procedure to procedure in the write up.

Each procedure write up is broken into 7 categories as follows:

1. Purpose
2. Call Format
3. Limitation and Accuracy
4. Procedure Characteristics
5. Other Procedures Called
6. Mathematical Method Unit on Programming Technique Used
7. Example

Under the section of Call Format the parameters used are not necessarily those actually used in the procedure. Rather they are chosen, if possible to be indicative of what the parameters stand for. In the description of the parameters an indication is made as to how the parameters are defined in the procedure (e.g. floating point, 8 character hollerith, etc.)

In many cases the accuracy is not documented because no tests have been made to establish a documentable value. In most cases where applicable, the accuracy should at least be equal to eight or nine decimal places.

Under the section Procedure Characteristics: The approximate complete time for the procedure to run is given whenever possible. This time is actually measured run time and not the simple summation of all the instruction execution times. An indication is also given as to whether the time is expected to vary or not depending on the particular function that a procedure is performing.

The amount of core that is required by the procedure is indicated in Octal. This figure is for the one particular procedure only. If the procedure calls in other procedures, the total storage required will be a sum of the storage required by the initial procedure used and any other procedures that it might call.

An indication is also made under this heading whether or not the procedure is written in JOVIAL only, or also contains direct code. A pure JOVIAL procedure will greatly facilitate the change over to another computer using JOVIAL. However, it must be remembered that there are some functions that can only be performed in direct code and others that can only be written with considerable awkwardness in JOVIAL alone.

d. Appendix

Appendix A is essentially identical to Appendix 1-06 of the PIN and contains a complete listing of the procedure library by subject category. Each procedure is listed with a brief statement as to what its purpose is. This makes it possible for the user to quickly scan a number of procedures under a given subject heading in searching for the particular procedure that he wants. From this point he can go to the more detailed writeup in the manual.

It is suggested that the File Processing System Manual, NAVCOSSACT Document No. 88M902A CM-01A be placed in Appendix B. This serves an an added reference aid for the File Processing System procedures in Section 4.

2. Library Tape Format

A generalized format has also been standardized for the procedure card images on tape. The procedures requested are brought in by the compiler in alphabetical sequence. These will occur on the users listing after his main program and any procedures defined local to his program. The procedure name, which the user never sees, begins in column 17 so that it may be readily accessed by PPS functions. This card has a

sequence number of one. The second card is a blank card and is the first card pulled into the users program during compilation. This is followed by a brief two or three line, indented, statement as to the procedures purpose. A blank card separates this statement from the procedure call. Another blank card separates the procedure call from a two or three line statement indicating what other procedures this particular procedure may may call. The main body of the procedure follows immediately after another blank card.

The key sequence field contains a coded sequence number. The first three letters represent, usually, the first three letters of the procedure name except where they duplicate another procedure. In such cases usually the first two letters and the last letter are used. If this is a duplication, an appropriate set of letters were chosen with the first letter always corresponding to the first letter of the procedure name. The next three columns comprise of the sequencing field. The seventh column of the sequence field will contain an F, W, or A. This indicates where this particular procedure originally came from; FOCCPAC, NAVCOSSACT in Washington, or FOCCLANT, respectively. The last column of the key sequence field indicates under what section in the documentation the procedure is found. For example, a 3 would indicate that the procedure is one of the conversion procedures and is described in that section of the manual.

The last card of every procedure will be so indicated by the comment END PROC "name" beginning in column 50.

C. PROCEDURE USAGE

A few facts should be brought out concerning the use of these procedures in general before a user tries to make use of them.

Functions and procedures can both be classed under the heading of closed subroutines. The names of these routines must be spelled exactly as indicated in the call parameter. The other parameters of the call may be spelled in any manner desired by the user. The most important factor regarding the input parameters is the order in which they are used. The actual parameters used by the programmer need not be of the same type as indicated in the parameter definition, but should be of the same form.

In dealing with functions and function names it should be clearly understood that the function name, whether it be SIN, COS, ARCOS, ALPHCON, or anything else, is the function name and function name only - not a simple item. Within all functions there is a simple item declared with the same name as the function. Though the item name and function name are synonmous this does not mean that they are the same, but rather they are unique entities unto themselves. This item declared within the function and carrying the same name as the function will contain the output parameter from the function.

In the description of the functions in this document, the function name is obviously just that. But how the simple item, with the same name as the function, and containing the output parameter is declared, is also indicated. It should be understood that in the case of the following example:

> ARCCOS = the function name and is a floating point number, in radians, representing the arccosine of NUM.

that the portion of the description, "is a floating point number, in radians, representing the arccosine of NUM" refers to the simple item declared in the function as ARCCOS and not the function name.

Normally "XX", to which the function call is set equal to, is declared in the same manner as the output items. However, this does not imply that "XX" must be declared in an identical format, as this is a programmers option.

Functions may be nested within functions or procedures. For example in the case:

```
XX = SIN[RADIANS(DEG)]  $
```

The input parameter (DEG) is in degrees. The function RADIANS converts the degrees to radians and the SIN function computes the sine of the angle. Thus for an input parameter the user may use a function call.

D. USING THIS MANUAL EFFICIENTLY

1. If you are looking to see if there is an available procedure under a given subject category, or, you just want to know in general what is available, use Appendix A.

2. Once a given procedure name is known and you want detailed information on its characteristics, look it up alphabetically under the appropriate subject section.

3. If you are familiar with the procedure, know its general characteristics and limitations, and just want such information as: how to call it, definition of the parameter, number of words it will occupy, accuracy, running time, or what other procedure that it may call; look it up alphabetically under the quick reference index. If you find the material imcomplete, the section under which it is located is indicated where it may be referred to in detail.

SECTION II

QUICK REFERENCE INDEX

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
ALPHCON(ARGUM,WHICH) [proc]	Converts IBM 7090 Code to OPCON code and vice versa. ALPHCON = Converted output value, in HOL 8. ARGUM = Input value to be converted, in HOL 8. WHICH = Indicates: 0 = OPCON to BCD 1 = BCD to OPCON	COMP 43	UNKN	3	REMQUO
ANGLINE(DIST1,DIST2,XX, YY, ANGLE) [proc]	Draws a straight line between point A and point B on a given vector. DIST1 = Distance from X,Y to A, in Flt Pt Inch. DIST2 = Distance from X,Y to B, in Flt Pt Inch. XX = X Coord of base Pt, in Flt Pt Inch. YY = Y Coord of base Pt, in Flt Pt Inch. ANGLE = Vector angle, measured counter clockwise from the X axis, in Flt Pt Deg.	N/A 33	UNKN	7	COSIN DRAW MOVE RADIANS SIN
ANGPOSIT(DIST,XB,YB,ANGLE = XX,YY) [proc]	Returns the X and Y coordinates of a point on a given vector. DIST = Distance from the base Pt, XB: in Flt Pt Inch. XB = X Coord of base, in Flt Pt Inch. YB = Y Coord of base, in Flt Pt Inch. ANGLE = Angle of vector having a base point of XB, YB; in Flt Pt Deg. XX = X Coord of the defined point, in Flt Pt Inch. YY = Y Coord of the defined point, in Flt Pt Inch.	N/A 17	UNKN	7	COSIN RADIANS SIN
ARC(RAD1,XX,YY,ANG1,ANG2) [proc]	Draws an arc defined by the input parameters. RAD1 = Radius of the arc, in Flt Pt Inch. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch. ANG1 = Starting angle, in Flt Pt Deg. ANG2 = Terminating angle, in Flt Pt Deg.	N/A 170	UNKN	7	COSIN DRAW NORM360 RADIANS SIN

CALL TYPE	DESCRIPTION	ACCUR / #WRDS	TIME MSEC	SEC	OTHERS USED
ARCCOS(XX) [proc]	Computes the arccosine of a number. ARCCOS = Output value, in Flt Pt Rad. XX = Input value, in Flt Pt.	$\frac{10^{-5}}{15}$	1-3	2	ARCSIN
ARCSIN(XX) [func]	Computes the arcsine of a number. ARCSIN = Output value, in Flt Pt Rad. XX = Input value, in Flt Pt.	$\frac{10^{-5}}{89}$	1-3	2	
ARCTAN(XX) [func]	Computes the arctangent of a number. ARCTAN = Output value, in Flt Pt Rad. XX = Input value, in Flt Pt.	$\frac{10^{-5}}{89}$	1.6	2	
ASPECIAL(XX,YY,PEN,HEIGHT, SINC,COINC,SYMB) [proc]	Draws one of the special characters from the special character set. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch. PEN = Pen position, in Int according to: Ø = up 1 = down HEIGHT = Character height, in Flt Pt Inch. SINC = Sine of the angle of inclination, in Flt Pt. COINC = Cosine of the angle of inclination, in Flt Pt. SYMB = An Int value specifying the special character desired. See end of Index for Symbols.	$\frac{N/A}{UNKN}$	UNKN	7	PLOT
AXIS(XEXT,YEXT) [proc]	Draws a line from Coord point Ø,Ø to X,Ø and from Ø,Ø to Y,Ø. XEXT = Length of Xaxis, in Flt Pt Inch. YEXT = Length of Yaxis, in Flt Pt Inch.	$\frac{N/A}{12}$	UNKN	7	DRAW MOVE
BARGRAF(BGRØ,BGR1,BGR2, BGR3,BGU) [proc]	Draws a shaded or unshaded bar. BGRØ = X Coord of lower left corner of the bar in Flt Pt Inch.	$\frac{N/A}{114}$	UNKN	7	DRAW MOVE

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	BGR1 = Y Coord of lower left corner of the bar, in Flt Pt Inch. BGR2 = Height of basic bar, in Flt Pt Inch. BGR3 = Height of the shaded portion, in Flt Pt Inch. BGU = Number of Int units the bar is wide: 1 unit = 0.2 Inch.				
BETTERXY(X1,Y1,X2,Y2 = [proc]	Determines which of the two X,Y Coord supplied is closest to the present pen position. X1 = X Coord of 1st set, in Flt Pt Inch. Y1 = Y Coord of 1st set in Flt Pt Inch. X2 = X Coord of 2nd set, in Flt Pt Inch. Y2 = Y Coord of 2nd set, in Flt Pt Inch. XB1 = X Coord of closest se , in Flt Pt Inch. YB1 = Y Coord of closest set, in Flt Pt Inch. XB2 = X Coord of farthest set, in Flt Pt Inch. YB2 = Y Coord of farthest set, in Flt Pt Inch.	N/A 39	UNKN	7	SQRT
BINSRCH(SRTBL,SRWDS,ARTBL, ARWDS = SRENT,SRALT) [proc]	Performs a binary search, using a single or multiple word key, on a sorted serial table. SRTBL = Name of the table to be searched. SRWDS = Number of words per entry of table SRTBL. ARTBL = Table name that contains the argument. ARWDS = Number of words per entry of table ARTBL. SRENT = Entry at which the match was found or point where the argument should be inserted if no match was found. SRALT = Statement label to which a transfer is made if a match is not found.	N/A UNKN	UNKN	8	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
CENTER(CEND,CENC,CENH) [func]	Supplies the X or Y Coord at which a label is to begin if it is to be centered on given X or Y point. CENTER = X or Y Coord that the label should begin at, in Flt Pt. CEND = Point over which the label is to be centered, in Flt Pt Inch. CENC = Number of characters in the label, in Flt Pt. CENH = Height of characters, in Flt Pt Inch.	N/A 8	UNKN	7	
CIRCLE(RADI,XX,YY) [proc]	Draws a circle around Coord X,Y with the given radius. RADI = Radius of the circle in, Flt Pt Inch. XX = X Coord of center, in Flt Pt Inch. YY = Y Coord of center, in Flt Pt Inch.	N/A 8	UNKN	7	ARC
CK'STATS(FC,RESULT,HIST = ALT.) [proc]	Waits for a non-busy status to be returned by an I/O device, checks for other non-normal stati and takes appropriate action. FC = File code of file being checked in Int. RESULT = Ø; then if the status of FC is other than 0, 1, 2, 3, of 13 the task will be terminated with TERM REAS 37XX where "XX" is the status found. = 1; then, if illegal stati are found, an abnormal termination will occur only if total errors previously detected by CK'STATS is 99. HIST = File code of list tape for every message, in Int. If equal to Ø no logging will be done. ALT. = Statement label to which CK'STATS will return if a segment mark or end-of-file is found.	N/A 46	UNKN	5	INTHI

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
CLEAR(WHERE,NREGS,TYPE) [proc]	Clears an area of core to +Ø or blanks. WHERE = The first word of core to be cleared, in Int. NREGS = The number of words to clear, in Int. TYPE = Status item, if equal to: 1; clear to +Ø. 2; clear to blanks. If not equal to 1 or 2 will return without clearing.	N/A 30	UNKN	5	
CORE(FWA,NWD,NAME,NDWPE, FORM) [proc]	Dumps a specified memory area on file Ø4. FWA = First word address in Int. NWD = Number of words to be dumped in Int. NAME = Up to 8 OPCON characters that may be printed as an Ident in the control line in Hol 8. NDWPE = Number of words per sub-section, or entry, in Int. FORM = Format of data to be printed. Ø = octal 1 = OPCON	N/A 140	UNKN	5	
COSH(XX) [func]	Computes the hyperbolic cosin of a number. COSH = Output value, in Flt Pt. XX = Input value, in Flt Pt.	10^{-11} 11	1.8	2	FEXP
COSIN(XX) [func]	Computes the cosine of a number. COSIN = Output value, in Flt Pt. XX = Input value, in Flt Pt Rad.	$.5X10^{-5}$ 46	2.0	2	SIN
COTAN(XX) [func]	Computes the cotangent of a number. COTAN = Output value, in Flt Pt. XX = Input value, in Flt Pt Rad.	10^{-10} 11	3.5	2	TAN
CSCAN(MOD = FTYP,FCOL,NCOL) [proc]	Locates and classifies fields within an unpacked (one character per word) image that generally represent card columns.	N/A 430	UNKN	5	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	MOD = Int, indicating when CSCAN is to return to the calling program. ∅ = locate next field and return. 1 = return where a $ has been encountered or all columns have been examined. FTYP = Int specifying the type of field located by the scan. ∅-15 indicates a separator. 22-30 indicates data. See procedure writeup for individual codes. FCOL = Int, specifying the first column of a field containing a character that is not separator, space, or part of a field definition. NCOL = Int, specifying the number of columns comprising the field. Will be ∅ for separators.				
CURVE(PTAB,PSPAC,PSTRT, PCHR,PCHT) [proc]	Draws a smooth curve between N equally spaced values of Y. PTAB = Table name of value to be ploted. Nent must equal the number of points. PSPAC = X spacing between values of Y, in Flt Pt Inch. PSTRT = Position of the first point from the X axis, in Flt Pt Inch. PCHR = Number of the special character set to mark the points within Arit. See end of Index for symbols. PCHT = Height of character, in Flt Pt Inch. If = ∅ no characters will be printed.	N/A 10	UNKN	7	PLOT
DASH(XASH,YASH,XASH1,YASH1, DASLEN,DOTLEN) [proc]	Draws a dashed line from X_0,Y_0 to X_1,Y_1. XASH = X Coord of the init'al Pt, in Flt Pt Inch. YASH = Y Coord of the initial Pt, in Flt Pt Inch. XASH1 = X Coord of the final Pt, in Flt Pt Inch.	N/A 56	UNKN	7	DRAW FIX MOVE SQRT

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	YASHI - Y Coord of the final Pt, in Flt Pt Inch. DASLEN - Dash length, in Flt Pt Inch. DOTLEN - Length of space, in Flt Pt Inch.				
DDSORT(ADDR,NUMB,EWDS, SWDS,FWSK) [proc]	Sorts logically in ascending order, entries of a single or multiple word per entry serial table via a key field in the entry. ADDR - Location of the table, in Int. NUMB - Numt of the table, in Int. EWDS - Number of words per entry in the table (NWDSEN or some calculated value), in Int. SWDS - Number of full words in the sort key, in Int. FWSK - Starting word of the sort key, in Int.	N/A 62	UNKN	8	
DEGREES(RAD) [func]	Converts radians to degrees. DEGREES - The output value, in Flt Pt Deg. RAD - The input value, in Flt Pt Rad.	10^{-9} 3	UNKN	5	
DELET(INDX,OPER) [proc]	Reallocates core buffer area due to opening or closing a file in FPS. INDX - Index to file entry in tables FDES and CBUF, in Int. OPER - Entrance point to DELET, in Int. 1 = Entered from procedure EXIT. 2 = Entered from procedure ENTER.	N/A 421	UNKN	4	
DESUNK(LATA,LONGA,CRSA, DTIME,SOA = LATB, LONGB,CRSB) [proc]	Calculates a new position and course (great circle) when given position, course, time, and SOA, assuming constant speed for the time interval specified and great circle track. LATA - Original Lat, in Flt Pt Rad. LONGA - Original Long, in Flt Pt Rad.	UNKN 125	UNKN	7	ARCCOS ARCSOM COSIN SIN

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	CRSA = Original course, in Flt Pt Rad. DTIME = Number of hours ship is to travel, in Flt Pt. SOA = Speed, in Flt Pt Kt. LATB = New Lat, in Flt Pt Rad (negative if south). LONGB = New Long, in Flt Pt Rad. GRSB = Great circle course, in Flt Pt Rad.				
DLOAD(TP,DR,HIST,FMS'SW, TRK1,KEY,DPE,LAST = NUMT,NUMR) [proc]	Reads unpacked records from tape and places them on disc. TP = Input file code, in Int. DR = Random disc output file code, in Int. HIST = Output tape file code for listing error messages, in Int. FMS'SW = A Bool item indicating input tape format: 0 = Not in FMS format. 1 = In FMS format. RDBUF = Table name for n+1 words per entry where "n" = number of words in a key. TRK1 = Beginning track number, in Int. KEY = Name of the Key table to be built. WPE = Number of words per entry (n+1) in KEY, in Int. LAST = 'LOC (statement label) where the procedure is to return to. NUMT = Number of tracks used, in Int. NUMR = Number of records packed, in Int.	N/A 260	N/A	8	CK'STATS
DRAW(XX,YY) [proc]	Draws a line from the present pen Coord to the designated Coord. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch.	N/A 5	UNKN	7	PLOT
DUMP(FADDR,LADDR,ID,FORM) [proc]	Initiates a dump service request to SSEC for that portion of memory requested. FADDR = Address of the first word to be dumped, in Arit. LADDR = Address of the last word to be dumped, in Arit. ID = User identification of the	N/A 23	UNKN	5	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	dump, in Hol 2. FORM = Format of the output, in Arit. 0 = Octal 1 = OPCON				
ENTER(FDI) [proc]	Opens the file designated and allocates an I/O work area in core to be used for the file being opened, in FPS. FDI = File description item, in Hol 50. See procedure writeup for details.	N/A 1017	UNKN	5	DELET I'O LADDL TYPF
ENTERF(FDI) [proc]	Opens an IPS master tape for use with FPS library procedures by bypassing the first two records. FDI = File description item, in Hol 50. See procedure writeup for details.	N/A 11	UNKN	3	ENTER
ENY(DUM'FILE) [func]	Determine the current entry position of any file in FPS. ENY = Entry position, in Int. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 14	UNKN	5	FDES'DENT
EOF(DUM'FILE) [func]	Tests for an end-of-file on an FPS tape. EOF = 1; for end-of-file present. Ø; for end-of-file not present. In Bool format. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 14	UNKN	5	FDES'DENT
EOR(DUM'FILE) [func]	Tests for an end-of-record on an FPS pate. EOR = 1; for end-of-record present. 0; for end-of-record not present. In Bool format. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 17	UNKN	5	FDES'DENT

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
EOS(DUM'FILE) **[func]**	Tests for an end-of-segment on an FPS tape. EOS = 1; for segment mark present. Ø; for segment mark not present. In Bool format. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 17	UNKN	5	FDES'ENT
EXIT(DUM'FILE) **[proc]**	Closes the processing of an FPS file. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 24	UNKN	5	DELET FDES'ENT I'O RWRIT
FBINRY(FOPCOD) **[func]**	Converts numeric values of OPCON code to its integer equivalent. FBINRY = Integer equivalent, in Arit. FOPCOD = Input parameter, in Hol 8.	100% 42	UNKN	3	
FDES'ENT(DUM'FILE) **[func]**	Obtain the entry number in table FDES for the file designated in the call in FPS. FDES'ENT = Entry number, in Int. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 15	UNKN	5	TYPE
FEXP(FUU) **[func]**	Raises the constant E to a given power. FEXP = Output results, in Flt Pt. FUU = Input parameter, in Flt Pt.	10^{-10} 56	.85	1	
FIGURES(XX,YY,COUNT, HEIGHT,ANGLE, CARACTOR) **[proc]**	Draws up to 8 characters from the OPCON character set. XX = X Coord of 1st character, in Flt Pt Inch. YY = Y Coord of 1st character, in Flt Pt Inch. COUNT = Number of characters to be printed, (1 to 8), in Arit. HEIGHT = Character height, in Flt Pt Inch.	N/A 11	UNKN	7	PLOT

CALL TYPE	DESCRIPTION	ACCUR FWRDS	TIME MSEC	SEC	OTHERS USED
	ANGLE = Angle of the characters from the + X axis, in Flt Pt Deg. CARACTOR = Characters to be plotted, in Hol 8.				
FL2HO(FLOTE,FIRST,NUMB) [proc]	Converts a Flt Pt item to OPCON code. FLOTE = Input parameter, in Flt Pt. FIRST = First entry of a table that the OPCON conversion will go into, in Int. NUMB = Number of table entries the converted value is to occupy, in Int.	100% 80	UNKN	3	IN2HO OC2HO
FOPCON(FBININ) [func]	Converts an integer item to an 8 byte OPCON item. FOPCON = Output parameter, in Hol 6. FBININ = Input parameter, in Arit.	10^{-9} 77	UNKN	3	
FRSTGRAF(PLOTTER,OUTFILE, WORK,STRT'PT) [proc]	Plotter housekeeping routine. Must be called before any other plotter procedure. PLOTTER = 0; for 30 Inch Plotter. 1; for 11 Inch Plotter. In Flt Pt. OUTFILE = Output file code for the generated plotter tape, in Arit. WORK = Table name allocating a work area for the system. Nent should be as large as possible and at least 8. If Nent = 1, locations 40000_8 to 77776_8 will be used. STRT'PT = Starting point from the right margin, in Flt Pt Inch.	N/A 33	UNKN	7	DRAW NUGRAF PLOT TICKS
FX2HO(FIN,FRA,FIRST,NUMB, NUMPL) [proc]	Converts an Arithmetic item to OPCON code. FIN = Int portion of the input parameter to be converted, in Int. FRA = Fractional portion of the input parameter to be converted, in Int. FIRST = First entry of a table that the OPCON conversion will to into, in Int.	100% 50	UNKN	3	IN2HO

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	NUMB = Number of table entries that the converted value is to occupy, in Int. NUMFL = Number of table entries allocated for the fractional portion of the converted value, in Int.				
GET'CARD(CD'FL,ER'FL = EDFL.) [proc]	Reads cards, double buffered, and unpacks them as required by CSCAN and the "XX2XX" conversion procedures. CD'FL = Card reader input file code, in Int. ER'FL = List tape file code for error, in Int. EDFL. = Statement label specifying where the procedure is to return control to if a segment mark or EOFL is encountered.	N/A 104	UNKN	5	CK'STATS
GRCRBL(LATA,LONGA,LATB, LONGB,IND = DIST, CRSE) [proc]	Calculates either the Great Circle or Rhumb Line course and distance between two given Pts. LATA = Lat of departure Pt. in Flt Pt Rad. LONGA = Long of departure Pt, in Flt Pt Rad. LATB = Lat of destination, in Flt Pt Rad. LONGB = Long of destination, in Flt Pt Rad. IND = 0; for Great Circle computation 1; for Rhumb Line computation. In Boolean Format. DIST = Distance from departure point to destination, in Flt Pt Rad. CRSE = Initial course to sail, in Flt Pt RAD.	UNKN 270	UNKN	6	ARCCOS ARCSIN ARCTAN COSIN

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
HL2FL(FIRST,NUMB, = FLT) (proc)	Converts OPCON coded data to a Flt Pt item. FIRST = First entry of a table that the OPCON coded byte is in, in Int. NUMB = Number of table entries to be converted, in Int. FLT = Converted output item, in Flt Pt.	UNKN / 116	UNKN	3	HL2IN
HL2FX(FIRST,NUMB = SIGN, INT, FRAC) (proc)	Converts OPCON coded data to an Arit item. FIRST = First entry of a table that the OPCON coded byte is in, in Int. NUMB = Number of table entries to be converted, in Int. SIGN = 1: for negative. = 0: for positive. In Int. INT = Int portion of the converted value, in Int. FRAC = Fractional portion of the converted value, in Int.	UNKN / UNKN	UNKN	3	HLCON
HL2HL(FIRST,NUMB = HOL) (proc)	Packs 8 bytes of OPCON coded data from a table to an 8 byte OPCON item. FIRST = First entry of a table that the OPCON coded byte is in, in Int. NUMB = Number of table entries to be converted (1 to 8), in Int. HOL = Packed OPCON item, in Hol 8.	100% / 46	UNKN	3	
HL2HO(HOL,FIRST,NUMB) (proc)	Unpacks an 8 byte OPCON coded item and places it in a one byte per entry table. HOL = OPCON item to be unpacked, in Hol 8. FIRST = First table entry that the first byte is to be placed in, in Int. NUMB = Number of table entries to be used in the unpacking, in Int.	100% / 33	UNKN	3	

CALL TYPE	DESCRIPTION	ACCUR %WRDS	TIME MSEC	SEC	OTHERS USED
HL2IN(FIR,NO = INT) [proc]	Converts OPCON coded data to a signed decimal integer item. FIR = First table entry that the first byte is to be placed in, in Int. NO = Number of table entries to be used in the unpacking, in Int. INT = Converted output value, in Int.	100% 76	UNKN	3	
HL2OC(FIRST,NUMB = OCT) [proc]	Converts OPCON coded data to an unsigned Octal integer. FIRST = First table entry that the first byte is to be placed in, in Int. NUMB = Number of table entries to be used in the unpacking, in Int. OCT = Converted output item, in Int.	100% 46	UNKN	3	
HL2ST(FIRST,NUMB = TRC) [proc]	Converts OPCON coded data to a simple STC coded item. FIRST = First table entry that the first byte is to be placed in, in Int. NUMB = Number of table entries to be used in the unpacking, in Int. TRC = Converted output item, in STC 8.	100% 24	UNKN	3	
IN2HO(INT,FIRST,NUMB) [proc]	Converts a simple Int item to OPCON code. INT = Input value to be converted, in Int. FIRST = First entry of a table that the OPCON coded item is placed in, in Int. NUMB = Number of table entries the converted value is to occupy, in Int.	100% 50	UNKN	3	REMQUO
INTHI(INT = HOL) [proc]	Converts an unsigned integer item to an 8 byte OPCON item. INT = Input value to be converted, in Int.	100% 27	UNKN	3	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	HOL = Converted output value, in Hol 8.				
INTERRUPT(=FAULT,INSTRCTN, ADDRESS) [proc]	Interrogates the interrupt status for internal (arithmetic faults). FAULT = Fault type, in Arit. 1 = Divide 2 = Shift 3 = Overflow 4 = Exponent (overflow) 5 = Exponent (underflow) 6 = all others INSTRCTN = Upper or lower address indicator, in Bool 0 = upper address 1 = lower address ADDRESS = Address where the fault occurred, in Arit.	N/A 8	UNKN	5	
I'O(ENTNR,DESOP) [proc]	Handles all periphal device manipulation and actual data transmission for the procedures of the FPS system. ENTNR = File description entry number in table FDES, in Int. DESOP = I/O operation desired, in Int. 1 = Open File 2 = Read 3 = Write 4 = Position 5 = Close File 6 = Write Segment Mark	N/A 1016	UNKN	4	LOST TYPE

	CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
LADDL	[func]	Obtain the address plus 100 of the last memory word loaded by the SSEC loader in FPS. LADDL = Address + 100 of the last address loaded.	N/A 11	UNKN	4	
LASTGRAF	[proc]	The last procedure called in making use of the plotter procedures. Performs housekeeping chores.	N/A 12	UNKN	7	PLOT PRESENT
LOGNAT(UU)	[func]	Computes the natural logarithm of a number. LOGNAT = Output value, in Flt Pt. UU = Input value, in Flt Pt.	100% 61	.85	1	
MOVE(XX,YY)	[proc]	Positions the plotter pen at the indicated X,Y Coord without drawing a line. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch.	N/A 5	UNKN	7	PLOT
NORM360(ANGLE)	[func]	Normalizes any angle supplied in degrees to the range of 0 to 360. NORM360 = Normalized value, in Flt Pt Deg. ANGLE = Input value, in Flt Pt Deg.	100% 12	UNKN	3	
NUGRAF(XX,YY)	[proc]	Clears and housekeeps plotter routines at the end of each block. XX = New X Coord, in Flt Pt Inch from the previous X,Y Coord. YY = New Y Coord, in Flt Pt Inch from the previous X,Y Coord.	N/A 7	UNKN	7	PLOT
OC2HO(OCT,FIRST,NUMB)	[proc]	Converts an unsigned octal integer to OPCON code. OCT = Octal input parameter, in Int. FIRST = First entry of a table that the OPCON conversion will go into, in Int.	100% 24	UNKN	3	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	NUMB = Number of table entries the converted value is to occupy, in Int.				
OPCTOB(HOL) [func]	Converts an OPCON coded number to its floating point equivalent. OPCTOB = Converted output value, in Flt Pt. HOL = Input value, in Hol 8.	100% 150	3-16	3	
PCL(LATOP,LNGOP,FRMAT = LATRN,LNGRN) [proc]	Converts OPCON coded latitude and longitude to signed floating point radians. LATOP = Lat, in Hol 8. LNGOP = Long, in Hol 8. FRMAT = Format indication, in Arit. 0 = 5 character Lat = 6 character Long 1 = 7 character Lat = 8 character Long. LATRN = Converted Lat, in Flt Pt Rad. LNGRN = Converted Long, in Flt Pt Rad.	UNKN 212	UNKN	6	
PGC(LAT1,LON1,LAT2,LON2, FORMT = GCDEG,GCNMI) [proc]	Calculates the great circle distance between two points. LAT1 = Lat of the first point, in Hol 8. LON1 = Long of the first point, in Hol 8. LAT2 = Lat of the second point, in Hol 8. LON2 = Long of the second point, in Hol 8. FORMT = Format indicator, in Arit 0 = 5 character Lat 6 character Long 1 = 7 character Lat 8 character Long GCDEG = Great Circle distance, in Flt Pt Rad. GCNMI = Great Circle distance, in Flt Pt Naut Mi.	UNKN 62	UNKN	6	ARCOS ARCSIN COSIN PCI SIN
PLA(LTCK1,FRMT1 = LATIN) [proc]	Checks magnitude and format validity of Latitude in OPCON code. LTCK1 = Lat to be checked, in Hol 8.	N/A 55	UNKN	6	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	FRMT1 = Format indicator, in Arit. 0 = 5 character Lat 1 = 7 character Lat LATIN = Validity check indicator, in Arit. 0 = Invalid 1 = Valid 2 = blank				
PLN(LNCK2,FRMT2 = LNGIN) [proc]	Checks magnitude and format validity of Longitude in OPCON code. LNCK2 = Long to be checked, in Hol 8. FRMT2 = Format indicator, in Long 0 = 7 character Long 1 = 8 character Long LNGIN = Validity check indicator, in Arit. 0 = Invalid 1 = Valid 2 = Blank	N/A 52	UNKN	6	
PLOT(PP) [proc]	Never called by the user. This is a central routine used by the plotter procedures.	N/A 1056	N/A	7	COSIN RADIANS SIN
POINTER(X1,Y1,X2,Y2,SIZE) [proc]	Draws a line from X1,Y1, to X2,Y2 with an arrow-head located at X2,Y2 in designated Size. X1 = X Pt at which to start the line, in Flt Pt Inch. Y1 = Y Pt at which to start the line, in Flt Pt Inch. X2 = X Pt to start the arrow head, in Flt Pt Inch. SIZE = Length size of the arrow head, in Flt Pt Inch.	N/A 33	N/A	7	ASPECIAL SQRT
POSITION(DUM'FILE,DSEGD, DCRED,DENTD)	Positions an input file to a particular entry in the file in FPS. DUM'FILE = File description item, in Hol 50. See procedure write up for details. DSEGD = Desired segment number, in Int. DRECD = Desired record number, in Int. DENTD = Desired entry number, in Int.	N/A 151	N/A	4	FDES'ENT I'O TYPE

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
POSITP(LATA,LONGA,LATB, LONGB,SOA,TAB,TAP, IND = LATP,LONGP) [proc]	Solves the Lat and Long of an unknown point along a Gt Circle or Rhumb Ling Course. LATA = Lat of Dep Pt, in Flt Pt Rad. LONGA = Long of Dep Pt, in Flt Pt Rad. LATB = Lat of Dest Pt, in Flt Pt Rad. LONGB = Long of Dest Pt, in Flt Pt Rad. SOA = Speed of advance from Dep Pt to the unknown point in Naut Mi and tenths of Naut Mi, in Arit. TAB = Time in Hr and tenths, required to sail from Dep Pt to Dest, in Arit. TAP = Time in Hr and tenths, required to sail from Dep Pt to Unkn posit, P, in Arit. IND = Boolean indicator as follows: 0 = Great Circle Computation 1 = Rhumb Line Computation LATP = Lat of Unkn posit, P, in Flt Pt. LONGP = Long of Unkn posit, P, in Flt Pt.	UNKN 437	UNKN	3	ARCCOS ARCSIN ARCTAN COSIN SIN SQRT
PRESENT(= XX,YY) [proc]	Supplies the present X and Y Coord of the pen. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch.	N/A 7	UNKN	7	PLOT

CALL TYPE	DESCRIPTION	ACCUR FWRDS	TIME MSEC	SEC	OTHERS USED
RADIANS(DEG) [func]	Converts angular degrees and fractions to Radians. RADIANS = Output value, in Flt Pt. DEG = Input value, in Flt Pt degrees.	10-11 3	UNKN	3	
READ(DUM'FILE,DUM'TABL) [proc]	Transfer an entry of an input file to the first entry of a serial table in FPS. DUM'FILE = File Description Item, in Hol 50. See procedure write up for details. DUM'TABL = Table name of the serial table that the entry will be placed in.	N/A 238	UNKN	4	FDES'ENT I'O TYPE
READF(DUM'FILE,DUM'TABL) [proc]	Reads a record from an IPS format tape, converts it to FPS format 4 supplying it to the user one set at a time. DUM'FILE = File Description item, in Hol 50. See procedure write up for details. DUM'TABL = Table name of the serial table that the entry will be placed in.	N/A 352	UNKN	4	FDES'ENT I'O TYPE
REC(DUM'FILE) [func]	Determines the current record of any file in FPS. REC = Current record in the given file, in Int. DUM'FILE = File Description item, in Hol 50. See procedure write up for details.	N/A 13	UNKN	4	FDES'ENT TYPE
REEL(FC) [func]	Queries the TINF table and returns the reel number assigned to a given magnetic tape file. REEL = Reel number, right justified, in Hol 8. FC = File code, in Int.	N/A UNKN	UNKN	5	FOPCON
RELEASE(DUM'FILE) [proc]	Skips over existing records in an entry, in FPS. DUM'FILE = File Description item, in Hol 50. See procedure write up for details.	N/A 25	UNKN	4	FDES'ENT RWRIT I'O TYPE

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
REMQUO(DND,SOR = QUO,REM) [proc]	Performs Int division and produces a separate quotient and remainder. DND = Dividend, in Int. SOR = Divisor, in Int. QUO = Quotient, in Int. REM = Remainder, in Int.	COMP 11	UNKN	1	
RESTART(LVP,PVP] [proc]	Initiates a request to SSEC to establish restart points. LVP = Last valid restart point, in Arit. PVP = Penultimate valid restart point, in Arit.	N/A 5	UNKN	5	
RWRIT(DUMI) [proc]	Writes the contents of a file's I/O work area onto tape, in FPS. DUMI = Entry number in Table FDES for the file to be processed, in Int.	N/A 15	UNKN	4	I'O TYPE
SECURITY(FIIL = CLASS) [proc]	Initiates a request to SSEC to determine the security classification of a file. FIIL = File code of the desired file, in Arit. CLASS = A status item indicating the file classification. Ø = Unclassified 1 = Confidential 2 = Secret 3 = Top Secret	N/A 5	UNKN	5	
SEG(DUM'FILE) [func]	Determines the current segment of any file in FPS. SEG = A counter containing the segment number, in Arit. DUM'FILE = File Description item, in Hol 50. See procedure write up for details.	N/A 14	UNKN	4	FDES'ENT TYPE
SIN(XX) [func]	Computes the Sine of any angle supplied in radians. SIN = Output value, in Flt Pt. XX = Input value, in Flt Pt Radians.	10^{-10} 50	1.25	1	

	CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
SINH(UU)	[func]	Computes the hyperbolic sine of a number. SINH = Output value in, Flt Pt. UU = Input value in, Flt Pt.	$\frac{10^{-11}}{12}$	1.8	2	FEXP
SNOCTAL	[proc]	Provides for an unlimited number of octal correction and core dump cards per run.	N/A 532	UNKN	5	
SORT(DUM'FILE,DUMBFILE)	[proc]	Sorts the records of an input file according to the key item value in each file in FPS. DUM'FILE = Input or Output File Description item, in Hol 50. See procedure write up for details. DUMBFILE = Output or input File Description Item, in Hol 50. See procedure write up for details.	N/A 362	UNKN	4	FDES'ENT
SPECIAL(XX,YY,PEN,HEIGHT, ANGLE,SYMBOL)	[proc]	Draws a character from the special character set centered on Pt X,Y. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch. PEN = Position of the pen, in Arit. Ø = up 1 = down HEIGHT = Character height, in Flt Pt Inch. ANGLE = Slope of the symbol, in Flt Pt Deg. SYMBOL = Number of the special character, in Arit. See end of Index for symbols.	N/A 7	UNKN	7	PLOT
SQRT(XX)	[func]	Calculates the square root of a number. SQRT = Output value, in Flt Pt. XX = Input value, in Flt Pt.	UNKN 58	.75	1	
SRC(NAME = XREF,XENT)	[proc]	Locates an identifier in the PPS III Compool directory. NAME = Identifier to be located, in 1 to 8 characters. in Hol 8. XREF = Value from the cross	N/A 46	UNKN	8	

CALL TYPE	DESCRIPTION	ACCUR	TIME MSEC	SEC	OTHERS USED
	reference item in the directory entry containing the identifier equal to NAME, in Int. XENT = Number of the directory entry containing the identifier equal to NAME, in Int.				
STAT'CK(DITEMF,DITEMT,XX) [proc]	Generates a typewriter comment for all data transmission resulting in other than a normal or EOF status. DITEMF = UNKNOWN DITEMT = UNKNOWN XX = UNKNOWN	N/A 56	UNKN	5	
STATS(CORE,SKIP,WONT,TYPE) [func]	Computes the Arit mean, Geometric mean, harmonic mean, mode, medium, variance, and standard deviation of a set of values assumed normal in distribution. STATS = Output of routine, in Flt Pt. CORE = Table name containing the values to be operated on. SKIP = Words per entry in table CORE, in Int. WONT = Relative word-in-entry of values to be operated on in table CORE, in Int. TYPE = Status item specifying the operation desired. 0 = Mean 1 = Mode 2 = Median 3 = Standard Deviation 4 = Variance 5 = Geometric Mean 6 = Harmonic Mean	UNKN 162	UNKN	1	
ST2HO(TRC,FIRST,NUMB) [proc]	Converts a simple STC coded item to OPCON code. TRC = Input value to be converted in STC 8. FIRST = First entry a table that the OPCON coded item is placed in, in Int. NUMB = Number of table entries that the converted value is to occupy, in Int.	COMP 33	UNKN	5	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
STORE(WHEN,BLANK'CK,EXCESS, FILE'CLD,CORE, DEGREE,LIMIT,ERROR) [proc]	Block "prints" lines, 23 lines per block and writes them double buffered. WHEN = Status item where ∅ = Normal call, neither the first nor last. 1 = First call to write the file. 2 = Force end-of-report. BLANK'CK = Bool item, where: 1 = Ignore blank lines ∅ = Process blank lines as good data EXCESS = Space needed for a logical data unit, in Int. FILE'CD = File code of output file, in Int. CORE = Table name containing data to be stored. DEGREE = Classification, in Int. ∅ = unclassified 1 = confidential 2 = secret 3 = top secret LIMIT = Lines per page, including header, but not classification and page-no lines, in Int. ERROR = File code of the file for listing I/O error messages, in Int.	N/A 328	UNKN	5	CK'STATS INTH1

	CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
TAN(UU)	[func]	Computes the tangent of an angle in Radians. TAN = Output value, in Flt Pt. UU = Input value, in Flt Pt Rad.	$\frac{.5\times10^{-11}}{14}$	3.4	2	COSIN SIN
TANH(UU)	[func]	Computes the hyperbolic tangent of a number. TANH = Output value, in Flt Pt. UU = Input value, in Flt Pt.	$\frac{10^{-11}}{15}$	1.8	2	FEXP
TICKS(TCST,TCNO,TCDS,TCLT)	[proc]	Draws any number of tick marks of any length, at any interval, beginning at any point along Y. TCST = Y value of first Tick in Flt Pt Inch. TCNO = Number of ticks, in Arit. TCDS = Dist between ticks, in Flt Pt Inch. TCLT = Tick length, in Flt Pt Inch.	N/A 68	UNKN	7	DRAW MOVE
TIME(CLOCK)	[proc]	Initiates a request to SSEC for a query of the real time clock. CLOCK = Real time by: YRMODAHRMNSC right justified, in Hol 16.	N/A 11	UNKN	5	
TPRELSE(FC,LNTH,UNIT, DATE1 = RELDT)	[proc]	Generates a release date and reel number for a given tape file code to be used by the operator. FC = File code of the tape to be saved, in Int. LNTH = Number of days, weeks, or months, in Int. UNIT = Units for LNTH: DA = days WK = weeks Mo = months DATE1 = Real time clock substitute, in Hol 6. RELDT = Release date as printed for the operator, in Hol 9. YRXMONXDA YR = year X = space MON = month DA = day	N/A UNKN	UNKN	5	REEL TIME TYPE

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
TYPE(DUM'MESS) [proc]	Outputs a 50 Hollerith item onto the typewriter, in FPS. DUM'MESS = An item containing the output message, in Hol 50.	N/A 135	UNKN	4	
TYPE'MSG(FC,CORE,WORDS, ER'FL) [proc]	Outputs up to 48 characters on the typewriter. FC = Typewriter file code, in Int. CORE = Table name containing the message to be outputted. WORDS = Number of words in the message in Int. ER'FL = File Code of a file to place I/O transmission errors on, in Int.	N/A 130	UNKN	5	CK'STATS
WRITE(DUM'FILE,DUM'TABL) [proc]	Transfers an entry from a serial table to an output file, in FPS. DUM'FILE = File description item, in Hol 50. See procedure write up for details. DUM'TABL = Table name of the serial table that the entry will be read from.	N/A 350	UNKN	4	FDES'ENT I'O TYPE
WSEG(DUM'FILE) [proc]	Writes an end of segment mark on a file, in FPS. DUM'FILE = File description item, in Hol 50. See procedure write up for details.	N/A 26	UNKN	4	FDES'ENT I'O RWRIT TYPE
ZZ(XX = BCDF) [proc]	Converts a Flt Pt Numb to an OPCON coded item 16 characters long. XX = Input value, in Flt Pt. BCDF = Output value, in Hol 16.	UNKN 247	UNKN		

Notes To The Quick Reference Index

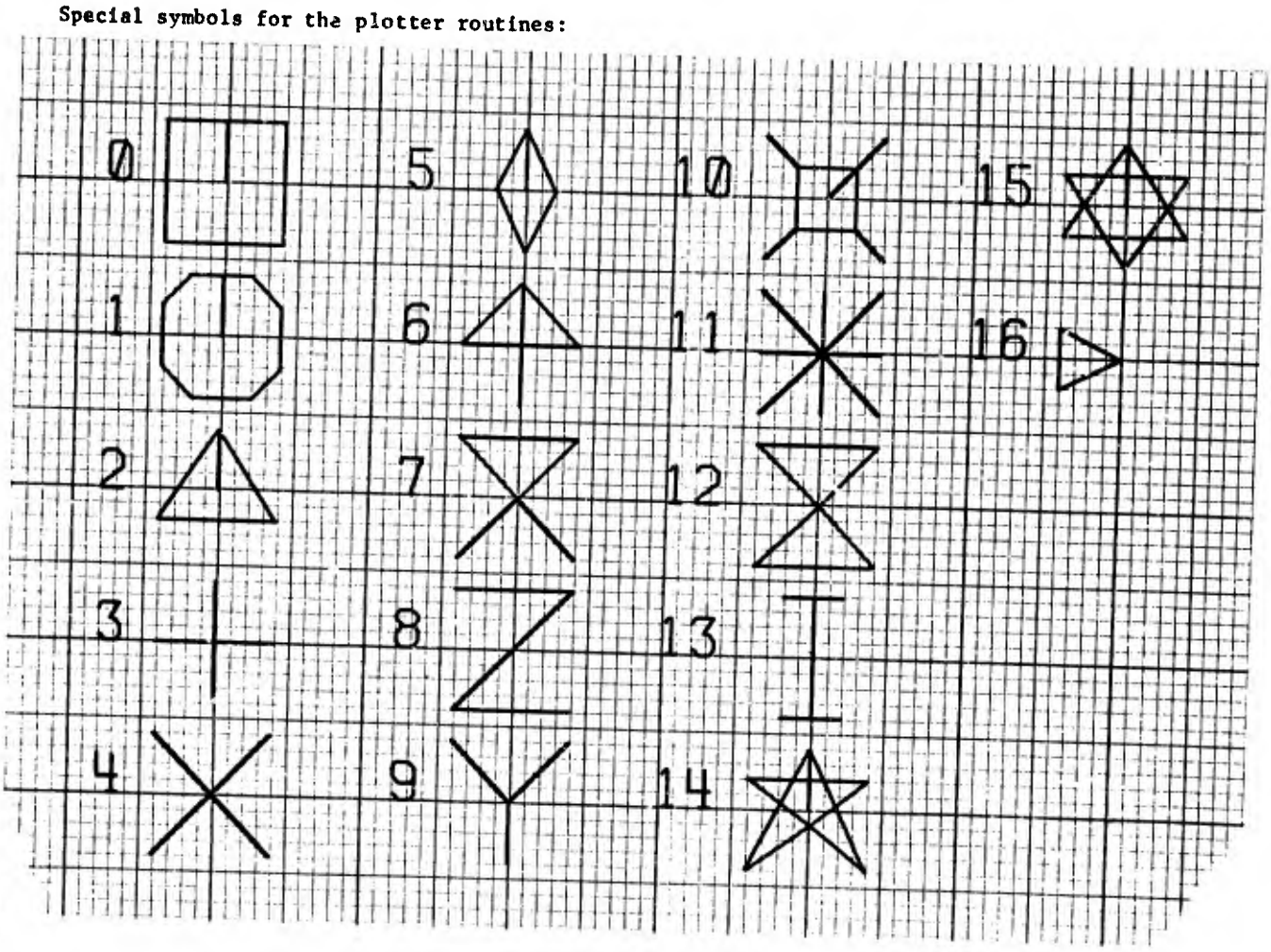
The heading call indicates the procedure name and the parameters as actually declared in the procedure. Under this, in brackets, is an indication as to whether this is a "procedure" or "function" call. In the description of the parameters, how that parameter is actually declared in the procedure is indicated.

SEC is the Section under which the complete documentation will be found.

The abbreviations used are listed below:

Arit	= Arithmetic	Int	= Integer
Bool	= Boolean	Lat	= Latitude
Coord	= Coordinate	Long	= Longitude
Comp	= Complete	Naut	= Nautical
Deg	= Degrees	Mi	= Miles
Dep	= Departure	N/A	= Not Applicable
Dest	= Destination	Proc	= Procedure Call
Flt	= Floating	Pt	= Point
FPS	= File Processing System	Rad	= Radians
Func	= Function Call	REAS	= Reason
Hol	= Hollerith	SOA	= Speed of Advance
Inch	= Inches	UNKN	= Unknown

Special symbols for the plotter routines:



SECTION III

LIBRARY

INTRODUCTION TO THE ARITHMETIC PROCEDURES

1. The procedures grouped in this category are:

   a. FEXP
   b. LOGNAT
   c. REMQUO
   d. SQRT
   e. STATS

2. All these procedures operate from FUNCTION calls.

3. No checks or tests are made on the input for validity. All input is assumed valid. There is no provision for abnormal terminations.

1. PURPOSE

FEXP is a function that raises the constant 'e' to a given power.

2. CALL FORMAT

XX = FEXP(NUM)

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

FEXP = the function name; declared as a floating point number representing the results of the operation e^{NUM}.

NUM = any valid floating point number.

3. LIMITATIONS AND ACCURACY

a. The routine is limited to the maximum size allowed by a floating point number at the output of the routine. This is because 'e' is raised to some value. Thus the approximate maximum size that NUM can be set to is 709.08956572. Any larger value will result in overflow in the output. Similarly there is a limitation as to how small an input value can be before the output becomes equal to zero. This is 1.0×10^{-10}.

4. PROCEDURE CHARACTERISTICS

a. Minimum time of operation is about 0.85 milliseconds. Time may be longer and is a function of how large the input item is. The larger the number, the longer the time required.

b. Storage - 61 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. MATHEMATICAL METHOD USED

a. If NUM is equal to a whole number, X, the value of 'e', (2.7182818285), is raised by repeated multiplication with itself, X number of times.

b. For fractional numbers, a modified Taylor Series is used.

c. For mixed numbers the above methods (a) and (b) are both used and the results added.

7. EXAMPLE

IF:

```
Item  XX  F  $
Item  NUM  F  P  2.0 $
```

AND:

```
XX = FEXP(NUM) $
```

THEN:

```
XX = 7.3890560989
```

1. PURPOSE

LOGNAT is a function that computes the natural (base e) logarithm of a number greater than zero.

2. CALL FORMAT

```
XX = LOGNAT(NUM)
```

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

LOGNAT = the function name; declared as a floating point number representing the natural log of NUM.

NUM = a valid floating point number greater than zero.

3. LIMITATIONS AND ACCURACY

a. The procedure checks to see that the input value (NUM) greater than zero. The procedure returns a value of 0.0 for any input less than or equal to zero.

4. PROCEDURE CHARACTERISTICS

a. Time of operation varies between 1 and 3 milliseconds.

b. Storage - 75 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. MATHEMATICAL METHOD USED

a. A truncated Taylor Series is evaluated, using Chebyshev Polynomials to reduce the number of terms required.

7. **EXAMPLE**

IF:

```
Item  XX   F   $
Item  NUM  F   P  1.5  $
```

AND:

THEN:

```
XX = LOGNAT(NUM) $

XX = 0.4054651081
```

1. PURPOSE

REMQUO is a procedure that performs integer division and produces a separate quotient and remainder.

2. CALL FORMAT

REMQUO(DND, DVS = QUO, REM)

WHERE:

REMQUO = the procedure name.

DND = a signed 48 bit integer to be divided by DVS.

DVS = a signed 48 bit integer by which DND is to be divided.

QUO = a signed 48 bit integer representing the integer result of DND/DVS.

REM = a signed 48 bit integer representing the integer result of DND - (QUO*DVS).

3. LIMITATIONS AND ACCURACY

a. No limitations.

b. Accuracy is limited only to the size of the machine word.

4. PROCEDURE CHARACTERISTICS

a. Minimum time for operation is in order of 150 microseconds.

b. Storage - 11 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. MATHEMATICAL METHOD USED

a. Integer multiply and integer divide.

7. EXAMPLE

IF:

```
Item  DND  I  48  S  P  3  $
Item  DVS  I  48  S  P  2  $
```

AND:

```
REMQUO(DND, DVS = QUO, REM) $
```

THEN:

```
QUO = 1
REM = 1
```

1. PURPOSE

SQRT is a function that computes the square root of any floating point number greater then zero.

2. CALL FORMAT

XX = SQRT(NUM)

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

SQRT = the function name; declared as a floating point number representing the square root of NUM.

NUM = any valid floating point number greater than zero.

3. LIMITATIONS AND ACCURACY

a. The routine does test the input value for less than or equal to zero. If the test condition is met, SQRT is set equal to zero and control returned to the main program. No differentiation is made between a zero input value and a less than zero input value.

4. PROCEDURE CHARACTERISTICS

a. The average time of operation is around 0.75 milliseconds.

b. Core storage requirement is for 58 machine words.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. MATHEMATICAL METHOD USED

a. The computation is performed by doing two iterations of the Newton's approximation.

7. EXAMPLE

IF:

Item XX F $
Item NUM F P 81.0

AND:

THEN: XX = SORT(CUM) <

XX = 0.0

1. PURPOSE

STATS is a multiple input function that is capable of determining the arithmetic mean, geometric mean, harmonic mean, mode, median, variance, and standard deviation of a set of values assumed normal in distribution.

2. CALL FORMAT

XX = STATS(CORE, SKIP, WDNT, TYPE)

WHERE:

XX = a program declared item of the user to which the function output is assigned.

STATS = the function name: declared as a floating point item and containing the computational result of the procedure.

CORE = the name of a user defined table in the main program which contains the floating point values that the procedure is to operate on.

SKIP = a 48 bit signed integer item representing the number of words per entry as specified for table CORE.

WDNT = a 48 bit signed integer item specifying the relative word-in-entry of the values to be operated on in table CORE.

TYPE = a status item that specifies the particular operation that the user desires to have done as indicated below:

0 = Calculate the Mean
1 = Find the Mode
2 = Find the Median
3 = Calculate the Standard Deviation
4 = Find the Variance
5 = Calculate the Geometric Mean
6 = Calculate the Harmonic Mean

3. LIMITATIONS AND ACCURACY

a. The NENT (of CORE) must reflect the true number of meaningful entries and must have a value greater than 1.

3. LIMITATIONS AND ACCURACY (cont)

b. The Geometric Mean is meaningless if the calling table (CORE) contains negative values. No check is made to see that the input values are positive when calculation of the Geometric Mean is requested.

c. The Calling table (CORE) may or may not be a programmer allocated table. Similarly, it may be declared as variable or rigid in length.

d. The Calling table (CORE) must be rigid-entry-length table.

e. The Calling table (CORE) must be defined as a Serial table, or a parallel table having one word per entry.

f. All values to be operated on in the table must be in a floating point format.

g. No checks are made for illegal or impossible data.

4. PROCEDURE CHARACTERISTICS

a. Operating time is unknown.

b. Storage - 162 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. MATHEMATICAL METHOD USED

a. Standard statistical formulae are used for all but the Mode and Median. Both of the latter adhere to their customary statistical meanings and are found by searching - i.e., they are values in the set contained in the calling table (CORE).

7. EXAMPLE

IF:

```
TABLE CORE  R  10  S  2  $
  BEGIN
    Item  VALU1  F  Ø  Ø  $
    Item  VALU2  F  1  Ø  $
  END
Item  SKIP  I  48  S  P  2  $
Item  WDNT  I  48  S  P  2  $
Item  TYPE  I  48  S  P  5  $
Item  XX    F  $
```

AND:

THEN:

```
XX = STATS(CORE, SKIP, WDNT, TYPE) $
```

XX will be set equal to the floating point value representing the Geometric Mean of the 10 Items labeled VALU2.

INTRODUCTION TO THE TRIGONOMETRIC PROCEDURES

1. The procedures grouped in this category are:

   a. ARCOS
   b. ARCSIN
   c. ARCTAN
   d. COSH
   e. COSIN
   f. COTAN
   g. SIN
   h. SINH
   i. TAN
   j. TANH

2. All these procedures operate from FUNCTION calls.

3. With the exception of the hypebolic functions, all the trigonometric procedures require values in radians as an input or output parameter. The conversion procedures, RADIANS, will convert degrees into radians; DEGREES, will convert radians into degrees, if so required.

4. All input parameters are assumed to be valid and no checks are made for invalid parameters.

ARCCOS

1. PURPOSE

ARCCOS is a function that computes the arccosine of a number between minus one and plus one inclusive.

2. CALL FORMAT

XX = ARCCOS(NUM)

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

ARCCOS = a floating point number, in radians, representing the arccosine of NUM.

NUM = a floating point number between plus one and minus one inclusive.

3. LIMITATIONS AND ACCURACY

a. If the absolute value of the input parameter (NUM) is greater than one, a value of zero will be returned by the function.

b. The accuracy for angles up to 89 degrees is 10^{-7}, for angles from 89 to 89 1/2 degrees, the accuracy is 10^{-5}. For angles from 89 1/2 to 90 degrees, the accuracy is less than 10^{-5}.

4. PROCEDURE CHARACTERISTICS

a. Time of operation varies between 1 and 3 milliseconds.

b. Storage - 15 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. ARCSIN

6. MATHEMATICAL METHOD USED

a. The arccos x = 2 - arcsin x.

7. EXAMPLE

```
IF:
     Item  XX  F  $
     Item  NUM  F  P  0.5  $

AND:
                          XX = ARCCOS(NUM) $
THEN:
                          XX = 1.047197550
```

ARCSIN

1. PURPOSE

ARCSIN is a function that computes the arcsine of a number between minus one and plus one inclusive.

2. CALL FORMAT

XX = ARCSIN(NUM)

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

ARCSIN = a floating point number between plus one and minus one inclusive.

NUM = a floating point number between plus one and minus one inclusive.

3. LIMITATIONS AND ACCURACY

a. If the absolute value of the input parameter (NUM) is greater than one, a value of zero will be returned by the function.

b. The accuracy for angles up to 89 degrees is 10^{-7}, for angles from 89 to 89 1/2 degrees, the accuracy is 10^{-5}. For angles from 89 1/2 to 90 degrees inclusive the accuracy is less than 10^{-5}.

4. PROCEDURE CHARACTERISTICS

a. Time of operation varies between 1 and 3 milliseconds.

b. Storage - 89 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. MATHEMATICAL METHOD USED

a. If the input parameter (NUM) is equal to zero, plus one, or minus one, the output parameter is set equal to a constant.

b. For value between the absolute value of one and zero a truncated Taylor series is evaluated, using the Chebyshev polynomials to reduce the number of terms required.

7. EXAMPLE

IF:

```
Item  XX  F  $
Item  NUM  F  P  0.5  $
```

ADD:

```
XX = ARCSIN(NUM)
```

THEN:

```
XX = 0.5235087750
```

1. PURPOSE

ARCTAN is a function that computes the arctangent of any number.

2. CALL FORMAT

XX = ARCTAN(NUM)

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

ARCTAN = a floating point number, in radians, representing the arctangent of NUM.

NUM = a valid floating point number.

3. LIMITATIONS AND ACCURACY

a. The accuracy for angles up to 88° 51' is 10^{-7}, for angles from 88° 51' to 89° 11', the accuracy is 10^{-5}. For angles from 89° 11' to 90° the accuracy is less than 10^{-5}.

4. PROCEDURE CHARACTERISTICS

a. The average time of operation is about 1.6 milliseconds. The time may be greater, depending upon the size of the input parameter.

b. Storage - 12 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. ARCSIN, SQRT

6. MATHEMATICAL METHOD USED

a. Arctan x = arcsin $[x(x^2 + 1)^{1/2}]$.

7. **EXAMPLE**

IF:

```
Item  XX  F  $
Item  NUM  F  P  0.95  $
```

AND:

```
                    XX = ARCTAN(NUM) $
```

THEN:

```
                    XX = 0.7597627548
```

COSH

1. PURPOSE

COSH is a function that computes the hyperbolic cosine of a number.

2. CALL FORMAT

```
XX = COSH(NUM)
```

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

COSH = a floating point number representing the hyperbolic cosine of NUM.

NUM = a valid floating point number.

3. LIMITATIONS AND ACCURACY

a. This routine is limited to the maximum size allowed by a floating point number at the output of the FEXP routine. This is because the exponetial, 'e' is raised to some value. The approximate maximum size that NUM can be set to is 709.08956572. Any larger value will result in an overflow in the output. Similarly there is a limitation as to how small an input value can be before the output (FEXP) becomes equal to zero. This is 1.0×10^{-10}.

4. PROCEDURE CHARACTERISTICS

a. The average time of operation is about 1,8 milliseconds.

b. Storage - 11 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. FEXP

6. MATHEMATICAL METHOD USED

a. $\cosh x = (e^{x} + e^{-x})/2$.

7. EXAMPLE

IF:

```
Item  XX   F  $
Item  NUM  F  P  2.6  $
```

AND:

```
                          XX = COSH(NUM) $
```

THEN:

```
                          XX = 6.769005807
```

COSIN

1. PURPOSE

COSIN is a function that computes an approximation of the trigometric cosine.

2. CALL FORMAT

XX = COSIN(RAD)

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

COSIN = a floating poing number representing the cosine of RAD.

RAD = a floating point number representing the angle in radians.

3. LIMITATIONS AND ACCURACY

a. There is no limitation as to the maximum size allowed for an input parameter (RAD) except as dicated by the largest size attainable in a floating point word.

b. If the input parameter, (RAD), is equal to or greater than 0, but less than $0.9999999999 \times 10^{12}$, a value of 1.0 is returned.

c. If the input parameter, (RAD), is greater than 3.14159265358900, but less than 3.14159265358999, a value of -1.0 is returned.

d. If the input parameter, (RAD), is greater than 1.57079632679400, but less than 1.57079632679499; OR, if greater than 4.71238898038400, but less than 4.71238898038499 a value of 0 is returned.

e. The maximum error is 0.5×10^{-10} of any angle.

4. PROCEDURE CHARACTERISTICS.

a. The average time of operation is about 2.0 milliseconds.

b. Storage - 46 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

   a. SIN

6. MATHEMATICAL METHOD USED

   a. All input values, positive and negative, are reduced to a range lying between 0 and 2π before being evaluated.

   b. If the input parameter, (RAD), falls within the area specified in the limitations above, a constant equal to the value indicated is returned.

   c. For input value other than indicated above:
   cosin x = sin $(\pi/2-x)$.

7. EXAMPLE

   IF:

```
Item  XX  F  $
Item  RAD  F  P  1.250  $
```

   AND:

   THEN:

```
XX = COSIN(RAD) $

XX = 0.3153223623
```

COTAN

1. PURPOSE

COTAN is a function that computes an approximation of the trigometric cotangent.

2. CALL FORMAT

```
XX = COTAN(RAD)
```

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

COTAN = a floating point number representing the cotangent of RAD.

RAD = a valid floating point number representing the angle in radians.

3. LIMITATIONS AND ACCURACY

a. The output parameter of the procedure will be set equal to zero when the cotangent of the input parameter (RAD), is undefined.

4. PROCEDURE CHARACTERISTICS

a. The average time of operation is 3.5 milliseconds.

b. Storage - 11 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. TAN

6. MATHEMATICAL METHOD USED

a. Cotan x = 1/tan x

7. EXAMPLE

IF:

```
Item  XX  F  $
Item  RAD  F  P  0.65  $
```

ADD:

```
XX = COTAN(RAD) $
```

THEN:

```
XX = 1.31543569
```

1. PURPOSE

SIN is a function that computes an approximatior of the trigometric sine.

2. CALL FORMAT

XX = SIN(RAD)

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

SIN = a floating point number representing the sine of RAD.

RAD = a valid floating point number representing the angle in radians.

3. LIMITATIONS AND ACCURACY

a. There is no limitation as to the maximum size allowed for an input parameter (RAD), except as dictated by the largest size attainable in a floating point word.

b. If the input parameter, (RAD), is greater than or equal to zero and less than $0.9999999999 \times 10^{-12}$; OR, if greater than 3.14159265358900, but less than 3.14159265358999, a value of 0 is returned.

c. If the input parameter, (RAD), is greater than 1.57079632679400, but less than 1.57079632679499, a value of 1.0 is returned.

d. If the input parameter, (RAD), is greater than 4.71238898038400, but less than 4.71238898038499, a value of -1.0 is returned.

e. The maximum error is 0.5×10^{-10} at 89.9°. At 20° the error is about 0.8×10^{-11}; at 0.5°, 0.3×10^{-12}: and at 0.1°, 0.9×10^{-13}.

4. PROCEDURE CHARACTERISTICS

a. The average time of operation is about 1.25 milliseconds.

b. Storage - 50 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. MATHEMATICAL METHOD USED

a. All input values, positive and negative, are reduced to a range lying between 0 and 2 before being evaluated.

b. If the input parameter, (RAD), falls within the area specified in the limitations above, a constant equal to the value indicated is returned.

c. For input values other than indicated above, the Chebyshev-Taylor Approximating Polynomial is used.

7. EXAMPLE

IF:

```
Item  XX  F  $
Item  RAD  F  P  1.100  $
```

AND:

THEN:

```
XX = SIN(RAD) $

XX = 0.8912073600
```

SINH

1. PURPOSE

SINH is a function that computes the hyperbolic sine of a number.

2. CALL FORMAT

```
XX = SINH(NUM)
```

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

SINH = a floating point number representing the hyperbolic cosine of NUM.

NUM = a valid floating point number.

3. LIMITATIONS AND ACCURACY

a. This routine is limited to the maximum size allowed by a floating point number at the output of the FEXP routine. This is because the exponential, 'e' is raised to some value. The approximate maximum size that NUM can be set to is 709.08956572. Any larger value will result in an overflow in the output. Similarly, there is a limitation as to how small an input value can be before the output (FEXP) becomes equal to zero. This is 1.0×10^{-10}.

4. PROCEDURE CHARACTERISTICS

a. The average time of operation is about 1.8 milliseconds.

b. Storage - 12 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. FEXP

6. MATHEMATICAL METHOD USED

a. $\sinh x = (e^x - e^{-x})/2$

7. EXAMPLE

IF:

```
Item  XX  F  $
Item  NUM  F  P  2.6  $
```

AND:

```
                    XX = SINH(NUM) $
```

THEN:

```
                    XX = 6.69473228
```

TAN

1. PURPOSE

TAN is a function that computes an approximation of the trigometric tangent.

2. CALL FORMAT

XX = TAN(RAD)

WHERE:

XX = the program declared item of the user which is assigned the evaluated value.

SIN = a floating point number representing the tangent of RAD.

RAD = a valid floating point number representing the angle in radians.

3. LIMITATIONS AND ACCURACY

a. There is no limitation as to the maximum size allowed for an input parameter (RAD), except as dictated by the largest size attainable in a floating point word.

b. If the input parameter, (RAD), is equal to zero, a value of zero is returned for the output.

c. The maximum error is 0.5×10^{-11}.

4. PROCEDURE CHARACTERISTICS

a. The average time of operation is about 3.4 milliseconds.

b. Storage - 14 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. SIN, COSIN

6. MATHEMATICAL METHOD USED

a. tan x = sin x/cos x.

7. EXAMPLE

IF:

```
Item  XX  F  $
Item  RAD  F  P  0.55  $
```

AND:

```
                        XX = TAN(RAD) $
```

THEN:

```
                        XX = 0.61310521
```

TANH

1. PURPOSE

TANH is a function that computes the hyperbolic tangent of a number.

2. CALL FORMAT

XX = TANH(NUM)

WHERE:

XX = the program declared item of the user to which is assigned the evaluated value.

TANH = a floating point number representing the hyperbolic tangent of NUM.

NUM = a valid floating point number.

3. LIMITATIONS AND ACCURACY

a. This routine is limited to the maximum size allowed by a floating point number at the output of the FEXP routine. This is because the exponential, 'e' is raised to some value. The approximate maximum size that NUM can be set to is 709.08956572. Any larger value will result in an overflow condition in the routine. Similarily, there is a limitation as to how small an input value can be before the output FEXP becomes equal to zero. This is 1.0×10^{-10}.

4. PROCEDURE CHARACTERISTICS

a. The average time of operation is about 1,8 milliseconds.

b. Storage - 15 machine words are required.

c. This is a pure JOVIAL program.

5. OTHER PROCEDURES CALLED

a. FEXP

6. MATHEMATICAL METHOD USED

a. $\tanh x = (e^{x} - e^{-e})/(e^{x} + e^{-x})$

7. EXAMPLE

IF:

```
Item  XX  F  $
Item  NUM  F  P  1.65  $
```

AND:

```
                          XX = TANH(NUM) $
```

THEN:

```
                          XX = 0.92885762
```

Introduction to the Conversion Procedures

1. The procedures grouped in this category are:

a. ALPHCON	i. HL2HL	q. OC2HO
b. DEGREES	j. HL2HO	r. OPCTOB
c. FBINRY	k. HL2IN	s. PCL
d. FL2HO	l. HL2OC	t. RADIANS
e. FOPCON	m. HL2ST	u. ST2HO
f. FX2HO	n. IN2HO	v. TYPE
g. HL2FL	o. INTHI	w. TYPE'MSG
h. HL2FX	p. NORM360	x. ZZ

2. This section consists primarily of a series of procedures of an XX2XX name format for item format conversions. This general category can be broken down into two groups; XX2HO and HL2XX. Each series make use of a different, centrally defined table that is declared by the user in his main program.

The HL2XX series of procedures are designed to take OPCON coded data, generally representing card columns, and located in an unpacked, one character per word image table; and convert it to a specified simple item. Similarly, the XX2HO is a series of procedures designed to convert various item types into OPCON code and store these characters in an unpacked image table, one character per word, and generally representing a print line.

These procedures were designed primarily to accomplish the same results as the JOVIAL ENCODE and DECODE statements. The HL2XX series will take an 80 column card and decode the various items on that card to simple items according to the particular procedure used. The reverse of this is accomplished by the XX2HO series which takes various items and converts them into OPCON and places them in a 120 entry table for printing.

Each byte is placed in the low order of each word or entry. This also makes the table structure convenient for testing each byte or a particular set or combination of bytes. The location of the items in the table may be governed by the user. The table format used in this group of procedures is the same as required in the CSCAN procedure.

Each procedure in one series has a complimentary procedure in the other series as indicated below:

FL2HO - Converts a simple floating point item to OPCON code placing it in the unpacked table.

HL2FL - Converts the OPCON code in the unpacked table to a simple floating point item.

FX2HO - Converts a simple arithmetic item to OPCON code placing it in the unpacked table.
HL2FX - Converts the OPCON code in the unpacked table to a simple arithmetic item.

HL2HO - Takes an OPCON coded simple item and unpacks it, placing it in the unpacked table.

IN2HO - Converts a simple decimal integer item to OPCON code placing it in the unpacked table.
HL2IN - Converts the OPCON code in the unpacked table to a simple decimal integer item.

OC2HO - Converts a simple octal integer item to OPCON code placing it in the unpacked table.
HL2OC - Converts the OPCON code in the unpacked table to a simple octal integer item.

ST2HO - Converts a Standard Transmission coded simple item to OPCON code placing it in the unpacked table.
HL2ST - Converts the OPCON code in the unpacked table to a simple Standard Transmission coded item.

3. Data format conversions available not making use of a table, but that are word-to-word conversion are:

a. FBINRY - Converts an 8 byte OPCON coded item to an integer item.
b. FOPCON - Converts an integer item to an 8 byte OPCON coded item.
c. INTHI - Converts an unsigned integer item to an 8 byte OPCON coded item.
d. OPCTOB - Converts an 8 byte OPCON coded item to a floating point item.
e. ZZ - Converts a floating point item to an OPCON coded item 16 bytes long.

4. The remaining procedures provide various conversions of a miscellaneous nature.

ALPHCON

1. PURPOSE

ALPHCON is a multiple input function that converts IBM 7090 code to OPCON code, or OPCON code to IBM 7090 code.

2. CALL FORMAT

XX = ALPHCON(ARGUMENT, WHICHWAY)

WHERE:

XX = a program declared item of the user to which the function output is assigned.

ALPHCON = the function name; declared as an 8 character hollerith item, containing the converted results of the routine.

ARGUMENT = the input value to be converted defined as an 8 character hollerith item.

WHICHWAY = a status item, indicating whether OPCON code is to be converted to IBM 7090 code or IBM 7090 code is to be converted to OPCON code as indicated below:

0 = OPCON to IBM 7090
1 = IBM 7090 to OPCON

3. LIMITATIONS AND ACCURACY

a. One 8-byte word (item) is converted at a time.

b. All characters are converted, regardless of legality. If illegal characters are present, blanks are generated.

c. This routine is very system dependent and is not transferable, unless the table containing the conversion code is changed to match the system being used.

4. PROCEDURE CHARACTERISTICS

a. Operating time is unknown, but it will be the same for all input parameters.

b. Storage - 43 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. REMQUO

6. PROGRAMMING TECHNIQUE USED

a. Direct indexing is used to avoid table look-up and minimize conversion time and core requirements.

7. EXAMPLE

IF:

```
Item  XX  H  8  $
Item  ARGUMENT  H  8  P  8H(ACEGI135)  $
Item  WHICHWAY  I  47  U  P  0  $
```

AND:

THEN:

```
XX = ALPHCON(ARGUMENT, WHICHWAY)  $
```

XX will print as: 13579

NOTE: Item ARGUMENT, ACEGI135, will be represented in OPCON as: 0610121416616365. The output item, XX, after the conversion takes place will contain the IBM 7090 code representation, 6163656771010305. This is in turn interpreted when outputed by this system as 13579 UPPERCASE TAB SPACE, the latter three being interpreted as blanks on the printer.

DEGREES

1. PURPOSE

DEGREES is a function that converts floating point radians into angular floating point degrees.

2. CALL FORMAT

```
XX = DEGREES(RAD)
```

WHERE:

XX = the program declared item of the user to which the function output is assigned.

DEGREES = the function name; declared as a floating point number containing the converted value.

RAD = a floating point number, containing the input value in radians, to be converted into degrees.

3. LIMITATIONS AND ACCURACY

a. There is no limitation on the size of the input parameter, RAD, or its format other than it be compatible with a floating point format. The output value is not broken down into minutes and seconds, but occurs as whole and fractional degrees.

b. No checks are made on the input parameters. All input is assumed to be valid.

c. The accuracy of the conversion should be greater than ten decimal places.

4. PROCEDURE CHARACTERISTICS

a. The time of operation is unknown, but should be the same for all input parameters.

b. Storage - 3 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. MATHEMATICAL METHOD USED

a. The input parameter, RAD, is multiplied by the appropriate constant to convert it into degrees.

7. EXAMPLE

IF:

```
Item  XX   F  $
Item  RAD  F  P  100.0  $
```

AND:

```
XX = DEGREES(RAD)  $
```

THEN:

```
XX = 5729.57795131
```

FBINRY

1. PURPOSE

FBINRY is a function that converts numeric values of OPCON code to its integer equivalent one word at a time.

2. CALL FORMAT

```
INTVALU = FBINRY(HOLVALU)
```

WHERE:

INTVALU	=	a program declared item of the user to which the function output is assigned.
FBINRY	=	the function name; and is declared as a 48 bit signed arithmetic item containing no fractional bits.
HOLVALU	=	the input value that the user wishes to have converted, declared as an 8 character OPCON item.

3. LIMITATIONS AND ACCURACY

a. Certain assumptions are made by this procedure as to what the input parameters may be. The only check made is for imbedded non-numeric characters other than plus signs, minus signs, and blanks. Leading plus signs, minus signs, and blanks are legal. If a non-numeric character other than a plus sign, minus sign, or blank occurs, the output parameter will be set equal to 70000_8 which is printed as -4095. If an incorrect combination of blanks, plus signs, or minus signs occur, an erroneous conversion will occur.

b. Leading minus signs are carried through in the conversion. Leading plus signs are converted to blanks. Values that are left justified on input will be right justified on output. Thus trailing blanks, but not imbedded blanks, are acceptable.

4. PROCEDURE CHARACTERISTICS

a. The time of operation is unknown, but will be the same for each input parameter.

b. Storage - 42 machine words are required.

c. This is not a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. 60 is subtracted from each byte of the word and the remaining digits repacked to form the converted value.

7. EXAMPLE

IF:

```
Item  INTVALU  I  48  S  $
Item  HOLVALU  H  8   P  8H(-3549    )  $
```

AND:

```
                          INTVALU  =  FBINRY(HOLVALU)  $
```

THEN:

```
                          INTVALU  =  -3549
```

Note:

In memory the preset item HOLVALU would look like:

4163656471050505

This is the general format of the value that is being converted to an integer or binary format from OPCON. After being converted, the memory location for INTVALU would look like:

7777777777771042

which is interpreted as -6735_8 or equal to -3549_{10}.

1. PURPOSE

FL2HO is a procedure that converts a floating point item to OPCON code and places it in a main program defined table, one byte per word.

2. CALL FORMAT

FL2HO(FLOAT, FCOL, NUMB)

WHERE:

FL2HO = the procedure name.

FLOAT = an item defined as a floating point number that the user wishes to convert to OPCON.

FCOL = an integer value specifying which column in the range of 0 to 120 on a printed page that the first character is to go into. This is also the number of the entry in a user defined table that the first character is placed in.

NUMB = an integer value specifying the number of places that converted word is to occupy. Seven places must be reserved for the sign, decimal point, E, and a three-digit signed exponent. NUMB will thus be equal to seven plus the number of fractional positions desired to be maintained in the converted value.

3. LIMITATIONS AND ACCURACY

a. The procedure checks the input parameter, FLOAT, to see that it is a valid floating point number. If not, the input parameter is treated as an octal integer and converted to OPCON.

b. This procedure does not round the converted value. Total accuracy is thus limited by this only factor. For example, the value 0.2, when converted will be 0.19999.... .

c. The procedure does not clear and/or reset the user defined table described below. The user must decide to what value he wants the table set to before the conversion is made. If he presets the table to zeros and then tries to print it, the converted value is surrounded by %. For such a case it should be preset to OPCON blanks. The example shows how the entries containing only zeros will come out as %.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure. In any case this would be a variable because the time would be a function of the input parameters supplied.

b. Storage - 80 machine words are required.

c. This is a pure JOVIAL program.

d. This procedure requires a user defined table in which to place the converted output values. This table may be defined according to any name the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users need, however 120 is usually chosen to correspond to the 120 print columns available on the printer.

The item declared in this table must have the name CPOSN. It also must be defined as hollerith containing one byte, starting in the zeroth entry of the table. The first bit should be 42 to place it in the lowest position of the word, but this may be changed at the user's discretion.

A sample table declaration is as follows:

```
TABLE  COLUMN  R  120  S  1  $
BEGIN
  ITEM  CPOSN  H  1  Ø  42  $
END
```

5. OTHER PROCEDURES CALLED

a. IN2HO, OC2HO

6. PROGRAMMING TECHNIQUE USED

a. The input parameter, (FLOAT) is reduced to a fractional value lying between 0.1 and 1.0. The number of times it is reduced, (done by dividing the input parameter by ten) becomes the exponent for the converted value. The fractional portion is converted by taking each digit, adding it to 60_8 and placing it in the output table.

7. EXAMPLE

IF:

```
Table COLUMN  R  120  S  1  $
Begin
  Item  CPOSN  H  1  0  42  $
End
Item  FLOAT F P  235.65  $
Item  FCOL  I  47  U  P  2  $
Item  NUMB  I  47  U  P  12  $
```

AND:

```
                                FL2HO(FLOAT, FCOL, NUMB)  $
```

THEN:

Table COLUMN would look as follows in core:

```
0000000000000170      =   nent word contains  120
0000000000000000      =   0 entry contains    %
0000000000000000      =   1 entry contains    %
0000000000000005      =   2 entry contains    blank
0000000000000075      =   3 entry contains    .
0000000000000062      =   4 entry contains    2
0000000000000063      =   5 entry contains    3
0000000000000065      =   6 entry contains    5
0000000000000066      =   7 entry contains    6
0000000000000065      =   8 entry contains    5
0000000000000012      =   9 entry contains    E
0000000000000041      =  10 entry contains    -
0000000000000005      =  11 entry contains    blank
0000000000000005      =  12 entry contains    blank
0000000000000063      =  13 entry contains    3
0000000000000000      =  14 entry contains    %
        .                       .
        .                       .
        .                       .
0000000000000000      = 119 entry contains    %
```

1. PURPOSE

FOPCON is a function that converts an integer item into an 8 byte OPCON item.

2. CALL FORMAT

```
HOL = FOPCON(INT)
```

WHERE:

- HOL = a program declared item of the user to which the OPCON coded output of the function is assigned.
- FOPCON = the function name; declared as an 8 character hollerith item and containing the converted output value.
- INT = the input value to be converted, declared as an integer or arithmetic item containing no fractional bits.

3. LIMITATIONS AND ACCURACY

a. One word is converted at a time.

b. If the input parameter, (INT), is greater than 99,999,999 or less than -9,999,999 the output parameter will be set to MMMMMMMM (8 M's). This prevents an erroneous conversion of an integer too large to fit in an 8 byte hollerith item.

c. The conversion should be completely accurate since no rounding is involved.

4. PROCEDURE CHARACTERISTICS

a. Operating time is unknown, but will be the same for all input parameters.

b. Storage - 77 machine words are required.

c. This is not a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. All input parameters are checked to see that they lie in the range of 99,999,999 to -9,999,999. If not, the output is returned with an error indicator of 8 M's.

b. The input parameter is then unpacked and repacked by adding 60 to each digit to build a new word.

7. EXAMPLE

IF:

```
Item  HOL  H  8   $
Item  INT  A  48  S  P  345  $
```

AND: HOL = FOPCON(INT) $

THEN: HOL = 345 when printed

NOTE:

In core, item INT will appear as 0000000000000531 which is 345 in octal. Similarily, item HOL after the conversion will contain 0505050505656361. This will then be printed as 345.

Had the input parameter been equal to 453506894, item HOL would be set equal to 2222222222222222 or 8 M's, due to the oversized input value.

1. PURPOSE

FX2HO is a procedure that converts an arithmetic item to OPCON code, placing it in a main program defined table, one byte per word.

2. CALL FORMAT

FX2HO(FIX, FRA, FCOL, NUMB, NOPL)

WHERE:

FX2HO = the procedure name.

FIX = an input parameter defined as a 48 bit signed integer item and representing the integer portion of the arithmetic item being converted.

FRA = the input parameter representing the fractional portion of the arithmetic item to be converted. Defined as arithmetic with 48 bits, 43 of which are fractional.

FCOL = an integer value specifying which column in the range of 0 to 120 on a printed page that the first character is to go into. This is also the number of the entry in a user defined table that the first character is placed in.

NUMB = an integer value specifying the number of places that the converted word is to occupy. Two places should be considered for the sign and decimal point.

NOPL = an integer value indicating the number of places that the user wishes the fractional portion of the input value to be carried to in the conversion.

3. LIMITATIONS AND ACCURACY

a. There are no checks made on the input parameters and they are assumed to be valid as supplied.

b. The converted value will be rounded if it occupies less space then the original input value. Leading zeros will be suppressed and the first column specified by the input parameter FCOL will contain a minus sign or be blank depending on the isgn of the value being converted.

c. The procedure does not clear and/or reset the user defined table described below. The user must decide to what value he wants the table set to before the conversion is made. If he presets the table to zeros and then tries to print it, the converted value is going to be surrounded by '%'. For such a case it should be preset to OPCON blanks. The example shows how the entries containing only zeros will come out as %'s.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure. It will however be variable, depending on the input parameters supplied.

b. Storage - 50 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table, in which, is placed the converted output values. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users need, however 120 is usually chosen to correspond to the 120 print columns available on the printer.

The item declared in this table must have the name CPOSN. It also must be defined as hollerith, containing one byte, and starting in the zeroth entry of the table. The first bit should be 42 to place it in the lowest position of the word, but this may be changed at the user's discretion.

A sample table declaration is as follows:

```
TABLE  COLUMN  R  120  S  1  $
BEGIN
  ITEM  CPOSN  H  1  Ø  42  $
END
```

5. OTHER PROCEDURES CALLED

a. IN2HO

6. PROGRAMMING TECHNIQUE USED

a. Procedure IN2HO is used to convert the integer portion of the arithmetic item being converted. The fractional portion is converted by taking each digit and adding it to 60 to produce the new OPCON coded character.

7. EXAMPLE

IF:

```
Table COLUMN  R  120  S  1  $
Begin
  Item  CPOSN  H  1  0  42  $
End
Item  FIX  I  48  S  P  235
Item  FRA  A  48  S  43  P  .65  $
Item  FCOL  I  47  U  P  2  $
Item  NUMB  I  47  U  P  8  $
Item  NOPL  I  47  U  P  1  $
```

AND:

```
                                FX2HO(FIX, FRA, FCOL, NUMB, NOPL) $
```

THEN:

Table COLUMN would look as follows in core:

```
0000000000000170      =  nent word contains   120
0000000000000000      =  0 entry contains     %
0000000000000000      =  1 entry contains     %
0000000000000005      =  2 entry contains     blank
00000 )0000000062     =  3 entry contains     2
0000000000000063      =  4 entry contains     3
0000000000000065      =  5 entry contains     5
0000000000000075      =  6 entry contains     .
0000000000000066      =  7 entry contains     6
000000000000000       =  8 entry contains     %
        .                       .
        .                       .
        .                       .
0000000000000000      = 120 entry contains    %
```

1. PURPOSE

HL2FL is a procedure that converts OPCON coded data to a floating point item from a main program defined table containing one byte per word.

2. CALL FORMAT

HL2FL(FCOL, NUMB = FLOAT)

WHERE:

HL2FL = the procedure name.

FCOL = an integer value specifying the column in a range of 1 to 80 on a card. This is also the number of the entry in a user defined table that the first character of the word is found in.

NUMB = an integer value, specifying the maximum number of characters or bytes to be considered in the conversion; i.e. how many characters of the OPCON item are to be converted into a floating point format.

FLOAT = an item defined as floating point and containing the converted output value from the routine.

3. LIMITATIONS AND ACCURACY

a. Leading blanks are acceptable, but trailing blanks are not.

b. The input parameter may contain either the normal or exponential format for a floating point number.

Example:

345.56 or .34556E + 1Ø

c. The procedure does not clear and/or reset the user defined table described below. The user must clear out this table each time before the procedure is called. This is to insure against possible errors that may result from picking up undesired characters from a previous conversion.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure. In any case this would be variable, as it would be a function of the size of the input parameter.

b. Storage - 116 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table from which is taken the unpacked OPCON items for repacking into a floating point word. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users requirements, however 80 is usually chosen for an 80 column card.

The item declared in this table must have the name COLUM, and declared as a one byte, hollerith item, starting in the zeroth entry of the table. The first bit should be 42 to place the byte in the lowest position of the word.

A sample table declaration is as follows:

```
TABLE  CITU  R  80  S  1  $
BEGIN
  ITEM  COLUM  H  1  Ø  42  $
END
```

5. OTHER PROCEDURES CALLED

a. HL2IN

6. PROGRAMMING TECHNIQUE USED

a. The whole and fractional portion of the input parameter is converted by means of the HL2IN procedure and then repacked to form a floating point word.

7. EXAMPLE

IF:

```
Table  CITU  R  80  S  1  $
Begin
  Item  COLUMN  H  1  0  42  $
End
Item  FLOAT  F  $
Item  FCOL  I  47  U  P  2  $
Item  NUMB  I  47  U  P  12  $
```

AND:

THEN: HL2FL(FCOL, NUMB = FLOAT) $

FLOAT = 235.65

NOTE:

Table CITU would look as follows in core:

```
0000000000000121      =  nent word contains   80
0000000000000000      =  0 entry contains     %
0000000000000000      =  1 entry contains     %
0000000000000005      =  2 entry contains     blank
0000000000000075      =  3 entry contains     .
0000000000000062      =  4 entry contains     2
0000000000000063      =  5 entry contains     3
0000000000000065      =  6 entry contains     5
0000000000000066      =  7 entry contains     6
0000000000000065      =  8 entry contains     5
0000000000000012      =  9 entry contains     E
0000000000000042      = 10 entry contains     +
0000000000000005      = 11 entry contains     blank
0000000000000005      = 12 entry contains     blank
0000000000000063      = 13 entry contains     3
0000000000000000      = 14 entry contains     %
        .                     .
        .                     .
        .                     .
0000000000000000      = 79 entry contains     %
```

1. PURPOSE

HL2FX is a procedure that converts OPCON coded data to fixed point (or Arithmetic formatted) data from a main program defined table containing one byte per word.

2. CALL FORMAT

HL2FX(FCOL, NUMB = SIGN, INT, FRAC)

WHERE:

HL2FX = the procedure name.

FCOL = an integer value specifying the column in a range of 1 to 80 on a card. This is also the number of the entry in a user defined table that the first character of the word is found in.

NUMB = an integer value, specifying the maximum number of characters or bytes to be considered in the conversion; i.e. how many characters of the OPCON item are to be converted into a floating point format.

SIGN = a one bit integer item indicating the sign of the converted value:

0 = positive
1 = negative

INT = a 47 bite integer item containing the integer portion of the fixed point value being converted.

FRAC = a 47 bit integer item containing the fractional portion of the fixed point value being converted.

3. LIMITATIONS AND ACCURACY

a. The character to be converted must be right justified in the field described by FCOL and NUMB. The sign need not be in the first column.

b. The procedure does not clear and/or reset the user defined table described below. The user must clear out this table each time before the procedure is called. This is to insure against possible errors that may result from picking up undesired characters from a previous conversion.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure. In any case this would be variable, as it would be a function of the size of the input parameter.

b. Storage - Unknown, probably about 125 words.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table from which is taken the unpacked OPCON items for repacking into a floating point word. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users requirements, however, 81 is usually chosen for an 80 column card.

The item declared in this table must have the name COLUM, and declared as a one byte, hollerith item, starting in the zeroth entry of the table. The first bit should be 42 to place the byte in the lowest position of the word.

A sample table declaration is as follows:

```
TABLE CITU R  80  S  1  $
BEGIN
  ITEM  COLUM  H  1  Ø  42  $
END
```

5. OTHER PROCEDURES CALLED

a. HL2IN

6. PROGRAMMING TECHNIQUE USED

a. The whole and fractional portion of the input parameter is converted by means of the HL2IN procedure and then placed in their respective items.

7. EXAMPLE

IF:

```
Table  CITU  R  80  S  1  $
Begin
  Item  COLUMN  H  1  0  42  $
End
Item  FCOL  I  47  U  P  2  $
Item  NUMB  I  47  U  P  8  S
Item  SIGN  I   1  U  $
Item  INT   I  47  U  $
Item  FRAC  I  47  U  $
```

AND:

THEN:

```
HL2FX(FCOL, NUMB = SIGN, INT, FRAC)  $
```

SIGN = 1
INT = ∅
FRAC = the octal equivalent of 0.23565 in core.

NOTE:

Table CITU would look as follows in core:

```
0000000000000120    =  nent word contains  80
0000000000000000    =  0 entry contains    %
0000000000000000    =  1 entry contains    %
0000000000000041    =  2 entry contains    -
0000000000000075    =  3 entry contains    .
0000000000000062    =  4 entry contains    2
0000000000000063    =  5 entry contains    3
0000000000000065    =  6 entry contains    5
0000000000000066    =  7 entry contains    6
0000000000000065    =  8 entry contains    5
0000000000000000    =  9 entry contains    %
        .                    .
        .                    .
        .                    .
0000000000000000    = 79 entry contains    %
```

1. PURPOSE

HL2HL is a procedure that takes up to 8 bytes of OPCON code from a predefined, one byte per entry table and repacks them into one word.

2. CALL FORMAT

HL2HL(FCOL, NUMB = HOL)

WHERE:

HL2HL = the procedure name.

FCOL = an integer value specifying the column in a range of 1 to 80 on a card. This is also the number of the entry in a user defined table that the first character of the word is found in.

NUMB = an integer value, specifying the maximum number of characters or bytes (up to eight) to be considered in the conversion.

HOL = the output value, declared as a hollerith item, that will contain the newly packed word.

3. LIMITATIONS AND ACCURACY

a. Never more than 8 characters can be transferred into one word. It is the programmers responsibility to see that NUMB is never greater than 8. No test is made to protect the programmer in this regard.

b. If NUMB-FCOL is less than 8 the leading characters of HOL will be set to blanks.

c. Accuracy is not applicable in this packing routine.

d. The procedure does not clear and/or reset the user defined table described below. The user must clear this table each time before the procedure is called. This is to insure against possible errors that may result from picking up undesired characters from a previous conversion.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure, but it should be essentially the same for each input parameter.

b. Storage - 46 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table from which is taken the unpacked OPCON items for repacking into a hollerith defined word. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users requirements, however 80 is usually chosen for an 80 column card.

The item declared in this table must have the name COLUM, and be declared as a one byte, hollerith item, starting in the zeroth entry of the table. The first bit should be 42 to place the byte in lowest portion of the word.

A sample table declaration is as follows:

```
TABLE  CITU  R  80  S  1  $
BEGIN
  ITEM  COLUM  H  1  Ø  42  $
END
```

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. For statements and the BYTE modifier is used to move each character from its table entry to the output parameter (HOL).

7. EXAMPLE

```
IF:
     Table  CITU  R  80  S  1  S
     Begin
       Item  COLUMN  H  1  0  42  $
     End
     Item  HOL  H  8  $
     Item  FCOL  I  47  U  P  5  $
     Item  NUMB  I  47  U  P  8  $

AND:
                                   HL2HL (FCOL, NUMB = HOL)  $
THEN:
                          HOL  =  3565E-3 when printed
```

NOTE:
Table CITU would look as follows in core:

```
0000000000000121      =  nent word contains   80
0000000000000000      =  0 entry contains     %
0000000000000000      =  1 entry contains     %
0000000000000005      =  2 entry contains     blank
0000000000000075      =  3 entry contains     .
0000000000000062      =  4 entry contains     2
0000000000000063      =  5 entry contains     3
0000000000000065      =  6 entry contains     5
0000000000000066      =  7 entry contains     6
0000000000000065      =  8 entry contains     5
0000000000000012      =  9 entry contains     E
0000000000000041      = 10 entry contains     -
0000000000000005      = 11 entry contains     blank
0000000000000063      = 12 entry contains     3
0000000000000000      = 13 entry contains     3
0000000000000000      = 14 entry contains     %
        .                      .
        .                      .
        .                      .
0000000000000000      = 79 entry contains     %
```

Item HOL, after the transfer was made, would look as follows in core:

```
6356666512410563
```

1. PURPOSE

HL2HO is a procedure that takes up to 8 characters of an OPCON coded item (hollerith) and places them in a format of one character per word in a programmer defined table.

2. CALL FORMAT

HL2HO(HOL, FCOL, NUMB)

WHERE:

HL2HO = the procedure name

HOL = the input value, declared as a hollerith item, that the user wishes to have unpacked to one character per word.

FCOL = an integer value specifying which column in the range of 0 to 120 on a printed page that the first character is to go into. This is also the number of the entry in a user defined table that the first character is placed in.

NUMB = an integer value specifying the number of places that the converted word is to occupy.

3. LIMITATIONS AND ACCURACY

a. If NUMB is greater than the number of bytes in the input item, the field will contain one or more leading spaces. If NUMB is less than the number of bytes in the input item, the leading bytes will not be transferred.

b. No provision is made to check the input parameter to see if it is greater than 8 characters in length. If greater then 8 characters, only the first 8 characters will be transferred.

c. The procedure does not clear and/or reset the user defined table described below. The user must clear out this table each time before the procedure is called. This is to insure against possible errors that may result from picking up undesired characters from a previous conversion.

d. Accuracy is not applicable in this unpacking routine.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time to operate the procedure, but it should be essentially the same for each input parameter.

b. Storage - 23 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table, in which, is placed the converted output values. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users need, however 120 is usually chosen to correspond to the 120 point columns available on the printer.

The item declared in this table must have the name CPOSN. It also must be defined as hollerith, containing one byte, and starting in the zeroth entry of the table. The first bit should be 42 to place it in the lowest position of the word, but this may be changed at the user's discretion.

A sample table declaration is as follows:

```
TABLE  COLUMN  R  120  S  1  $
BEGIN
  ITEM  CPOSN  H  1  Ø  42  $
END
```

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. Making use of a FOR statement and an assignment statement, the input parameter is unpacked one byte at a time and placed in the table as specified by the other input parameters.

7. EXAMPLE

IF:

```
Table COLUMN  R  120  S  1  $
Begin
  Item  CPOSN  H  1  0  42  $
End
Item  HOL  H  8  8H(CX3HØL7M)  $
Item  FCOL  I  47  U  P  2  $
Item  NUMB  I  47  U  P  8  $
```

AND:

```
                                HL2HO(HOL, FCOL, NUMB)  $
```

THEN:

Table COLUMN would look as follows in core:

```
0000000000000170      =   nent word contains    120
0000000000000000      =   0 entry contains      %
0000000000000000      =   1 entry contains      %
0000000000000010      =   2 entry contains      C
0000000000000035      =   3 entry contains      X
0000000000000063      =   4 entry contains      3
0000000000000020      =   5 entry contains      K
0000000000000060      =   6 entry contains      Ø
0000000000000021      =   7 entry contains      L
0000000000000067      =   8 entry contains      7
0000000000000022      =   9 entry contains      M
        .                       .
        .                       .
        .                       .
0000000000000000      = 120 entry contains      %
```

1. PURPOSE

HL2IN is a procedure that converts OPCON coded data to a signed decimal integer item from a main program defined table containing one byte per word.

2. CALL FORMAT

HL2IN(FCOL, NUMB = INTEG)

WHERE:

HL2IN = the procedure name

FCOL = an integer value, specifying a column in the range of 1 to 80 on a card. This is also the number of the entry in a user defined table that the first character of the word is found in.

NUMB = an integer value, specifying the maximum number of characters of bytes to be considered in the conversion; i.e. how many characters of the OPCON item are to be converted into a signed decimal integer.

INTEG = an item defined as an integer and containing the converted output value from the routine.

3. LIMITATIONS AND ACCURACY

a. Leading blanks are acceptable, but trailing blanks are not.

b. The input parameter may be of either the normal or exponential format. However, no decimal points may be included.

Example:

-34 or 51E + 3

c. The procedure does not clear and/or reset the user defined table described below. The user must clear this table each time before the procedure is called. This is to insure against possible errors that may result from picking up undesired characters from a previous conversion.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure. The time will be a variable factor however, as it will depend on the size of the input required.

b. Storage - 77 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table from which is taken the unpacked OPCON items for repacking into an integer declared item. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users requirements, however 80 is usually chosen for an 80 column card.

The item declared in this table must have the name COLUM, and be declared as a one byte, hollerith item, starting in the zeroth byte in the lowest position of the word.

A sample table declaration is as follows:

```
TABLE  CITU  R  80  S  1  $
BEGIN
  ITEM  COLUM  H  1  0  42  $
END
```

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. The OPCON value is converted, byte by byte, to an octal value that is equivalent to the original decimal value that is now in the OPCON mode.

7. EXAMPLE

IF:

```
Table CITU  R  80  S  1  $
Begin
  Item  COLUMN  H  1  0  42  $
End
Item  INTEG  I  48  S  $
Item  FCOL  I  47  U  P  2  $
Item  NUMB  I  47  U  P  7  $
```

AND:

```
                                HL2IN(FCOL, NUMB = INTEG)  $
```

THEN:

```
                        INTEG =  10101
```

NOTE:

Table CITU would look as follows in core:

```
0000000000000120      =  nent word contains   80
0000000000000000      =  0 entry contains     %
0000000000000000      =  1 entry contains     %
0000000000000005      =  2 entry contains     blank
0000000000000005      =  3 entry contains     blank
0000000000000062      =  4 entry contains     2
0000000000000063      =  5 entry contains     3
0000000000000065      =  6 entry contains     5
0000000000000066      =  7 entry contains     6
0000000000000065      =  8 entry contains     5
0000000000000000      =  9 entry contains     %
        .                      .
        .                      .
        .                      .
0000000000000000      = 79 entry contains     %
```

After the conversion INTEG will look as follows in core:

```
0000000000010101
```

The 23565 is assumed to octal and is converted into decimal.

1. PURPOSE

HL2OC is a procedure that converts unsigned OPCON coded data to an unsigned octal integer item from a main program defined table containing one byte per entry.

2. CALL FORMAT

HL2OC(FCOL, NUMB = OCTAL)

WHERE:

HL2OC = the procedure name

FCOL = an integer value, specifying a column in a range 1 to 80 on a card. This is also the number of the entry in a user defined table that the first character of the word is found in.

NUMB = an integer value, specifying the maximum number of characters or bytes to be considered in the conversion; i.e. how many characters of the OPCON item are to be converted into a signed decimal integer.

OCTAL = an item declared as an unsigned integer and containing the converted output value from the routine.

3. LIMITATIONS AND ACCURACY

a. Leading blanks are acceptable, but trailing blanks are not.

b. The input parameter may not be signed, consequently the output will not be signed.

c. The procedure does not clear and/or reset the user defined table described below. The user must clear this table each time before the procedure is called. This is to insure against possible errors that may result from picking up undesired characters from a previous conversion.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure. The time will be a variable factor however, as it will depend on the size of the input parameter.

4. PROCEDURE CHARACTERISTICS (cont)

b. Storage - 47 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table from which is taken the unpacked OPCON item for repacking into an integer declared item. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users requirement, however 80 is usually chosen for an 80 column card.

The item declared in this table must have the name COLUM, and be declared as a one byte, hollerith item, starting in the zeroth byte in the lowest position of the word.

A sample table declaration is as follows:

```
TABLE CITU  R  80  S  1  $
BEGIN
  ITEM  COLUM  H  1  0  42  $
END
```

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. FOR statements and the BIT modifier are made use of to convert each OPCON byte to its octal integer equivalent.

7. EXAMPLE

IF:

```
Table  CITU  R  80  S  1  $
Begin
  Item  COLUMN  H  1  0  42  $
End
Item  OCTAL  I  48  S  $
Item  FCOL  I  47  U  P  3  $
Item  NUMB  I  47  U  P  6  $
```

AND:

```
                              HL2OC (FCOL, NUMB = OCTAL)  $
```

THEN:

```
                        OCTAL =  23565 (base 8)
```

NOTE:

Table CITU would look as follows in core:

```
0000000000000120      =  nent word contains   80
0000000000000000      =  0 entry contains     %
0000000000000000      =  1 entry contains     %
0000000000000000      =  2 entry contains     %
0000000000000005      =  3 entry contains     blank
0000000000000062      =  4 entry contains     2
0000000000000063      =  5 entry contains     3
0000000000000065      =  6 entry contains     5
0000000000000066      =  7 entry contains     6
0000000000000065      =  8 entry contains     5
0000000000000000      =  9 entry contains     %
        .                     .
        .                     .
        .                     .
0000000000000000      = 79 entry contains     %
```

Item OCTAL, after the conversion is made, would look as follows in core:

```
0000000000023565
```

1. PURPOSE

HL2ST is a procedure that converts up to 8 bytes of OPCON coded data to a standard transmission code item from a main program defined table containing one byte per entry.

2. CALL FORMAT

HL2ST(FCOL, NUMB = STCODE)

WHERE:

HL2ST	=	the procedure name.
FCOL	=	an integer value, specifying a column in the range of 1 to 80 on a card. This is also the number of the entry in a user defined table that the first character of the word is found in.
NUMB	=	an integer value, specifying the maximum number of characters or bytes (up to eight) to be considered in the conversion.
STCODE	=	an item defined as standard transmission code and containing the converted output value from the routine.

3. LIMITATIONS AND ACCURACY

a. Never more than 8 characters can be transferred into one word. It is the programmers responsibility to see that NUMB is never greater than 8. No test is made to protect the programmer in th_ regard.

b. If NUMB-FCOL is less than 8 the leading characters of STCODE will be set to blanks.

c. There will be no errors in packing the characters into a word from the table.

d. The procedure does not clear and/or reset the user defined table described below. The user must clear this table each time before the procedure is called. This is to insure against possible errors that may result from picking up undesired characters from a previous conversion.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure, but it should be essentially the same for each input parameter.

b. Storage - 34 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table from which is taken the unpacked OPCON items for packing into a STC defined item. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users requirements, however 80 is usually chosen for an 80 column card.

The item declared in this table must have the name COLUM, and be declared as a one byte, hollerith item, starting in the zeroth entry of the table. The first bit should be 42 to place the byte in lowest portion of the word.

A sample table declaration is as follows:

```
TABLE  CITU  R  80  S  1  $
BEGIN
  ITEM  COLUM  H  1  Ø  42  $
END
```

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. For statements and the BYTE modifier is used to move each character from its table entry to the output parameter (STCODE).

7. EXAMPLE

IF:

```
Table  CITU  R  80  S  1  $
Begin
  Item  COLUMN  H  1  0  42  $
End
Item  STCODE  T  8  $
Item  FCOL  I  47  U  P  5  $
Item  NUMB  I  47  U  P  8  $
```

AND:

THEN: HL2ST (FCOL, NUMB = STCODE) $

STCODE = 3565E-3 when printed

NOTE:

Table CITU would look as follows in core:

```
0000000000000120      = nent word contains   80
0000000000000000      = 0 entry contains     %
0000000000000000      = 1 entry contains     %
0000000000000005      = 2 entry contains     blank
0000000000000075      = 3 entry contains     .
0000000000000062      = 4 entry contains     2
0000000000000063      = 5 entry contains     3
0000000000000065      = 6 entry contains     5
0000000000000066      = 7 entry contains     6
0000000000000065      = 8 entry contains     5
0000000000000012      = 9 entry contains     E
0000000000000041      = 10 entry contains    -
0000000000000000      = 11 entry contains    blank
0000000000000063      = 12 entry contains    3
0000000000000000      = 13 entry contains    3
0000000000000000      = 14 entry contains    %
        .                     .
        .                     .
        .                     .
0000000000000000      = 79 entry contains    %
```

Item STCODE, after the transfer was made, would look as follows in core:

6365666512410563

1. PURPOSE

IN2HO is a procedure that converts an integer item to an OPCON coded decimal value and places it in a main program defined table, one byte per word.

2. CALL FORMAT

IN2HO(INTEG, FCOL, NUMB)

WHERE:

IN2HO = the procedure name

INTEG = the input value defined as integer that the user wishes to have converted.

FCOL = an integer value specifying which column in the range of 1 to 120 on a printed page that the first character is to go into. This is also the number of the entry in a user defined table that the first character is placed in.

NUMB = an integer value specifying the number of places that the converted word is to occupy. One space should be allowed for the sign.

3. LIMITATIONS AND ACCURACY

a. If NUMB is equal to zero an immediate return is executed before any other instructions are executed. If NUMB is greater than the number of bytes in the input item, the field will contain one or more leading zeros. If NUMB is less than the number of bytes in the input item, the leading bytes will not be transferred.

b. The first column will be blank or contain a minus sign depending on the sign of the input parameter (INTEG). Leading zeros will be surpressed.

c. The accuracy should be complete.

d. The procedure does not clear and/or reset the user defined table described below. The user must decide, to what value he wants the table set to before the conversion is made. If he presets the table to zeros and then tries to print it, the converted value is going to be surrounded by %. For such a case it should be preset to OPCON blanks.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure. The time will be slightly variable as it will be a function of the size of the input parameter.

b. Storage - 50 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table in which to place the converted output values. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entrys may be chosen to fit the users need, however 120 is usually picked to correspond to the 120 print columns available on the printer.

The item declared in this table must have the name CPOSN. It also must be defined as hollerith, containing one byte, and starting in the zeroth entry of the table. The first bit should be 42 to place it in the lowest position of the word, but this may be changed at the user's discretion.

A sample table declaration is as follows:

```
TABLE  COLUMN  R  120  S  1  $
BEGIN
  ITEM  CPOSN  H  1  Ø  42  $
END
```

5. OTHER PROCEDURES CALLED

a. REMQUO

6. PROGRAMMING TECHNIQUE USED

a. Integer division is done by making use of the REMQUO procedure. The remainder is added to 60_8 to convert the value from octal to decimal OPCON one digit at a time.

7. EXAMPLE

IF:

```
Table  COLUMN  R  120  S  1  $
Begin
  Item  CPOSN  H  1  0  42  $
End
Item  INTEG  I  48  S  P  -35971  $
Item  FCOL  I  47  U  P  1  $
Item  NUMB  I  47  U  P  7  $
```

AND:

IN2HO(INTEG, FCOL, NUMB) $

THEN:

Table COLUMN would look as follows in core:

```
0000000000000170      =  nent word contains    120
0000000000000000      =  0 entry contains      %
0000000000000041      =  1 entry contains      -
0000000000000005      =  2 entry contains      blank
0000000000000063      =  3 entry contains      3
0000000000000065      =  4 entry contains      5
0000000000000071      =  5 entry contains      9
0000000000000067      =  6 entry contains      7
0000000000000061      =  7 entry contains      1
0000000000000000      =  8 entry contains      %
        .                      .
        .                      .
        .                      .
0000000000000000      = 119 entry contains     %
```

NOTE:

In core item INTEG would be:

7777777777671574 which is the octal equivalent of -35971.

1. PURPOSE

INTHI is a procedure that converts an unsigned integer item into an 8 byte OPCON item one word at a time.

2. CALL FORMAT

```
INTHI(INTEG = HOL)
```

WHERE:

INTHI = the procedure name.

INTEG = the input value to be converted, declared as an unassigned integer or arithmetic item containing no fractional bits.

HOL = the output parameter declared as an 8 byte hollerith item and containing the converted value.

3. LIMITATIONS AND ACCURACY

a. A complete word is converted one at a time.

b. No checks are made on the input parameter (INTEG). It is assumed that the input value is valid and lies between 0 and 99,999,999 which is the largest value that will fit in an 8 byte word.

c. The accuracy should be complete.

4. PROCEDURE CHARACTERISTICS

a. Operating time is unknown, but will be the same for all input parameters.

b. Storage - 27 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. The conversion is made by doing a successive series of divisions by 10. The converted value is added to 60, byte by byte to create the OPCON coded word.

7. EXAMPLE

IF:

```
Item  INTEG  I  47  U  P  25  $
Item  HOL    H  8   $
```

AND:

THEN: INTHI(INTEG = HOL) $

HOL = 25 when printed

NOTE:

Item HOL will look as follows in core after the conversion takes place:

0505050505056265

1. PURPOSE

NORM360 is a function that normalizes any angle supplied in degrees, to the range of 0 to 360.

2. CALL FORMAT

XX = NORM360(ANGLE)

WHERE:

XX	=	a program declared floating point item of the user to which is assigned the normalized angle.
NORM360	=	the function name; declared as a floating point item and containing the normalized output value.
ANGLE	=	the input parameter expressed in floating point degrees.

3. LIMITATIONS AND ACCURACY

a. The input parameter (ANGLE) must be expressed in degrees and/or fractional degrees.

b. The normalization will be completely accurate.

c. Input values equal to 0.0° or 360.0° will be returned unchanged.

4. PROCEDURE CHARACTERISTICS

a. The operating time is unknown, but it will be a function of the size of the input parameter. The larger the absolute value of the parameter is, the longer the time required to normalize it.

b. Storage - 12 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. If the input angle is greater than 360°, repetitive subtraction is done until the angle is reduced to a value equal to or less than 360°, but equal to or greater than 0°. If the input value is less than 360°, repetitive addition is done until the angle is increased to a value equal to or greater than 0°, but less than or equal to 360°.

7. EXAMPLE

IF:

```
Item  XX  F  $
Item  XXA  F  $
Item  ANGLE  F  P  730.0  $
Item  ANGLEA  F  P  -330.0  $
```

AND:

```
XX = NORM360(ANGLE)  $

XXA = NORM360(ANGLEA)  $
```

THEN:

```
XX = 10.0

XXA = 30.0
```

1. PURPOSE

OC2HO is a procedure that converts an unsigned octal integer item to OPCON code and places it in a main program defined, one entry per word table.

2. CALL FORMAT

OC2HO(OCTAL, FCOL, NUMB)

WHERE:

OC2HO = the procedure name.

OCTAL = the input parameter declared as an unsigned integer item with no fractional bits and containing the octal value to be converted.

FCOL = an integer value specifying which column in the range of 0 to 120 on a printed page that the first character is to go into. This is also the number of the entry in a user defined table that the first character is placed in.

NUMB = an integer value specifying the number of places that the converted word is to occupy.

3. LIMITATIONS AND ACCURACY

a. If NUMB is greater than the number of digits in the input item, the field will contain one or more leading zeros. If NUMB is less than the number of input digits, the leading digits will not be transferred.

b. NUMB can not be greater than 16. However, no check is incorporated to restrict NUMB to 16.

c. The procedure does not clear and/or reset the user defined table described below. The user must clear out this table each time before the procedure is called. This is to insure against possible errors that may result from picking up undesired characters from a previous conversion.

d. There will be no inaccuracy from unpacking the word and placing it in the table.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time to operate the procedure, but it should be essentially the same for each input parameter.

b. Storage - 24 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table, in which, is placed the converted output values. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users requirements, however 120 is usually chosen to correspond to the 120 point columns available on the printer.

The item declared in this table must have the name CPOSN. It also must be defined as hollerith, containing one byte, and starting in the zeroth entry of the table. The first bit should be 42 to place it in the lowest position of the word, but this may be changed at the user's discretion.

A sample table declaration is as follows:

```
TABLE  COLUMN  R  120  S  1  $
BEGIN
  ITEM  CPOSN  H  1  0  42  $
END
```

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. Making use of a FOR statement and an assignment statement, the input parameter is unpacked one byte at a time and placed in the table as specified by the other input parameters.

7. EXAMPLE

IF:

```
Table COLUMN  R  120  S  1  $
Begin
  Item  CPOSN  H  1  0  42  $
End
Item  OCTAL  I  47  U  P  235
Item  FCOL  I  47  U  P  2  $
Item  NUMB  I  47  U  P  8  $
```

AND:

THEN: OC2HO(OCTAL, FCOL, NUME) $

Table COLUMN would look as follows in core:

```
0000000000000170      =  nent word contains      120
0000000000000000      =  0 entry contains        %
0000000000000000      =  1 entry contains        %
000000000000006Ø      =  2 entry contains        Ø
000000000000006Ø      =  3 entry contains        Ø
000000000000006Ø      =  4 entry contains        Ø
000000000000006Ø      =  5 entry contains        Ø
000000000000006Ø      =  6 entry contains        Ø
000000000000006Ø      =  7 entry contains        Ø
000000000000006Ø      =  8 entry contains        Ø
000000000000006Ø      =  9 entry contains        Ø
000000000000006Ø      =  10 entry contains       Ø
000000000000006Ø      =  11 entry contains       Ø
000000000000006Ø      =  12 entry contains       Ø
000000000000006Ø      =  13 entry contains       Ø
000000000000006Ø      =  14 entry contains       Ø
0000000000000063      =  15 entry contains       3
0000000000000065      =  16 entry contains       5
0000000000000063      =  17 entry contains       3
0000000000000000      =  18 entry contains       %
          .                    .
          .                    .
          .                    .
0000000000000000      = 119 entry contains       %
```

NOTE:

Since the input value 235 is in decimal, it is converted to octal and will appear as follows in core.

[illegible]00000000353

OPCTOB

1. PURPOSE

OPCTOB is a function that converts a numeric OPCON coded item to an equivalent floating point item.

2. CALL FORMAT

XX = OPCTOB(HOL)

WHERE:

XX	=	a program declared item of the user to which the function output is assigned.
OPCTOB	=	the function name; declared as a floating point item and containing the function output.
HOL	=	the input parameter of the user declared as a 16 character hollerith item.

3. LIMITATIONS AND ACCURACY

a. Up to 16 OPCON characters may be inputed representing any signed or unsigned integer, fraction, or mixed number followed by an optional power of 10 in the form of the letter E followed by a signed or unsigned integer.

b. The conversion routine should be completely accurate up to the limitations of a floating point word.

4. PROCEDURE CHARACTERISTICS

a. The time of operation will vary between 3 and 16 milliseconds.

b. Storage - 147 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. The input value has 60 subtracted from it byte by byte. The new floating point word is then built from these converted digits.

7. EXAMPLE

IF:

```
Item  XX   F   $
Item  XXA  F   $
Item  HOL  H   16  7H(-0.2E35)  $
Item  HOLA H   16  11H(-0.56978E-6)  $
```

AND:

```
XX = OPCTOB(HOL)  $

XXA = OPCTOB(HOLA)  $
```

THEN:

```
XX = 5615022754404223

XXA = 6024316064041705
```

NOTE:

XX will now represent the input value in floating point. 5615022754405223 is the floating point representation in the machine of -0.2E35

1. PURPOSE

PCL is a procedure that converts latitude and longitude expressed in degrees, minutes, with or without seconds from OPCON code to signed floating point radians.

2. CALL FORMAT

PCL(LATOP, LNGOP, FRMAT = LATRAD, LNGRAD)

WHERE:

PCL = the procedure name.

LATOP = the parameter specifying the latitude in OPCON code according to the format indicated below.

LNGOP = the input parameter specifying the longitude in OPCON code according to the format indicated below.

FORMAT = the input parameter declared as arithmetic, with three bits, unsigned, and no fractional bits indicating the format of the incoming latitude and longitude to be converted. Where:

Ø =	DDMMYbbb	5 character latitude
	DDDMMXbb	6 character longitude
1 =	DDMMSSYb	7 character latitude
	DDDMMSSX	8 character longitude

AND:

DD and DDD	= DEGREES
MM	= MINUTES
SS	= SECONDS
Y	= NORTH or SOUTH
X	= EAST or WEST
b	= BLANK

LATRAD = the output parameter declared as floating point and containing the converted latitude in floating point radians.

LNGRAD = the output parameter declared as floating point and containing the converted longitude in floating point radians.

3. LIMITATIONS AND ACCURACY

a. Both latitude and longitude must be of the corresponding formats. No checks for the validity of format are made before entering the routine.

b. The input parameters must also be left justified in the word.

c. There are no known inaccuracies in the routine.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure. However, the time should be about the same for each input parameter.

b. Storage - 212 machine words are required.

c. This is a pure JOVIAL procedure.

d. The converted value will be positive for North and East coordinates and negative for South and West coordinates.

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. The latitude and longitude are converted to binary fixed point degrees with the sign prefixed according to the quadrant that they lie in. Multiplication of the fixed point degrees by the floating point constant PI/180 results in floating point radians.

7. EXAMPLE

IF:

```
Item  LATOP   H  8  8H(4954N   )  $
Item  LNGOP   H  8  8H( 8930W  )  $
Item  FRMAT   A  3  U  P  Ø  $
Item  LATRAD  F  $
Item  LNGRAD  F  $
```

AND:

THEN:

```
PCL(LATOP, LNGOP, FRMAT = LATRAD, LNGRAD)  $
```

LATRAD = 0.865683309 in floating point format in core

LNGRAD = -1.56206968 in floating point format in core

RADIANS

1. PURPOSE

RADIANS is a function that converts angular degrees in floating point to floating point radians.

2. CALL FORMAT

XX = RADIANS(DEG)

WHERE:

XX = the program declared item of the user to which the function output is assigned.

RADIANS = the function name; declared as a floating point number containing the converted value.

DEG = the input value in floating point degrees that will be converted into radians.

3. LIMITATIONS AND ACCURACY

a. There is no limitation on the size of the input parameter, DEG, other than those normally imposed on a floating point word. The input parameter must be expressed in degrees and/or fractional degrees. The format of degrees, minutes, seconds is unacceptionable.

b. No checks are made on the input parameters. All input is assumed to be valid.

c. The accuracy of the conversion should be greater than ten decimal places.

4. PROCEDURE CHARACTERISTICS

a. The time of operations is unknown, but will be the same for all input parameters.

b. Storage - 3 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. MATHEMATICAL METHOD USED

a. The input parameter, DEG, is multiplied by the appropriate constant to convert it into radians.

7. EXAMPLE

IF:

```
Item  XX  F  $
Item  DEG  F  P  20.0  $
```

AND:

```
                XX  =  RADIANS(DEG) $
```

THEN:

```
                XX  =  0.3490658503
```

1. PURPOSE

ST2HO is a procedure that converts up to 8 characters of a standard transmission coded item (STC) to OPCON code, placing it in a main program defined table, one byte per word.

2. CALL FORMAT

ST2HO(STCODE, FCOL, NUMB)

WHERE:

ST2HO = the procedure name.

STCODE = the input parameter, declared as a standard transmission code item, that the user wishes to have converted to OPCON.

FCOL = an integer value specifying which column in the range of 0 to 120 on a printed page that the first character is to go into. This is also the number of the entry in a user defined table that the first character is placed in.

NUMB = an integer value specifying the number of placed that the converted word is to occupy.

3. LIMITATIONS AND ACCURACY

a. If NUMB is greater than the number of bytes in the input item, the field will contain one or more leading spaces. If NUMB is less than the number of bytes in the input item, the leading bytes will not be transferred.

b. No provision is made to check the input parameter to see if it is greater than 8 characters in length. If greater then 8 characters, only the first 8 characters will be transferred.

c. The procedure does not clear and/or reset the user defined table described below. The user must clear out this table each time before the procedure is called. This is to insure against possible errors that may result from picking up undesired characters from a previous conversion.

d. There will be no inaccuracy from unpacking the word and placing it in a table.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time to operate the procedure, but it should be essentially the same for each input parameter.

b. Storage - 33 machine words are required.

c. This is a pure JOVIAL procedure.

d. This procedure requires a user defined table, in which, is placed the converted output values. This table may be defined according to any name that the user may choose, but must be serial and have one word per entry. The total number of entries may be chosen to fit the users need, however 120 is usually chosen to correspond to the 120 print columns available on the printer.

The item declared in this table must have the name CPOSN. It also must be defined as hollerith, containing one byte, and starting in the zeroth entry of the table. The first bit should be 42 to place it in the lowest position of the word, but this may be changed at the user's discretion.

A sample table declaration is as follows:

```
TABLE  COLUMN  R  120  S  1  $
BEGIN
  ITEM  CPOSN  H  1  Ø  42  $
END
```

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. Making use of a FOR statement and an assignment statement, the input parameter is unpacked one byte at a time and placed in the table as specified by the other input parameters.

7. EXAMPLE

```
IF:
     Table  COLUMN  R  120  S  1  $
     Begin
       Item  CPOSN  H  1  0  42  $
     End
     Item  STCODE  T  8  P  8T(CX3KØL7M)  $
     Item  FCOL  I  47  U  P  2  $
     Item  NUMB  I  47  U  P  8  $

AND:
                                   ST2HO(STCODE, FCOL, NUMB)  $

THEN:
     Table COLUMN would look as follows in core:

       0000000000000170      = nent word contains   120
       0000000000000000      = 0 entry contains     %
       0000000000000000      = 1 entry contains     %
       0000000000000010      = 2 entry contains     C
       0000000000000035      = 3 entry contains     X
       0000000000000063      = 4 entry contains     3
       0000000000000020      = 5 entry contains     K
       0000000000000060      = 6 entry contains     Ø
       0000000000000021      = 7 entry contains     L
       0000000000000067      = 8 entry contains     7
       0000000000000022      = 9 entry contains     M
               .                      .
               .                      .
               .                      .
       0000000000000000      =119 entry contains    %
```

TYPE

1. PURPOSE

TYPE is a procedure that converts a 50 character Hollerith item into its typewriter TYPE code equivalent to be outputted on the console typewriter.

2. CALL FORMAT

```
TYPE(MESSAGE)
```

WHERE:

TYPE = the procedure name

MESSAGE = an item declared as 50 Hollerith characters that contains the desired message to be outputted.

3. LIMITATIONS AND ACCURACY

a. The input parameter must be declared as a 50 Hollerith item. If the actual message is less than 50 characters the remainder of the field must be blanked out with Hollerith blanks.

b. The following call format may also be used in addition to the one showed above:

```
TYPE(50H(...THE DESIRED MESSAGE OR BLANKS FOR 50 CHARACTERS...)
```

This eliminates the mechanics of declaring an item containing the desired message. However the whole field of 50 characters must be used, either with the message itself or blanks.

c. Only lower case characters will be typed.

d. A carriage return will be made at the end of the type out.

e. File 18 must be declared for the typewriter.

f. Accuracy is complete.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure.

b. Stores - 135 machine words are required, including the 50 character message.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. A table containing the typewriter code equivalent for the OPCON characters is referenced in the conversion and an IO call is made to output the 50 characters to the typewriter.

7. EXAMPLE

IF:

```
Item  MESSAGE  H   50  P
50H (TYPEWRITER MESSAGE HERE                              ) $
```

AND:

```
TYPE(MESSAGE)  $

TYPE(50H(TYPEWRITER MESSAGE HERE                          )) $
```

THEN:

TYPEWRITER MESSAGE HERE is printed out by the typewriter along with trailing blanks and a carriage return.

1. PURPOSE

TYPE'MSG is a procedure that converts 1 to 6 words of OPCON data to its typewriter TYPE code equivalent and types the result out on the console typewriter.

2. CALL FORMAT

```
TYPE'MSG(FC, CORE, WORDS, ERFILE)
```

WHERE:

TYPE'MSG = the procedure name.

FC = a 15 bit unsigned integer item containing the file code to which the typewriter is assigned to.

CORE = a table name in which the message is declared.

WORDS = a 15 bit unsigned integer item containing the number of words to be typed out.

ERFILE = a 15 bit unsigned integer item containing the file code of a print or list tape on which error messages should be written if data transmission errors should occur on the typewriter.

3. LIMITATIONS AND ACCURACY

a. Full 1604 words are typed; thus the user must insert blanks as needed.

b. Alphabetic characters will always appear in upper case.

c. The input parameter WORDS must be in the range of 1 to 6. If a value greater than 6 is supplied it will be reset to 6.

d. The input parameter CORE should be declared as a table name, however it may also be declared as an item name. If declared as an item name the user will generate a 664 error in compilation that may be ignored in running.

4. PROCEDURE CHARACTERISTICS

a. No figures are available for the time required to operate this procedure, however it will be a variable depending on the length of the output message.

b. Storage - 130 machine words are required.

c. This is a pure JOVIAL procedure.

d. Legal OPCON characters having no TYPE code equivalent will be interpreted as spaces.

e. The typewriter carriage is restored to the left margin before and after the message is typed.

5. OTHER PROCEDURES CALLED

a. CK'STATS

6. PROGRAMMING TECHNIQUE USED

a. Conversion is by direct indexing, rather then searching. Typing is done in character mode. Both are incorporated to optimize the routine.

7. EXAMPLE

```
IF:
     TABLE  CORE  R  6  S  $
       BEGIN
         Item  MSG  H  27  P  27H(TWEEDLE DUM AND TWEEDLE DEE)  $
       END
     Item  FC  I  15  U  P  18  $
     Item  ERFILE  I  15  U  P  27  $

AND:
          TYPE'MSG(FC, CORE, NWDSEN(CORE), ERFILE)  $
THEN:
       The message TWEEDLE DUM AND TWEEDLE DEE will be typed out on
       the console typewriter.
```

1. PURPOSE

ZZ is a procedure that converts a floating point number to an OPCON coded item 16 characters long.

2. CALL FORMAT

ZZ(FLOAT = HOL)

WHERE:

ZZ	=	the procedure name.
FLOAT	=	the input parameter declared as a floating point item.
HOL	=	the output parameter declared as a 16 character hollerith item containing the converted value.

3. LIMITATIONS AND ACCURACY

a. The input parameter, FLOAT, is not checked for validity as a floating point number.

b. The accuracy should only be limited by the converted value not being rounded off.

4. PROCEDURE CHARACTERISTICS

a. The time for operating this procedure is about 2.2 milliseconds.

b. Storage - 247 machine words are required.

c. This is a pure JOVIAL procedure.

d. The 16 character output parameter will contain the following format:

Byte #	Contents
0	space
1	plus or minus sign
2	period
3-10	mantissa
11	E
12	plus or minus sign
13-15	characteristic

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUE USED

a. Each portion of a floating point word is converted to an integer value and then repacked as an OPCON item.

7. EXAMPLE

IF:

```
Item  FLOAT  F  P  0.55  $
Item  HOL  H  16  $
```

AND:

```
ZZ(FLOAT = HOL)  $
```

THEN:

HOL +.55000000E+ 0 when printed

NOTE:

In core FLOAT would look like: 2000431463146314
In core HOL would look like: 0542756565606060
6060601242050560

APPENDIX A

APPENDIX A

SUMMARY OF JOVIAL PROCEDURE LIBRARY CONTENTS

I. ARITHMETIC PROCEDURES

1. FEXP	A function that raises the constant "e" to a given power.
2. LOGNAT	A function that calculates the natural (Base e) logarithm of a number greater than zero.
3. REMQUO	A procedure that performs integer division and produces a separate quotient and remainder.
4. SQRT	A function that calculates the square root of any floating point number greater than zero.
5. STATS	A multiple input function that is capable of determining the arithmetic mean, geometric mean, harmonic mean, mode, median, variance and standard deviation for a set of values assumed normal in distribution.

II. TRIGONOMETRIC PROCEDURES

1.	ARCCOS	A function that computes the arccosine of a number between minus one and plus one inclusive.
2.	ARCSIN	A function that computes the arcsine of a number between minus one and plus one inclusive.
3.	ARCTAN	A function that computes the arctangent of any number.
4.	COSH	A function that computes the hyperlic cosine of a number.
5.	COSIN	A function that computes an approximation of the trigonometric cosine of a specified angle in radians with a maximum error of less than 0.5×10^{-10} of any angle.
6.	COTAN	A function that computes an approximation of the trigometric cotangent.
7.	SIN	A function that computes an approximation of the trigonometric sine of an angle supplied in radians with a maximum error of less than 0.5×10^{-10} of any angle.
8.	SINH	A function that computes the hyperbolic sine of a number.
9.	TAN	A function that computes an approximation of the trigometric tangent.
10.	TANH	A function that computes the hyperbolic tangent of a number.

III. CONVERSION PROCEDURES

1.	ALPHCON	A multiple input function that converts IBM 7090 code to OPCON code, or OPCON code to IBM 7090 code.
2.	DEGREES	A function that converts radians expressed in floating point to angular degrees and fractions, expressed in floating point.
3.	FBINRY	A function that converts numeric values of OPCON code to its integer equivalent, one word at a time.
4.	FL2HO	A procedure that converts a floating point to OPCON code.
5.	FOPCON	A function that converts an integer item into an 8 byte OPCON item.
6.	FX2HO	A procedure that converts an arithmetic item to OPCON code.
7.	HL2FL	A procedure that converts OPCON coded data to floating point formatted item.
8.	HL2FX	A procedure that converts OPCON coded data to a fixed point formatted item.
9.	HL2HL	A procedure that takes up to 8 bytes of OPCON coded from a predefined one byte per entry table and reassembles them into one word.
10.	HL2HO	A procedure that takes up to 8 characters of an OPCON coded item and places them in a format of one character per word in a programmer defined table.
11.	HL2IN	A procedure that converts OPCON coded data to a signed decimal integer item.
12.	HL2OC	A procedure that converts unsigned OPCON coded data to an unsigned octal integer.
13.	HL2ST	A procedure that converts OPCON symbolic to Standard Transmission Code.
14.	IN2HO	A procedure that converts an integer item to an OPCON coded decimal value.

III. CONVERSION PROCEDURES (Continued)

15.	INTHI	A procedure that converts an unsigned integer item into an 8 byte OPCON item one word at a time.
16.	NORM360	A function that converts any angle, positive or negative, supplied in floating point degrees to its equivalent positive value between Ø and 36Ø degrees.
17.	OC2HO	A procedure that converts an unsigned octal integer to OPCON code.
18.	OPCTOB	A function that converts an OPCON coded number to its floating point equivalent.
19.	PCL*	A procedure that converts latitude and longitude in OPCON code to floating point radians.
20.	RADIANS	A function that converts floating point angular degrees to floating point radians.
21.	ST2HO	A procedure that converts up to 8 characters of a standard transmission coded item to OPCON code.
22.	TYPE*	A procedure that converts the 50 Hollerith character item designated in the call into its typewriter TYPE code equivalent to be outputted at the console typewriter.
23.	TYPE'MSG*	A procedure that converts 1 to 6 words of OPCON data to its typewriter TYPE code equivalent, and type the result on the console typewriter.
24.	ZZ	A procedure that converts a floating point number to an OPCON coded item 16 bytes long.

APPENDIX A

IV. FILE PROCESSING SYSTEM PROCEDURES

1.	DELET	A procedure that reallocates core buffer area due to the opening of a new file or the closing of an existing file in FPS.
2.	ENTER	A procedure that opens the file designated in the File Description Item and allocates an I/O work area in core memory to be used for the file being opened in FPS.
3.	ENTERF#	A procedure that opens an IPS master for use with FPS library procedures.
4.	ENY	A procedure that determines the current entry position of any file in FPS.
5.	EOF	A procedure that tests for End-of-File in FPS.
6.	EOR	A procedure that tests for End-of-Record in FPS.
7.	EOS	A procedure that tests for End-of-Segment in FPS.
8.	EXIT	A procedure that closes the processing of a file in FPS.
9.	FDES'ENT	A procedure that obtians the entry number in Table FDES for the file designated in the procedure call in FPS.
10.	I'O	A procedure that handles all peripheral device manipulations and actual data transmissions for the procedures of FPS.
11.	LADDL	A procedure that obtains the address plus 100 of the last memory word loaded by the SSEC loader in FPS.
12.	LOST	A procedure that logs the count of unrecoverable parity errors from each FPS file for which the I/O error parameter was specified.

APPENDIX A

IV. FILE PROCESSING SYSTEM PROCEDURES (Continued)

13.	READF#	A procedure that reads a record from an IPS format tape, converts it to FPS format 4, and transfer a set at-a-time to the user program.
14.	REC	A procedure that determines the current record of any file in FPS.
15.	RELEASE	A procedure that is used in conjunction with READ to provide a means of skipping over any remaining entries in a record in FPS.
16.	RWRIT	A procedure that writes the contents of a file's I/O work area onto tape in FPS.
17.	SEG	A procedure that determines the current segment of any file in FPS.
18.	SORT*	A procedure that sorts the records of the input file into either ascending or descending order with respect to key item values in each record in FPS.
19.	TYPE*	A procedure that converts the 50 Hollerith character item designated in the call into its typewriter character equivalent to be outputted at the console in FPS.
20.	WRITE	A procedure that transfers either an entry of a print line image from the first entry of a serial table to a magnetic tape file in FPS.
21.	WSEG	A procedure that writes an End-of-Segment mark in a file in FPS.

APPENDIX A

V. PROGRAM MAINTENANCE AND UTILITY TYPE PROCEDURES

1. CK'STATS	A procedure that waits for a "file not busy" status. Checks for other non-normal stati, and take appropriate action.
2. CLEAR	A procedure that clears an area of core, defined by first word address and number of words, to +0 or blanks.
3. CORE	A procedure that dumps a specified memory area onto file 27 whenever entered.
4. CSCAN	A procedure that locates and classifies fields within an unpacked (one character per word) image table that generally represents card columns.
5. DUMP	A procedure that initiates a dump service request to SSEC for that portion of memory requested in the call.
6. GET'CARD	A procedure that reads cards (or card images), double buffered, and unpack them as required by CSCAN and the 'nn2nn routines.
7. INTERRUPT	A procedure that initiates a request to interrogate the interrupt status for internal (arithmetic) faults.
8. REEL	A function that queries the TINF table and returns the reel number assigned to a given magnetic tape file.
9. RESTART	A procedure that initiates a request to SSEC to establish restart points.
10. SECURITY	A procedure that initiates an SSEC request to determine the security classification of a file.
11. SNOCTAL	A procedure that provides the capability of utilizing an unlimited number of octal correction. Core dump cards per computer run.
12. STAT'CK	A procedure that produces a typewriter comment for all data transmissions resulting in other than a normal or EOF status.

V. PROGRAM MAINTENANCE AND UTILITY TYPE PROCEDURES (Continued)

13.	STORE	A procedure that blocks "print" lines, 23 lines per block, and write the lines double buffered. Automatically insert page headings, classifications pagination as needed
14.	TIME	A procedure that initiates a request to SSEC for the real time.
15.	TPRELSE	A procedure that generates a typewriter message to the operator directing him to save a magnetic tape reel to some given date.
16.	TYPE	A procedure that converts a 50 Hollerith character item to its typewriter character code equivalent for outputting on the console typewriter.
17.	TYPE'MSG*	A procedure that converts 1 to 6 words of OPCON data to its TYPE code equivalent, and type the result on the console typewriter. The original message is untouched.

APPENDIX A

VI. NAVIGATIONAL PROCEDURES

1. DESTNK	A procedure that calculates a new position and course (great circle) when given a position, course, time, and SOA, assuming constant speed for the time interval specified and a great circle track.
2. GRCRBL	A procedure that calculates either the great circle or rhumb line course between two given points.
3. PCL*	A procedure that converts latitude and longitude in OPCON code to floating point radians.
4. PGC	A procedure that calculates the great circle distance between two points.
5. PLA	A procedure that checks latitudes in OPCON code to insure that they are between 90 degrees North and 90 degrees South.
6. PLN	A procedure that checks to see that East and West longitudes are between 0 degrees and 180 degrees.
7. POSITP	A procedure that solves for an unknown point along a great circle or rhumb line course.

APPENDIX A

VII. PLOTTER PROCEDURES

1. ANGLINE	A procedure that draws a straight line between point A and point B on a given vector. The vector is described having a base point X, Y, and angle O. A and B are described as distances from the base point.
2. ANGPOSIT	A procedure that supplies the X and Y coordinates of any point on a given vector. The vector is described having a base point S, Y and angle O. The point is described as a distance from the base point.
3. ARC	A procedure that draws an arc described by a radius; center X, Y; and angles at a and b. The arc will be drawn from A and B.
4. ASPECIAL	A procedure that draws one of the special characters from the special character set centered on p point X, Y, with the prescribed height with an inclination described by the sine and cosine of the angle. The pen may be in either an up or down position from the present pen position to X, Y.
5. AXIS	A procedure that draws a line from X, Ø to Ø, Ø and from Ø, Ø to Ø, Y.
6. BARGRAF	A procedure that draws a bar with the lower left corner at X, Y. A portion of the bar may be shaded. The width of the bar may vary by increments of Ø.2 inches.
7. BETTERXY	A procedure that takes two sets of S,Y coordinates and determines which set is closest to the present pen position. The closest will be placed in the first set of output coordinates, and the farthest in the second set.
8. CENTER	A procedure that will give the X or Y coordinate to be used as the starting point for a label to center it over a given X or Y coordinate.

APPENDIX A

VII. PLOTTER PROCEDURES (Continued)

9. CIRCLE	A procedure that draws a circle around point X, Y with a given radius.
10. CURVE	A procedure that draws a smooth curve between N equally spaced values of Y [where Y = F(X)], using the Gregory-Newton Forward and Backward Difference Formulae and the Bessel Central Difference Formula.
11. DASH	A procedure that draws a dashed line from X_0, Y_0 to X_1, Y_1. The length of the dash and the length of the space may be varied.
12. DRAW	A procedure that draws a line from the present pen coordinates to the designated coordinates.
13. FIGURES	A procedure that draws from 1 to 8 characters from the OPCON character set, with the lower corner of the first character at X, Y. The height and angle of the characters may vary.
14. FRSTGRAF	A procedure that must be called before any of the other plot library system procedures are called. The routine sets up the work area for the plot procedures. It opens the plot output file. Block 1 will be a test pattern to insure that the scale adjustment factor is properly set. Block two will be the first block used for your output. The Ø, Ø point will be 6 inches from the test pattern along the X axis and the number of inches from the right margin will be taken from the call parameter. The output can be produced for either the 11" or 3Ø" plotter.
15. LABELS	A procedure that draws from 9 to an infinite number of characters from the OPCON character set, with the lower left corner of the first character at X, Y. The height and angle of the base line of the character may vary.

VII. PLOTTER PROCEDURES (Continued)

16.	LASTGRAF	A procedure that must be the last procedure called when using the plot library system. Takes care of all final housekeeping required and closes output files.
17.	MOVE	A procedure that positions the pen at the X, Y coordinates designated. A line will not be drawn.
18.	NUGRAF	A procedure that clears and housekeeps plot routines at end of each block; establishes new Ø, Ø point for next plot, relative to old Ø, Ø point.
19.	PLOT	The central routine of system, contains PCA, PCC, PCD or PCF as described in FOCCPAC TR-23 with slight modifications to exist in a JOVIAL environment. It will not be called by the user of the system. All possible entrances to the routine have been incorporated in other procedures.
20.	POINTER	A procedure that draws a line from X_1, Y_1 to X_2, Y_2 with an arrow-head located at X_2, Y_2 in designated size.
21.	PRESENT	A procedure that determines the X, Y coordinate of the pen.
22.	SPECIAL	A procedure that draws of the special characters from the special character set centered on X, Y with a prescribed height at a prescribed angle. The pen may be moved in either an up or down position from its present location.
23.	TICKS	A procedure that draws any number of tick marks along the Y axis, beginning at any point on Y, with any interval, for any length.

APPENDIX A

VIII. SEARCH AND SORT PROCEDURES

1.	BINSRCH	A procedure that performs a history search, making use of a single or multiple word key, on a sorted serial table containing one or more words per entry.
2.	DDSORT	A procedure that will sort (logically) in ascending order, entries of a single or multiple word per entry serial table via a key field in the entry.
3.	DLOAD	A procedure that will read unpacked tape records from tape or cards. Select those meeting preestablished criteria and build a predetermined sort key in core. The records are packed on disc at maximum density.
4.	SORT*	Sorts the records of an input file into either ascending or descending order with respect to key item values in each record in FPS.
5.	SRC	Locates an identifier in a compool (PPS-111) directory using the binary search technique.

APPENDIX B

FLEET OPERATIONS CONTROL CENTER
U. S. PACIFIC FLEET
FPO San Francisco 96617

FOCCPAC:C31A:mi
5230
Ser 1080
21 April 1969

From: Commanding Officer, Fleet Operations Control Center,
U. S. Pacific Fleet
To: Distribution List

Subj: Changes to JOVIAL Procedure Library Documentation; forwarding of

Ref: (a) JOVIAL Procedure Library, FOCCPAC Tech Note 3 of Apr 1968

Encl: (1) Change 2 of 1 April 1969

1. Enclosure (1) is forwarded for incorporation into reference (a).

2. Changes should be made as indicated on the "Record of Changes".

3. The "Record of Changes" will replace the "Effective Changes" page issued with Change 1. It should be initialed and inserted following the Table of Contents.

F. M. SLASINSKI
By direction

Distribution:
CINCPAC (03)
NAVCOSSACT (05)
NAVCOSSACT PACOM DET (30)
DDC Cameron Station (20)

RECORD OF CHANGES

CHANGE NO. DATE	PROCEDURE (NO. OF PAGES)	ENTERED BY DATE
1 22 Jul 68	SECTION 3: REPLACE: PCL (1) SECTION 6: ADD: INTRODUCTION (1), DESUNK (2), GRCRBL (2), PCL (3), PGC (3), PLA (3), PLN (2), POSITP (3) APPENDIX A: REPLACE: VI. NAVIGATIONAL PROCEDURES (1)	_____
2 1 Apr 69	QUICK REFERENCE INDEX: REPLACE: All pages except the last page "NOTES TO THE QUICK REFERENCE INDEX" SECTION 3: ADD: XBCD'OPC (2) SECTION 5: ADD: CRFF (2), CRDSTR (2) SECTION 8: ADD: INTRODUCTION (1), BINSRCH (3), DDSORT (2), DLOAD (4), SORT (4), SRC (2) APPENDIX A: REPLACE: III. CONVERSION PROCEDURES (1) IV. FILE PROCESSING SYSTEM PROCEDURES (2) V. PROGRAM MAINTENANCE AND UTILITY TYPE PROCEDURES (2) VIII. SEARCH AND SORT PROCEDURES (1) ADD: The last page containing footnotes	_____

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
ALPHCON(ARGUM,WHICH) [func]	Converts IBM 7090 Code to OPCON code and vice versa. ALPHCON = Converted output value, in HOL 8. ARGUM = Input value to be converted, in HOL 8. WHICH = Indicates: 0 = OPCON to BCD 1 = BCD to OPCON	COMP 43	UNKN	3	REMQUO
ANGLINE(DIST1,DIST2,XX, YY, ANGLE) [proc]	Draws a straight line between point A and point B on a given vector. DIST1 = Distance from X,Y to A, in Flt Pt Inch. DIST2 = Distance from X,Y to B, in Flt Pt Inch. XX = X Coord of base Pt, in Flt Pt Inch. YY = Y Coord of base Pt, in Flt Pt Inch. ANGLE = Vector angle, measured counter clockwise from the X axis,in Flt Pt Deg.	N/A 41	UNKN	7	COSIN DRAW MOVE RADIANS SIN
ANGPOSIT(DIST,XB,YB,ANGLE = XX,YY) [proc]	Returns the X and Y coordinates of a point on a given vector. DIST = Distance from the base. Pt.,XB: in Flt Pt Inch. XB = X Coord of base,in Flt Pt Inch. YB = Y Coord of base, in Flt Pt Inch. ANGLE = Angle of vector having a base point of XB, YB; in Flt Pt Deg. XX = X Coord of the defined point, in Flt Pt Inch. YY = Y Coord of the defined point, in Flt Pt Inch.	N/A 31	UNKN	7	COSIN RADIANS SIN
ARC(RAD1,XX,YY,ANG1,ANG2) [proc]	Draws an arc defined by the input parameters RAD1 = Radius of the arc, in Flt Pt Inch. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch. ANG1 = Starting angle, in Flt Pt Deg. ANG2 = Terminating angle, in Flt Pt Deg.	N/A 1543	UNKN	7	COSIN DRAW NORM360 RADIANS SIN

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
ARCCOS(XX) [func]	Computes the arccosine of a number. ARCCOS = Output value, in Flt Pt Rad. XX = Input value, in Flt Pt.	10-5 15	1-3	2	ARCSIN
ARCSIN(XX) [func]	Computes the arcsine of a number. ARCSIN = Output value, in Flt Pt Rad. XX = Input value, in Flt Pt.	10-5 126	1-3	2	
ARCTAN(XX) [func]	Computes the arctangent of a number. ARCTAN = Output value, in Flt Pt Rad. XX = Input value, in Flt Pt.	10-5 16	1.6	2	
ASPECIAL(XX,YY,PEN,HEIGHT, SINC,COINC,SYMB) [proc]	Draws one of the special characters from the special character set. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch. PEN = Pen position, in Int according to: Ø = up 1 = down HEIGHT = Character height, in Flt Pt Inch. SINC = Sine of the angle of inclination, in Flt Pt. COINC = Cosine of the angle of inclination, in Flt Pt. SYMB = An Int value specifying the special character desired. See end of Index for Symbols.	N/A 17	UNKN	7	PLOT
AXIS(XEXT,YEXT) [proc]	Draws a line from Coord point Ø,Ø to X,Ø and from Ø,Ø to Y,Ø. XEXT = Length of Xaxis, in Flt Pt Inch. YEXT = Length of Yaxis, in Flt Pt Inch.	N/A 12	UNKN	7	DRAW MOVE
BARGRAF(BGRØ,BGR1,BGR2, BGR3,BGU) [proc]	Draws a shaded or unshaded bar. BGRØ = X Coord of lower left corner of the bar in Flt Pt Inch.	N/A 127	UNKN	7	DRAW MOVE

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	BGR1 = Y Coord of lower left corner of the bar, in Flt Pt Inch. BGR2 = Height of basic bar, in Flt Pt inch. BGR3 = Height of the shaded portion, in Flt Pt Inch. EGU = Number of Int units the bar is wide; 1 unit = 0.2 Inch.				
BETTERXY(X1,Y1,X2,Y2 =	Determines which of the two X,Y Coord supplied is closest to the present pen position. X1 = X Coord of 1st set, in Flt Pt Inch. Y1 = Y Coord of 1st set in Flt Pt Inch X2 = X Coord of 2nd set, in Flt Pt Inch. Y2 = Y Coord of 2nd set, in Flt Pt Inch. XB1 = X Coord of closest set, in Flt Pt Inch. YB1 = Y Coord of closest set, in Flt Pt Inch. XB2 = X Coord of farthest set, in Flt Pt Inch. YB2 = Y Coord of farthest set, in Flt Pt Inch.	N/A 60	UNKN	7	SQRT
BINSRCH(SRTBL,SRWDS,ETRY, ARTBL,ARWDS = SRENT,SRALT) [proc]	Performs a binary search, using a single or multiple word key, on a sorted serial table. SRTBL = Name of the table to be searched. SRWDS = Number of words per entry of table SRTBL. ETRY = Number of the word in the entry against which the sort will be carried out against, in integer. ARTBL = Table name that contains the argument. ARWDS = Number of words per entry of table ARTBL. SRENT = Entry at which the match was found or point where the argument should be inserted if no match was found. SRALT = Statement label to which a transfer is made if a match is not found.	N/A 132	UNKN	8	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
CENTER(CEND,CENC,CENH) [func]	Supplies the X or Y Coord at which a label is to begin if it is to be centered on given X or Y point. CENTER = X or Y Coord that the label should begin at, in Flt Pt. CEND = Point over which the label is to be centered, in Flt Pt Inch. CENC = Number of characters in the label, in Flt Pt. CENH = Height of characters, in Flt Pt Inch.	N/A 14	UNKN	7	
CIRCLE(RADI,XX,YY) [proc]	Draws a circle around Coord X,Y with the given radius. RADI = Radius of the circle in, Flt Pt Inch. XX = X Coord of center, in Flt Pt Inch. YY = Y Coord of center, in Flt Pt Inch.	N/A 12	UNKN	7	ARC
CK'STATS(FC,RESULT,HIST = ALT.) [proc]	Waits for a non-busy status to be returned by an I/O device, checks for other non-normal stati and takes appropriate action. FC = File code of file being checked in Int. RESULT = Ø; then if the status of FC is other than 0, 1, 2, 3, of 13 the task will be terminated with TERM REAS 37XX where "XX" is the status found. = 1; then, if illegal stati are found, an abnormal termination will occur only if total errors previously detected by CK'STATS is 99. HIST = File code of list tape for every message, in Int. If equal to Ø no logging will be done. ALT. = Statement label to which CK'STATS will return if a segment mark or end-of-file is found.	N/A 74	UNKN	5	INTHI

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
CLEAR(WHERE,NREGS,TYPE) [proc]	Clears an area of core to +Ø or blanks. WHERE = The first word of core to be cleared, in Int. NREGS = The number of words to clear, in Int. TYPE = Status item, if equal to: 1; clear to +Ø. 2; clear to blanks. If not equal to 1 or 2 will return without clearing.	N/A 40	UNKN	5	
CORE(FWA,NWD,NAME,NDWPE, FORM) [proc]	Dumps a specified memory area on file Ø4. FWA = First word address in Int. NWD = Number of words to be dumped in Int. NAME = Up to 8 OPCON characters that may be printed as an Ident in the control line in Hol 8. NDWPE = Number of words per sub-section, or entry, in Int. FORM = Format of data to be printed. Ø = octal 1 = OPCON	N/A 250	UNKN	5	
COSH(XX) [func]	Computes the hyperbolic cosin of a number. COSH = Output value, in Flt Pt. XX = Input value, in Flt Pt.	$\frac{10^{-11}}{12}$	1.8	2	FEXP
COSIN(XX) [func]	Computes the cosine of a number. COSIN = Output value, in Flt Pt. XX = Input value, in Flt Pt Rad.	$\frac{.5X10^{-5}}{46}$	2.0	2	SIN
COTAN(XX) [func]	Computes the cotangent of a number. COTAN = Output value, in Flt Pt. XX = Input value, in Flt Pt Rad.	$\frac{10^{-10}}{77}$	3.5	2	TAN

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
CRDSTR(FC,FWA,NUMB) [proc]	Prestores card images from the 1604 computer to an open ended file. FC = File code for file to which cards will be prestored, in Int. FWA = First word address of the cards to be prestored, in Int. NUMB = Number of cards to be prestored, in Int.	N/A 35	UNKN	5	
CRFF() [func]	Senses for a carriage return (typed by operator) on the 1604 console typewriter. CRFF = Boolean indicating: 1 = auto job termination 2 = auto job termination not requested	N/A 7	UNKN	5	
CSCAN(MOD = FTYP,FCOL,NCOL) [proc]	Locates and classifies fields within an unpacked (one character per word) image that generally represent card columns. MOD = Int, indicating when CSCAN is to return to the calling program. Ø = locate next field and return. 1 = return where a $ has been encountered or all columns have been examined. FTYP = Int specifying the type of field located by the scan. Ø-15 indicates a separator. 22-30 indicates data. See procedure writeup for individual codes.	N/A 545	UNKN	5	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	FCOL = Int, specifying the first column of a field containing a character that is not separator, space, or part of a field definition. NCOL = Int, specifying the number of columns comprising the field. Will be Ø for separators.				
CURVE(PTAB,PSPAC,PSTRT, PCHR,PCHT) [proc]	Draws a smooth curve between N equally spaced values of Y. PTAB = Table name of value to be plotted. Nent must be equal the number of points. PSPAC = X spacing between values of Y, in Flt Pt Inch. PSTRT = Position of the first point from the X axis, in Flt Pt Inch. PCHR = Number of the special character set to mark the points within Arit. See end of Index for symbols. PCHT = Height of character, in Flt Pt Inch. If = 0, no characters will be printed.	N/A 27	UNKN	7	PLOT
DASH(XASH,YASH,XASH1,YASH1, DASLEN,DOTLEN) [proc]	Draws a dashed line from X_0,Y_0 to X_1,Y_1. XASH = X Coord of the initial Pt, in Flt Pt Inch. YASH = Y Coord of the initial Pt, in Flt Pt Inch. XASH1 = X Coord of the final Pt, in Flt Pt Inch.	N/A 75	UNKN	7	DRAW FIX MOVE SQRT

CALL TYPE	DESCRIPTION	ACCUR FWRDS	TIME MSEC	SEC	OTHERS USED
	YASH1 = Y Coord of the final Pt, in Flt Pt Inch. DASLEN = Dash length, in Flt Pt Inch. DOTLEN = Length of space, in Flt Pt Inch.				
DDSORT(ADDR,NUMB,EWDS, SWDS,FWSK) [proc]	Sorts logically in ascending order, entries of a single or multiple word per entry serial table via a key field in the entry. ADDR = Location of the table, in Int. NUMB = Nent of the table, in Int. EWDS = Number of words per entry in the table (NWDSKN or some calculated value), in Int. SWDS = Number of full words in the sort key, in Int. FWSK = Starting word of the sort key, in Int.	N/A 113	UNKN	8	
DEGREES(RAD) [func]	Converts radians to degrees. DEGREES = The output value, in Flt Pt Deg. RAD = The input value, in Flt Pt Rad.	10^{-9} 4	UNKN	5	
DELET(INDX,OPER) [proc]	Reallocates core buffer area due to opening or closing a file in FPS. INDX = Index to file entry in tables FDES and CBUF, in Int. OPER = Entrance point to DELET, in Int. 1 = Entered from procedure EXIT. 2 = Entered from procedure ENTER.	N/A 551	UNKN	4	
DESUNK(LATA,LONGA,CRSA, DTIME,SOA = LATB, LONGB,CRSB) [proc]	Calculates a new position and course (great circle) when given position, course, time, and SOA, assuming constant speed for the time interval specified and great circle track. LATA = Original Lat, in Flt Pt Rad. LONGA = Original Long, in Flt Pt Rad.	UNKN 147	UNKN	7	ARCCOS ARCSIN COSIN SIN

CALL TYPE	DESCRIPTION	ACCUR FWRDS	TIME MSEC	SEC	OTHERS USED
	CRSA = Original course, in Flt Pt Rad. DTIME = Number of hours ship is to travel, in Flt Pt. SOA = Speed, in Flt Pt Kt. LATB = New Lat, in Flt Pt Rad (negative if south). LONGB = New Long, in Flt Pt Rad. GRSB = Great circle course, in Flt Pt Rad.				
DLOAD(TP,DR,HIST,FMS'SW, TRK1,KEY,DPE,LAST = RDBUF,NUMR) [proc]	Reads unpacked records from tape and places them on disc. TP = Input file code, in Int. DR = Random disc output file code, in Int. HIST = Output tape file code for listing error messages, in Int. FMS'SW = A Bool item indicating input tape format: 0 = Not in FMS format. 1 = In FMS format. RDBUF = Table name for n+1 words per entry where "n" = number of words in a key. TRK1 = Beginning track number, in Int. KEY = Name of the Key table to be built. WPE = Number of words per entry (n+1) in KEY, in Int. LAST = 'LOC (statement label) where the procedure is to return to. NUMT = Number of tracks used, in Int. NUMR = Number of records packed, in Int.	N/A 1130	N/A	8	CK'STATS
DRAW(XX,YY) [proc]	Draws a line from the present pen Coord to the designated Coord. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch.	N/A 10	UNKN	7	PLOT
DUMP(FADDR,LADDR,ID,FORM) [proc]	Initiates a dump service request to SSEC for that portion of memory requested. FADDR = Address of the first word to be dumped, in Arit. LADDR = Address of the last word to be dumped, in Arit. ID = User identification of the	N/A 17	UNKN	5	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	dump, in Hol 2. FORM = Format of the output, in Arit. 0 = Octal 1 = OPCON				
ENTER(FDI) [proc]	Opens the file designated and allocates an I/O work area in core to be used for the file being opened, in FPS. FDI = File description item, in Hol 50. See procedure writeup for details.	N/A 2Ø24	UNKN	5	DELET I'O LADDL TYPE
ENTERF(FDI) [proc]	Opens an IPS master tape for use with FPS library procedures by bypassing the first two records. FDI = File description item, in Hol 50. See procedure writeup for details.	N/A 11	UNKN	3	ENTER
ENY(DUM'FILE) [func]	Determine the current entry position of any file in FPS. ENY = Entry position, in Int. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 17	UNKN	5	FDES'DENT
EOF(DUM'FILE) [func]	Tests for an end-of-file on an FPS tape. EOF = 1; for end-of-file present. Ø; for end-of-file not present. In Bool format. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 17	UNKN	5	FDES'DENT
EOR(DUM'FILE) [func]	Tests for an end-of-record on an FPS pate. EOR = 1; for end-of-record present. 0; for end-of-record not present. In Bool format. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 22	UNKN	5	FDES'DENT

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
EOS(DUM'FILE) [func]	Tests for an end-of-segment on an FPS tape. EOS = 1; for segment mark present. Ø; for segment mark not present. In Bool format. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 2Ø	UNKN	5	FDES'ENT
EXIT(DUM'FILE) [proc]	Closes the processing of an FPS file. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 26	UNKN	5	DELET FDES'ENT I'O RWRIT
FBINRY(FOPCOD) [func]	Converts numeric values of OPCON code to its integer equivalent. FBINRY = Integer equivalent, in Arit. FOPCOD = Input parameter, in Hol 8.	100% 53	UNKN	3	
FDES'ENT(DUM'FILE) [func]	Obtain the entry number in table FDES for the file designated in the call in FPS. FDES'ENT = Entry number, in Int. DUM'FILE = File description item, in Hol 50. See procedure writeup for details.	N/A 31	UNKN	5	TYPE
FEXP(FUU) [func]	Raises the constant E to a given power. FEXP = Output results, in Flt Pt. FUU = Input parameter, in Flt Pt.	10^{-10} 77	.85	1	
FIGURES(XX,YY,COUNT, HEIGHT,ANGLE, CARACTOR) [proc]	Draws up to 8 characters from the OPCON character set. XX = X Coord of 1st character, in Flt Pt Inch. YY = Y Coord of 1st character, in Flt Pt Inch. COUNT = Number of characters to be printed, (1 to 8), in Arit. HEIGHT = Character height, in Flt Pt Inch.	N/A 17	UNKN	7	PLOT

CALL TYPE	DESCRIPTION	ACCUR %WRDS	TIME MSEC	SEC	OTHERS USED
	ANGLE = Angle of the characters from the + X axis, in Flt Pt Deg. CARACTOR = Characters to be plotted, in Hol 8.				
FL2HO(FLOTE,FIRST,NUMB) [proc]	Converts a Flt Pt item to OPCON code. FLOTE = Input parameter, in Flt Pt. FIRST = First entry of a table that the OPCON conversion will go into, in Int. NUMB = Number of table entries the converted value is to occupy, in Int.	100% 12ø	UNKN	3	IN2HO OC2HO
FOPCON(FBININ) [func]	Converts an integer item to an 8 byte OPCON item. FOPCON = Output parameter, in Hol 8. FBININ = Input parameter, in Arit.	10^{-9} 36	UNKN	3	
FRSTGRAF(PLOTTER,OUTFILE, WORK,STRT'PT) [proc]	Plotter housekeeping routine. Must be called before any other plotter procedure. PLOTTER = 0; for 30 Inch Plotter. 1; for 11 Inch Plotter. In Flt Pt. OUTFILE = Output file code for the generated plotter tape, in Arit. WORK = Table name allocating a work area for the system. Nent should be as large as possible and at least 8. If Nent = 1, locations 40000_8 to 77776_8 will be used. STRT'PT = Starting point from the right margin, in Flt Pt Inch.	N/A 41	UNKN	7	DRAW NUGRAF PLOT TICKS
FX2HO(FIN,FRA,FIRST,NUMB, NUMPL) [proc]	Converts an Arithmetic item to OPCON code. FIN = Int portion of the input parameter to be converted, in Int. FRA = Fractional portion of the input parameter to be converted, in Int. FIRST = First entry of a table that the OPCON conversion will to into, in Int.	100% 62	UNKN	3	IN2HO

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	NUMB = Number of table entries that the converted value is to occupy, in Int. NUMPL = Number of table entries allocated for the fractional portion of the converted value, in Int.				
GET'CARD(CD'FL,ER'FL = EDFL.) [proc]	Reads cards, double buffered, and unpacks them as required by CSCAN and the "XX2XX" conversion procedures. CD'FL = Card reader input file code, in Int. ER'FL = List tape file code for error, in Int. EDFL. = Statement label specifying where the procedure is to return control to if a segment mark or EOFL is encountered.	N/A 135	UNKN	5	CK'STATS
GRCRBL(LATA,LONGA,LATB, LONGB,IND = DIST, CRSE) [proc]	Calculates either the Great Circle or Rhumb Line course and distance between two given Pts. LATA = Lat of departure Pt. in Flt Pt Rad. LONGA = Long of departure Pt, in Flt Pt Rad. LATB = Lat of destination, in Flt Pt Rad. LONGB = Long of destination, in Flt Pt Rad. IND = 0; for Great Circle computation 1; for Rhumb Line computation. In Boolean Format. DIST = Distance from departure point to destination, in Flt Pt Rad. CRSE = Initial course to sail, in Flt Pt RAD.	UNKN 327	UNKN	6	ARCCOS ARCSIN ARCTAN COSIN

CALL TYPE	DESCRIPTION	ACCUR %WRDS	TIME MSEC	SEC	OTHERS USED
HL2FL(FIRST,NUMB, = FLT) [proc]	Converts OPCON coded data to a Flt Pt item. FIRST = First entry of a table that the OPCON coded byte is in, in Int. NUMB = Number of table entries to be converted, in Int. FLT = Converted output item, in Flt Pt.	UNKN 125	UNKN	3	HL2AE
HL2FX(FIRST,NUMB = SIGN, INT, FRAC) [proc]	Converts OPCON coded data to an Arit item. FIRST = First entry of a table that the OPCON coded byte is in, in Int NUMB = Number of table entries to be converted, in Int. SIGN = 1; for negative. = 0; for positive. In Int. INT = Int portion of the converted value, in Int. FRAC = Fractional portion of the converted value, in Int.	UNKN 127	UNKN	3	HLCON
HL2HL(FIRST,NUMB = HOL) [proc]	Packs 8 bytes of OPCON coded data from a table to an 8 byte OPCON item. FIRST = First entry of a table that the OPCON coded byte is in, in Int. NUMB = Number of table entries to be converted (1 to 8), In Int. HOL = Packed OPCON item, in Hol 8.	100% 51	UNKN	3	
HL2HO(HOL,FIRST,NUMB) [proc]	Unpacks an 8 byte OPCON coded item and places it in a one byte per entry table. HOL = OPCON item to be unpacked, in Hol 8. FIRST = First table entry that the first byte is to be placed in, in Int. NUMB = Number of table entries to be used in the unpacking, in Int.	100% 26	UNKN	3	

CALL TYPE	DESCRIPTION	ACCUR %WRDS	TIME MSEC	SEC	OTHERS USED
HL2IN(FIR,NO = INT) [proc]	Converts OPCON coded data to a signed decimal integer item. FIR = First table entry that the first byte is to be placed in, in Int. NO = Number of table entries to be used in the unpacking, in Int. INT = Converted output value, in Int.	100% 111	UNKN	3	
HL2OC(FIRST,NUMB = OCT) [proc]	Converts OPCON coded data to an unsigned Octal integer. FIRST = First table entry that the first byte is to be placed in, in Int. NUMB = Number of table entries to be used in the unpacking, in Int. OCT = Converted output item, in Int.	100% 52	UNKN	3	
HL2ST(FIRST,NUMB = TRC) [proc]	Converts OPCON coded data to a simple STC coded item. FIRST = First table entry that the first byte is to be placed in, in Int. NUMB = Number of table entries to be used in the unpacking, in Int. TRC = Converted output item, in STC 8.	100% 40	UNKN	3	
IN2HO(INT,FIRST,NUMB) [proc]	Converts a simple Int item to OPCON code. INT = Input value to be converted, in Int. FIRST = First entry of a table that the OPCON coded item is placed in, in Int. NUMB = Number of table entries the converted value is to occupy, in Int.	100% 54	UNKN	3	REMQUO
INTHI(INT = HOL) [proc]	Converts an unsigned integer item to an 8 byte OPCON item. INT = Input value to be converted, in Int.	100% 34	UNKN	3	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	HOL = Converted output value, in Hol 8.				
INTERRUPT(=FAULT,INSTRCTN, ADDRESS) [proc]	Interrogates the interrupt status for internal (arithmetic faults). FAULT = Fault type, in Arit. 1 = Divide 2 = Shift 3 = Overflow 4 = Exponent (overflow) 5 = Exponent (underflow) 6 = all others INSTRCTN = Upper or lower address indicator, in Bool 0 = upper address 1 = lower address ADDRESS = Address where the fault occurred, in Arit.	N/A 12	UNKN	5	
I'O(ENTNR,DESOP) [proc]	Handles all peripheal device manipulation and actual data transmission for the procedures of the FPS system. ENTNR = File description entry number in table FDES, in Int. DESOP = I/O operation desired, in Int. 1 = Open File 2 = Read 3 = Write 4 = Position 5 = Close File 6 = Write Segment Mark	N/A 1275	UNKN	4	LOST TYPE

	CALL TYPE	DESCRIPTION	ACCUR %WRDS	TIME MSEC	SEC	OTHERS USED
LADDL	[func]	Obtain the address plus 100 of the last memory word loaded by the SSEC loader in FPS. LADDL = Address + 100 of the last address loaded.	N/A 12	UNKN	4	
LASTGRAF	[proc]	The last procedure called in making use of the plotter procedures. Performs housekeeping chores.	N/A 16	UNKN	7	PLOT PRESENT
LOGNAT(UU)	[func]	Computes the natural logarithm of a number. LOGNAT = Output value, in Flt Pt. UU = Input value, in Flt Pt.	100% 114	.85	1	
MOVE(XX,YY)	[proc]	Positions the plotter pen at the indicated X,Y Coord without drawing a line. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch.	N/A 8	UNKN	7	PLOT
NORM360(ANGLE)	[func]	Normalizes any angle supplied in degrees to the range of 0 to 360. NORM360 = Normalized value, in Flt Pt Deg. ANGLE = Input value, in Flt Pt Deg.	100% 14	UNKN	3	
NUGRAF(XX,YY)	[proc]	Clears and housekeeps plotter routines at the end of each block. XX = New X Coord, in Flt Pt Inch from the previous X,Y Coord. YY = New Y Coord, in Flt Pt Inch from the previous X,Y Coord.	N/A 12	UNKN	7	PLOT
OC2HO(OCT,FIRST,NUMB)	[proc]	Converts an unsigned octal integer to OPCON code. OCT = Octal input parameter, in Int. FIRST = First entry of a table that the OPCON conversion will go into, in Int.	100% 26	UNKN	3	

CALL TYPE	DESCRIPTION	ACCUR %WRDS	TIME MSEC	SEC	OTHERS USED
	NUMB = Number of table entries the converted value is to occupy, in Int.				
OPCTOB(HOL) [func]	Converts an OPCON coded number to its floating point equivalent. OPCTOB = Converted output value, in Flt Pt. HOL = Input value, in Hol 8.	100% 165	3-16	3	
PCL(LATOP,LNGOP,FRMAT = LATRN,LNGRN) [proc]	Converts OPCON coded latitude and longitude to signed floating point radians. LATOP = Lat, in Hol 8. LNGOP = Long, in Hol 8. FRMAT = Format indication, in Arit. 0 = 5 character Lat = 6 character Long 1 = 7 character Lat = 8 character Long. LATRN = Converted Lat, in Flt Pt Rad. LNGRN = Converted Long, in Flt Pt Rad.	UNKN 233	UNKN	6	
PGC(LAT1,LON1,LAT2,LON2, FORMT = GCDEG,GCNMI) [proc]	Calculates the great circle distance between two points. LAT1 = Lat of the first point, in Hol 8. LON1 = Long of the first point, in Hol 8. LAT2 = Lat of the second point, in Hol 8. LON2 = Long of the second point, in Hol 8. FORMT = Format indicator, in Arit 0 = 5 character Lat 6 character Long 1 = 7 character Lat 8 character Long GCDEG = Great Circle distance, in Flt Pt Rad. GCNMI = Great Circle distance, in Flt Pt Naut Mi.	UNKN 112	UNKN	6	ARCOS ARCSIN COSIN PCL SIN
PLA LTCK1,FRMT1 = LATIN) [proc]	Checks magnitude and format validity of Latitude in OPCON code. LTCK1 = Lat to be checked, in Hol 8.	N/A 60	UNKN	6	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	FRMT1 = Format indicator, in Arit. 0 = 5 character Lat 1 = 7 character Lat LATIN = Validity check indicator, in Arit. 0 = Invalid 1 = Valid 2 = blank				
PLN(LNCK2,FRMT2 = LNGIN) [proc]	Checks magnitude and format validity of Longitude in OPCON code. LNCK2 = Long to be checked, in Hol 8. FRMT2 = Format indicator, in Long 0 = 7 character Long 1 = 8 character Long LNGIN = Validity check indicator, in Arit. 0 = Invalid 1 = Valid 2 = Blank	N/A 55	UNKN	6	
PLOT(PP) [proc]	Never called by the user. This is a central routine used by the plotter procedures.	N/A 1063	N/A	7	COSIN RADIANS SIN
POINTER(X1,Y1,X2,Y2,SIZE) [proc]	Draws a line from X1,Y1, to X2,Y2 with an arrow-head located at X2,Y2 in designated Size. X1 = X Pt at which to start the line, in Flt Pt Inch. Y1 = Y Pt at which to start the line, in Flt Pt Inch. X2 = X Pt to start the arrow head, in Flt Pt Inch. SIZE = Length size of the arrow head, in Flt Pt Inch.	N/A 43	N/A	7	ASPECIAL SQRT
POSITION(DUM'FILE,DSEGD, DCRED,DENTD)	Positions an input file to a particular entry in the file in FPS. DUM'FILE = File description item, in Hol 50. See procedure write up for details. DSEGD = Desired segment number, in Int. DRECD = Desired record number, in Int. DENTD = Desired entry number, in Int.	N/A 263	N/A	4	FDES'ENT I'O TYPE

CALL TYPE	DESCRIPTION	ACCUR FWRDS	TIME MSEC	SEC	OTHERS USED
POSITP(LATA,LONGA,LATB, LONGB,SOA,TAB,TAP, IND = LATP,LONGP) [proc]	Solves the Lat and Long of an unknown point along a Gt Circle or Rhumb Ling Course. LATA = Lat of Dep Pt, in Flt Pt Rad. LONGA = Long of Dep Pt, in Flt Pt Rad. LATB = Lat of Dest Pt, in Flt Pt Rad. LONGB = Long of Dest Pt, in Flt Pt Rad. SOA = Speed of advance from Dep Pt to the unknown point in Naut Mi and tenths of Naut Mi, in Arit. TAB = Time in Hr and tenths, required to sail from Dep Pt to Dest, in Arit. TAP = Time in Hr and tenths, required to sail from Dep Pt to Unkn posit, P, in Arit. IND = Boolean indicator as follows: Ø = Great Circle Computation 1 = Rhumb Line Computation LATP = Lat of Unkn posit, P, in Flt Pt. LONGP = Long of Unkn posit, P, in Flt Pt.	UNKN 532	UNKN	3	ARCCOS ARCSIN ARCTAN COSIN SIN SQRT
PRESENT(= XX,YY) [proc]	Supplies the present X and Y Coord of the pen. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch.	N/A 6	UNKN	7	PLOT

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
RADIANS(DEG) [func]	Converts angular degrees and fractions to Radians. RADIANS = Output value, in Flt Pt. DEG = Input value, in Flt Pt degrees.	10^{-11} 5	UNKN	3	
READ(DUM'FILE,DUM'TABL) [proc]	Transfer an entry of an input file to the first entry of a serial table in FPS. DUM'FILE = File Description Item, in Hol 50. See procedure write up for details. DUM'TABL = Table name of the serial table that the entry will be placed in.	N/A 341+	UNKN	4	FDES'ENT I'O TYPE
READF(DUM'FILE,DUM'TABL) [proc]	Reads a record from an IPS format tape, converts it to FPS format 4 supplying it to the user one set at a time. DUM'FILE = File Description item, in Hol 50. See procedure write up for details. DUM'TABL = Table name of the serial table that the entry will be placed in.	N/A 366+	UNKN	4	FDES'ENT I'O TYPE
REC(DUM'FILE) [func]	Determines the current record of any file in FPS. REC = Current record in the given file, in Int. DUM'FILE = File Description item, in Hol 50. See procedure write up for details.	N/A 16	UNKN	4	FDES'ENT TYPE
REEL(FC) [func]	Queries the TINF table and returns the reel number assigned to a given magnetic tape file. REEL = Reel number, right justified, in Hol 8. FC = File code, in Int.	N/A 42	UNKN	5	POPCON
RELEASE(DUM'FILE) [proc]	Skips over existing records in an entry, in FPS. DUM'FILE = File Description item, in Hol 50. See procedure write up for details.	N/A 26	UNKN	4	FDES'ENT RWRIT I'O TYPE

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
REMQUO(DND,SOR = QUO,REM) [proc]	Performs Int division and produces a separate quotient and remainder. DND = Dividend, in Int. SOR = Divisor, in Int. QUO = Quotient, in Int. REM = Remainder, in Int.	COMP 12	UNKN	1	
RESTART(LVP,PVP] [proc]	Initiates a request to SSEC to establish restart points. LVP = Last valid restart point, in Arit. PVP = Penultimate valid restart point, in Arit.	N/A 7	UNKN	5	
RWRIT(DUMI) [proc]	Writes the contents of a file's I/O work area onto tape, in FPS. DUMI = Entry number in Table FDES for the file to be processed, in Int.	N/A 2Ø	UNKN	4	I'O TYPE
SECURITY(FIIL = CLASS) [proc]	Initiates a request to SSEC to determine the security classification of a file. FIIL = File code of the desired file, in Arit. CLASS = A status item indicating the file classification. Ø = Unclassified 1 = Confidential 2 = Secret 3 = Top Secret	N/A 6	UNKN	5	
SEG(DUM'FILE) [func]	Determines the current segment of any file in FPS. SEG = A counter containing the segment number, in Arit. DUM'FILE = File Description item, in Hol 50. See procedure write up for details.	N/A 17	UNKN	4	FDES'ENT TYPE
SIN(XX) [func]	Computes the Sine of any angle supplied in radians. SIN = Output value, in Flt Pt. XX = Input value, in Flt Pt Radians.	10^{-10} 1Ø5	1.25	1	

	CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
SINH(UU)	[func]	Computes the hyperbolic sine of a number. SINH = Output value in, Flt Pt. UU = Input value in, Flt Pt.	$\frac{10^{-11}}{12}$	1.8	2	FEXP
SNOCTAL	[proc]	Provides for an unlimited number of octal correction and core dump cards per run.	N/A 532	UNKN	5	
SORT(DUM'FILE,DUMBFILE)	[proc]	Sorts the records of an input file according to the key item value in each file in FPS. DUM'FILE = Input or Output File Description item, in Hol 50. See procedure write up for details. DUMBFILE = Output or input File Description Item, in Hol 50. See procedure write up for details.	N/A 272+	UNKN	4	FDES'ENT
SPECIAL(XX,YY,PEN,HEIGHT, ANGLE,SYMBOL)	[proc]	Draws a character from the special character set centered on Pt X,Y. XX = X Coord, in Flt Pt Inch. YY = Y Coord, in Flt Pt Inch. PEN = Position of the pen, in Arit. Ø = up 1 = down HEIGHT = Character height, in Flt Pt Inch. ANGLE = Slope of the symbol, in Flt Pt Deg. SYMBOL = Number of the special character, in Arit. See end of Index for symbols.	N/A 15	UNKN	7	PLOT
SQRT(XX)	[func]	Calculates the square root of a number. SQRT = Output value, in Flt Pt. XX = Input value, in Flt Pt.	UNKN 64	.75	1	
SRC(NAME = XREF,XENT)	[proc]	Locates an identifier in the PPS III Compool directory. NAME = Identifier to be located, in 1 to 8 characters, in Hol 8. XREF = Value from the cross	N/A 55	UNKN	8	

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
	reference item in the directory entry containing the identifier equal to NAME, in Int. XENT = Number of the directory entry containing the identifier equal to NAME, in Int.				
STAT'CK(DITEMF,DITEMT,XX) [proc]	Generates a typewriter comment for all data transmission resulting in other than a normal or EOF status. DITEMF = UNKNOWN DITEMT = UNKNOWN XX = UNKNOWN	N/A 106	UNKN	5	
STATS(CORE,SKIP,WDNT,TYPE) [func]	Computes the Arit mean, Geometric mean, harmonic mean, mode, medium, variance, and standard deviation of a set of value, assumed normal in distribution. STATS = Output of routine, in Flt Pt. CORE = Table name containing the values to be operated on. SKIP = Words per entry in table CORE, in Int. WDNT = Relative word-in-entry of values to be operated on in table CORE, in Int. TYPE = Status item specifying the operation desired. Ø = Mean 1 = Mode 2 = Median 3 = Standard Deviation 4 = Variance 5 = Geometric Mean 6 = Harmonic Mean	UNKN 243	UNKN	1	
ST2HO(TRC,FIRST,NUMB) [proc]	Converts a simple STC coded item to OPCON code. TRC = Input value to be converted in STC 8. FIRST = First entry a table that the OPCON coded item is placed in, in Int. NUMB = Number of table entries that the converted value is to occupy, in Int.	COMP 36	UNKN	5	

CALL TYPE	DESCRIPTION	ACCUR WRDS	TIME MSEC	SEC	OTHERS USED
STORE(WHEN,BLANK'CK,EXCESS, FILE'CLD,CORE, DEGREE,LIMIT,ERROR) [proc]	Block "prints" lines, 23 lines per block and writes them double buffered.	N/A 1647	UNKN	5	CK'STATS INTH1

WHEN = Status item where
- ∅ = Normal call, neither the first nor last.
- 1 = First call to write the file.
- 2 = Force end-of-report.

BLANK'CK = Bool item, where:
- 1 = Ignore blank lines
- ∅ = Process blank lines as good data

EXCESS = Space needed for a logical data unit, in Int.

FILE'CD = File code of output file, in Int.

CORE = Table name containing data to be stored.

DEGREE = Classification, in Int.
- ∅ = unclassified
- 1 = confidential
- 2 = secret
- 3 = top secret

LIMIT = Lines per page, including header, but not classification and page-no lines, in Int.

ERROR = File code of the file for listing I/O error messages, in Int.

	CALL TYPE	DESCRIPTION	ACCUR / #WRDS	TIME MSEC	SEC	OTHERS USED
TAN(UU)	[func]	Computes the tangent of an angle in Radians. TAN = Output value, in Flt Pt. UU = Input value, in Flt Pt Rad.	.5x10-11 / 16	3.4	2	COSIN SIN
TANH(UU)	[func]	Computes the hyperbolic tangent of a number. TANH = Output value, in Flt Pt. UU = Input value, in Flt Pt.	10-11 / 21	1.8	2	FEXP
TICKS(TCST,TCNO,TCDS,TCLT)	[proc]	Draws any number of tick marks of any length, at any interval, beginning at any point along Y. TCST = Y value of first Tick in Flt Pt Inch. TCNO = Number of ticks, in Arit. TCDS = Dist between ticks, in Flt Pt Inch. TCLT = Tick length, in Flt Pt Inch.	N/A / 26	UNKN	7	DRAW MOVE
TIME(CLOCK)	[proc]	Initiates a request to SSEC for a query of the real time clock. CLOCK = Real time by: YRMODAHRMNSC right justified, in Hol 16.	N/A / 12	UNKN	5	
TPRELSE(FC,LNTH,UNIT, DATE1 = RELDT)	[proc]	Generates a release date and reel number for a given tape file code to be used by the operator. FC = File code of the tape to be saved, in Int. LNTH = Number of days, weeks, or months, in Int. UNIT = Units for LNTH: DA = days WK = weeks MO = months DATE1 = Real time clock substitute, in Hol 6. RELDT = Release date as printed for the operator, in Hol 9. YRXMONXDA YR = year X = space MON = month DA = day	N/A / 277	UNKN	5	REEL TIME TYPE

CALL TYPE	DESCRIPTION	ACCUR #WRDS	TIME MSEC	SEC	OTHERS USED
TYPE(DUM'MESS) [proc]	Outputs a 50 Hollerith item onto the typewriter, in FPS. DUM'MESS = An item containing the output message, in Hol 50.	N/A 160	UNKN	4	
TYPE'MSG(FC,CORE,WORDS, ER'FL) [proc]	Outputs up to 48 characters on the typewriter. FC = Typewriter file code, in Int. CORE = Table name containing the message to be outputted. WORDS = Number of words in the message of Int. ER'FL = File Code of a file to place I/O transmission errors on, in Int.	N/A 252	UNKN	5	CK'STATS
WRITE(DUM'FILE,DUM'TABL) [proc]	Transfers an entry from a serial table to an output file, in FPS. DUM'FILE = File description item, in Hol 50. See procedure write up for details. DUM'TABL = Table name of the serial table that the entry will be read from.	N/A 494+	UNKN	4	FDES'ENT I'O TYPE
WSEG(DUM'FILE) [proc]	Writes an end of segment mark on a file, in FPS. DUM'FILE = File description item, in Hol 50. See procedure write up for details.	N/A 35	UNKN	4	FDES'ENT I'O PWRIT TYPE
XBCD'OPC(CONVAL=CONVAL1) [proc]	Converts IBM external BCD code to OPCON. CONVAL = Input value to be converted, in Hol 8. CONVAL1 = Converted output value, in Hol 8.	N/A 73	UNKN	3	ALPHCON
ZZ(XX = BCDF) [proc]	Converts a Flt Pt Numb to an OPCON coded item 16 characters long. XX = Input value, in Flt Pt. BCDF = Output value, in Hol 16.	UNKN 275	UNKN		

1. PURPOSE

XBCD'OPC is a procedure that converts IBM external BCD to OPCON code.

2. CALL FORMAT

XBCD'OPC(CONVAL = CONVAL1)

WHERE:

XBCD'OPC	= the procedure name
CONVAL	= the input value to be converted, defined as an 8 character hollerith item
CONVAL1	= an 8 character hollerith item to which the output is assigned

3. LIMITATIONS AND ACCURACY

a. One 8-byte word (item) is converted at a time.

b. All characters are converted, regardless of legality. If illegal characters are present, blanks are generated.

c. This routine is system dependent and is not transferable.

4. PROCEDURE CHARACTERISTICS

a. Operating time is unknown, but will be the same for all input parameters.

b. Storage - 73 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

a. ALPHCON

6. PROGRAMMING TECHNIQUE USED

a. A for loop is used to check each byte.

b. Bit configurations are changed to equal IBM 7#94 code.

7. EXAMPLE

IF:

```
     Item CONVAL   U  8  $
     Item CONVAL1  U  8  $
```

AND:

```
                   CONVAL = 0(2022236151236020)  $
                   XBCD'OPC(CONVAL = CONVAL1)    $
```

THEN:

```
                   CONVAL1 = 0530310627314205
```

NOTE:

Characters other than alpha-numeric are not guaranteed. If a character is not found in the ALPHCON tables (see procedure ALPHCON), OPCON blanks are generated.

Input and output parameters can be defined as the same value.
Ex: XBCD'OPC (word = word) $

1. PURPOSE

CRDSTR is a procedure for prestoring card images from a 1604 computer to an open ended file.

2. CALL FORMAT

CRDSTR (FC, FWA, NUMB)

WHERE:

CRDSTR = the procedure name

FC = the file code for the file to which cards will be prestored, declared as a 48 bit unsigned integer

FWA = the first word address of the cards (or table containing the cards) to be prestored, declared as a 48 bit unsigned integer

NUMB = the number of cards to prestored, declared as a 48 bit unsigned integer

3. LIMITATIONS

a. User must include, in his Job Cards, a file card for each file that will be prestored to. Term Reason #401 will occur on an attempt to prestore to an undeclared file.

b. The number of cards to be prestored in a single procedure call is not tested for a maximum value.

c. The procedure does not check for valid OPCON characters in the card images being prestored.

4. PROCEDURE CHARACTERISTICS

a. The time required to operate this procedure depends on the accessibility of the Control 160 computer and the number of card images to be prestored.

b. Storage - 35 octal locations

c. This procedure is JOVIAL and Machine Code.

5. OTHER PROCEDURES CALLED

None

6. PROGRAMMING TECHNIQUE USED

Sends card images to Control 160 computer for prestore on the system card disk.

7. EXAMPLES

IF:

```
     Table Image        R    12Ø    S    I        $
       Begin
       Item Record      H     8Ø                  $
       End
     Item FC            I     48   U              $
     Item FWA           I     48   U              $
     Item NUMB          I     48   U    P    1    $
     File JOBA          V(n)    Ø4                $
AND:
                  FWA = 'LOC(IMAGE)                $
                  FC  = JOBA                       $

                  CRDSTR(FC, FWA, NUMB)            $

                            OR

                  CRDSTR(JOBA, 'LOC(RECORD), 1)    $

                            OR

                  CRDSTR(4, 0(21000), 1)           $
```

THEN:

All three of these procedure calls will cause one card image (RECORD) to be read from Table IMAGE to the open ended file, Ø4.

CRFF

1. PURPOSE

CRFF is a function for 1604 jobs that are interruptible (i.e., auto jobs). It senses for a carriage return (typed by operator) on the 1604 console typewriter and sets or clears a Boolean item accordingly.

2. CALL FORMAT

```
CRFF()
```

WHERE:

CRFF = the function name, defined as a Boolean item indicating:

1 = the operator has requested termination of the auto job

0 = the operator has not requested termination of the auto job

3. LIMITATION

None

4. PROCEDURE CHARACTERISTICS

a. The time required to operate this procedure varies from a minimum of 38 MSEC to a time dependent on the number of I/O operations in progress within the user program.

b. Storage - 7 words

c. This procedure is JOVIAL and machine code.

d. This procedure requires the user to define a Boolean item which is tested after the procedure has been called.

5. OTHER PROCEDURES CALLED

None

6. PROGRAMMING TECHNIQUE USED

Wait until channels 1 and 2 are inactive then test the carriage return status and set or clear the Boolean output parameter accordingly. If status is no, return. If status is yes, wait until all I/O operations in progress are completed, then return.

7. EXAMPLE

IF:

```
     Item CRS    B    $
```

AND:

```
                CRFF()    $

                If CRS
                  BEGIN
                  "User will terminate his auto job and
                   return control to the operator"
                  END
                "User may continue with his own program, control
                 has not been requested by the operator"
```

THEN:

The action described above will be taken.

INTORDUCTION TO THE SEARCH AND SORT PROCEDURE

1. The procedure grouped in this category are:

   a. BINSRCH
   b. DDSORT
   c. DLOAD
   d. SORT
   e. SRC

2. Of this set of procedures only SORT has an external sort capability. This procedure basically supplies the linkage for operating the SORT/MERGE utility. Documentation on this utility will be found in NAVCOSSACT Report No. 88M903A PM-01, 1604A Generalized SORT/MERGE Programs.

3. DLOAD, while not really a sort procedure, is included here because it does build a key table in core which may be readily sorted.

1. PURPOSE

BINSRCH is a procedure that performs a binary search on a single or multiple word per entry serial (or one word per entry parallel) sorted table.

2. CALL FORMAT

```
BINSRCH(TBLNAME, WDSENT, ETRYWRD, ARGTAB, ARGWDENT =
        SRCHENT, ALTADD.) $
```

WHERE:

BINSRCH = the procedure name.

TBLNAME = the name of the sorted serial table to be searched (or a sorted one word per entry parallel table.)

WDSENT = number of words per entry of the table to be searched, declared as a 15 bit unsigned integer. The form NWDSEN(TBLNAME) may be used.

ETRYWRD = the number of the word in the entry against which the search is desired to be carried out against, declared as a signed 48 bit integer. (Ø would be used for the Øth word, 1 for the 1st word, etc.)

ARGTAB = name of the table that contains the argument. For simple items the form 'LOC (simple item) may be used.

ARGWDENT = number of words per entry of the argument table. The form NWDSEN(ARGTAB) may be used.

SRCHENT = the entry at which a successful match was found or the entry at which the argument should be inserted if no valid match was found, declared as a signed 48 bit integer. In the latter case the alternate (ALTADD) exit must be used.

ALTADD = statement label to which control is transferred if a valid match is not found.

3. LIMITATIONS AND ACCURACY (cont)

a. The contents of the table to be searched must be sorted in ascending sequence so that numerics are high and the alphas are low.

b. The NENT of the table to be searched must be a true NENT, i.e., total number of words.

c. Accuracy should be complete.

4. PROCEDURE CHARACTERISTICS

a. No figures are available on the time required to perform a given search.

b. Storage - 145 machine words are required.

c. This is a pure JOVIAL procedure.

d. Procedure DDSORT may be used to sort the procedure in the appropriate sequence.

5. OTHER PROCEDURES CALLED

a. None

6. PROGRAMMING TECHNIQUES USED

a. The values to be searched are divided in half repeatedly while looking for a match.

7. EXAMPLE

IF:

```
Table HR'MO      R  13  S  $
Begin
  Item CUMHRMO   I  24  U  $
Begin
  0   744   1416   2160   2880   3624   4344
  5088   5832   6552   7296   8016   8760
End
Item  MO         I  24  U  P  3900  $
Item  ETRYWRD    I  48  S  P     0  $
Item  ETRY       I  48  S          $
```

AND:

```
        BINSRCH(HR'MO, NWDSEN(HR'MO), ETRYWRD, 'LOC(MO),
        1 = ETRY, ALT.)

        GOTO CONTNU          $
ALT.    GOTO EXIT            $
```

THEN:

ETRY = 6 and an exit will be made to statement label ALT where control is transferred to EXIT.

NOTE:

The value 3900 lies between 3624 and 4344, entries 5 and 6. If the value 3900 were inserted in the table, it would be in entry 6. Had MO been equal to 3624, then ETRY would equal 5 and normal control would be returned to the first statement after the procedure call, in this case the GOTO CONTNU $.

DDSORT

1. PURPOSE

DDSORT is a procedure that sorts (logically) in ascending order, entries of a single or multiple word per entry serial table.

2. CALL FORMAT

DDSORT(ADDR, NUMB, EWDS, SWDS, FWSK)

WHERE:

DDSORT = the procedure name.

ADDR = address of the table to be searched, declared as a 48 bit integer. The function 'LOC(table name) may be used.

NUMB = number of entries in the table to be sorted, declared as a 48 bit integer item. The function NENT(table name) may be used.

EWDS = number of words per entry in the table being sorted, declared as a 48 bit integer item. The function NWDSEN(table name) may be used.

SWDS = the number of full words in the sort key, declared as a 48 bit integer item.

FWSK = starting word of the table that the sort key is in, declared as a 48 bit integer item.

3. LIMITATIONS AND ACCURACY

a. The sort key may begin in any word, but must be a full word. If the key consists of multiple words, they must be in consecutive words.

b. Accuracy is complete.

4. PROCEDURE CHARACTERISTICS

a. Operating time is unknown.

b. Storage - 113 machine words are required.

c. This is a pure JOVIAL procedure.

4. PROCEDURE CHARACTERISTICS (cont)

   d. In sorting, the comparison is done only against the indicated key word or words of an entry. However, the whole entry is sorted, not just the key word.

5. OTHER PROCEDURES CALLED

   a. none

6. PROGRAMMING TECHNIQUES USED

   a. The SHELL sort technique is used to perform the sort.

7. EXAMPLE

```
IF:
    Table   SORTAB    R   1500   S   4   $
    Begin
      Item  DUMMY1    I     48   U   0   $
      Item  KEY1      H      8       1   $
      Item  KEY2      H      8       2   $
      Item  SORTKEY   H     16       1   $
      Item  DUMMY2    I     48   U   3   $
    End

AND:
    DDSORT('LOC(SORTAB), NENT(SORTAB), NWDSEN(SORTAB), 1, 1)  $

THEN:
    The table, SORTAB, will be sorted on the item KEY1.
```

1. PURPOSE

DLOAD is a procedure that reads unpacked records from tape or cards, selecting those meeting previously established criteria, it builds a sort key in core while packing the records on disc at maximum density.

2. CALL FORMAT

"A" DLOAD(TP, RDISC, HIST, FMS, RDBUF, TRK1, KEY, WPE,
LAST = NUMT, NUMR)

"B" If this and/or that and/or...something else
GOTO STATEMENTLABEL (a negative conditional transfer statement.)

"C" ENCODE' (Parameter List)

"D" STATEMENTLABEL.

WHERE:
"A"

DLOAD = the procedure name.

TP = file code of the input file, declared as a 15 bit unsigned integer.

RDISC = file code of the random disc output file, declared as a 15 bit, unsigned integer.

HIST = file code of the output history tape which lists all error message that might have been generated, declared as a 15 bit unsigned integer.

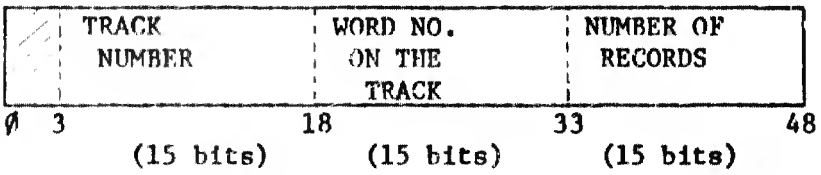
FMS = a boolean item indicating if the input tape is in FMS format or not according to:
0 = tape is not FMS formatted
1 = tape is FMS formatted

RDBUF = the name of the table into which the input data will be read. There will be n + 1 words per entry, where "n" = the number of words in a key, sufficient entries must be present to satisfy the selection criteria for all input records.

TRK1 = beginning track number to be used of the random disc output file, declared as a 15 bit unsigned integer.

2. CALL FORMAT (cont)

KEY = name of the table that will hold the generated keys. The first word, (0th), of an entry for this table of keys will take the following form:



All three values (Track Number, Word Number, and Number of Records) and declared as 15 bit, unsigned integer items.

WPE = words per entry in table KEY, declared as a 15 bit unsigned integer.

LAST = the address of the statement label that is given in "D" part of the call, declared as a 15 bit unsigned integer. The form 'LOC (statement label) may be used.

NUMT = number of tracks used, declared as a 15 bit unsigned integer.

"B"

A negative conditional transfer consisting of two statements, an "IF" and a "GOTO" of the type:

```
If a and not b or c $
    GOTO statementlabel $
```

where statementlabel is the "D" part of the call. The IF statement established the criteria upon which a record will be rejected.

"C"

An ENCODE' (parameters) that will contain the statements required to set up one key entry in the first n-1 words of the Ø entry of key table. This creates the sort key.

"D"

The STATEMENTLABEL is applied to the first statement that continues the main programs.

3. LIMITATIONS AND ACCURACY

a. The code generated by the negative conditional transfer statement and the ENCODE' statement may not exceed 50_{10} words.

b. Input record size may not exceed 350_{10} words.

c. Subscripting in the negative conditional transfer and ENCODE' statement is restricted to integer constants, simple items, items within the records being processed and combinations of them.

4. PROCEDURE CHARACTERISTICS

a. Operating time is unknown.

b. Storage 260 machine words are required.

c. This is not a pure JOVIAL procedure. It is highly system dependent and thus non-transferable.

d. Table RDBUF may be any table of appropriate size, the contents of which need not be saved during the operation of DLOAD.

e. There are four possible causes of abnormal termination. The term reason will be of the TERM REAS 37XX form. Where "XX" may be:

24 - NENT(key) exceeded
25 - Conditional and ENCODE' code exceeds 50_{10} words
26 - An input record was found containing more than 350 words
27 - User - defined disc area was exceeded

5. OTHER PROCEDURES CALLED

a. CK' STATS

6. PROGRAMMING TECHNIQUE USED

a. Code generatee for the user's negative conditional transfer statement "B" and ENCODE' statement "C" is moved to a DLOAD CASE and overlaid to a table. There, it is modified as necessary. Input and output files are opened. Records are read, and discarded if the selection criteria is not met. Those which "pass" are packed on disc at maximum density. For these latter records, keys are built in core, containing sort data (if provided for) and disc location information for later retrieval.

7. EXAMPLE

a. None.

1. PURPOSE

SORT is a procedure in the File Processing System (FPS) that sorts records in either ascending or descending order from an input file on tape to an output file on tape.

2. CALL FORMAT

```
SORT (FDIA, FDIB)
```

WHERE:

SORT = the procedure name.

FDIA = the item name of the file description item that describes the input format to SORT or the file description item that describes the output format from SORT, declared as a 50 hollerith item according to the format given below.

FDIB = the item name of the file description item that describes the output format from SORT or the file description item that describes the input format to SORT, declared as a 50 hollerith item according to the format given below.

Of the several formats available to the user in the FPS System, Format 3 must be used with this procedure. Detailed information describing this format may be found in Section 2, 3.3, 3.9 and 3.10 of the File Processing System Command Manual, NAVCOSSACT Document No. 88M902A CM-01A. Appendix B of this manual provides space for that manual.

Format 3 requires that each record be one entry, which can vary in length throughout the file.

FDIA is a File Description Item declared by the programmer as a 50 hollerith item in the following form. Though it may represent either the input or output file, the input file is assumed here.

ITEM itemname 5ØH(I n_1 R n_5) $

WHERE: I = an indicator to show that this item declaration is for an input file.

n_1 = the SSFC file code (1-3Ø) that is assigned to the input file.

R = an indicator that indicates that the next field will contain the record length.

2. CALL FORMAT (cont)

n_5 = the maximum number of words in any one record of the file.

FDIB similarly is a File Description Item declared by the programmer. In this case it is assumed to represent the output file. There are two different types of items for describing the output file format. One provides for sorting the file in ascending order, the other for sorting it in descending order.

The FDI for sorting entries in ascending order is:

ITEM itemname 5ØH(Ø n_1 R n_5 A f_1, l_1) $

The FDI for sorting entries in descending order is:

ITEM itemname 5ØH(Ø n_1 R n_5 D f_1, l_1) $

WHERE:

Ø = an indicator to show that this item declaration is for an output file.

n_1 = the SSEC File Code (1-3Ø) that is assigned to the output file.

R = an indicator that indicates that the next field will contain the record length.

n_5 = the maximum number of words in any one record of the file.

A = an indicator indicating that the output file will be built in ascending order.

D = an indicator indicating that the output file will be built in descending order.

$f_1 l_1$ = the location in each entry of the key item on which the entries of the file are to be ordered in either ascending or descending order. f_i is the number of an items first character, relative to character Ø of the entry. l_i is the length of the item given in characters. As many as six f_i, l_i may be present in the FDI declaration. The order in which they are listed determines the relative order - major to minor- of the key items.

3. LIMITATIONS AND ACCURACY

a. The key items being sorted must be OPCON coded or unsigned integers that are alone within bytes of a word.

b. The SORT/MERGE program (1604A Generalized Sort/Merge Programs, NAVCOSSACT Report No. 88M903A PM-01) must be available through File Code 26.

c. Four scratch files, tape or serial disc, are required on files 21, 22, 23 and 24. File 24 is used for saving and restoring core memory immediately before and after the execution of the SORT/MERGE program. The other three are used for intermediate storage during the sorting operation.

d. Only tape files may be used with this sort.

e. Accuracy is complete.

4. PROCEDURE CHARACTERISTICS

a. Operating time is unknown, but it will be a function of the number of entries to be sorted.

b. Storage - 362 machine words are required.

c. This is a pure JOVIAL procedure.

d. TERM REAS 3734 along with the typewritten message "NO SORT KEYS OR NO SORT OUTPUT FILE GIVEN", is given when the sort keys are not specified in the file description item for the output file.

e. Reference may be made to Section 3.3, 3.9, 3.10, 7.6, 16.2 and 18 of the File Processing System Command Manual, NAVCOSSACT Document No. 88M902A CM-01A for information on the use of this procedure.

5. OTHER PROCEDURES CALLED

a. FDES'ENT

b. SORT/MERGE program

6. PROGRAMMING TECHNIQUE USED

a. Sort provides the necessary linkage to utilize the SORT/MERGE program to do a tape to tape sort.

7. EXAMPLE

a. See Section 16.2 of the File Processing System Command Manual.

1. PURPOSE

SRC is a procedure that locates an identifier in a PPS-111 compool directory.

2. CALL FORMAT

SRC (NAME = XREF, XENT)

WHERE:

SRC = the procedure name.

NAME = an 8 character identifier to be located and declared as Hollerith 8. The identifier must be left justified with trailing blanks if less than 8 characters.

XREF = the value from item CXREF (definition cross reference) in the directory entry containing the identifier equal to NAME, declared as a 16 bit signed integer item.

XENT = number of the directory entry containing the identifier equal to NAME, declared as a 16 bit signed integer item.

3. LIMITATIONS AND ACCURACY

a. This search may be used against a Phase III compool only.

b. If the identifier is not found, both XREF and XENT will be set to zero.

c. The compool table (CMPL) and items CNAME, CXREF, and CNENT contained therein must be centrally defined.

d. Accuracy should be complete.

4. PROCEDURE CHARACTERISTICS

a. Operating time is unknown.

b. Storage - 46 machine words are required.

c. This is a pure JOVIAL procedure.

5. OTHER PROCEDURES CALLED

   a. none

6. PROGRAMMING TECHNIQUE USED

   a. Binary search technique is used.

7. EXAMPLE

   a. none

III. CONVERSION PROCEDURES (Continued)

15.	INTHI	A procedure that converts an unsigned integer item into an 8 byte OPCON item one word at a time.
16.	NORM360	A function that converts any angle, positive or negative, supplied in floating point degrees to its equivalent positive value between Ø and 36Ø degrees.
17.	OC2HO	A procedure that converts an unsigned octal integer to OPCON code.
18.	OPCTOB	A function that converts an OPCON coded number to its floating point equivalent.
19.	PCL*	A procedure that converts latitude and longitude in OPCON code to floating point radians.
20.	RADIANS	A function that converts floating point angular degrees to floating point radians.
21.	ST2HO	A procedure that converts up to 8 characters of a standard transmission coded item to OPCON code.
22.	TYPE*	A procedure that converts the 50 Hollerith character item designated in the call into its typewriter TYPE code equivalent to be outputted at the console typewriter.
23.	TYPE'MSG*	A procedure that converts 1 to 6 words of OPCON data to its typewriter TYPE code equivalent, and type the result on the console typewriter.
24.	XBCD'OPC	A procedure that converts IBM external BCD code to OPCON code.
25.	ZZ	A procedure that converts a floating point number to an OPCON coded item 16 bytes long.

APPENDIX A

IV. FILE PROCESSING SYSTEM PROCEDURES

1.	DELET	A procedure that reallocates core buffer area due to the opening of a new file or the closing of an existing file in FPS.
2.	ENTER	A procedure that opens the file designated in the File Description Item and allocates an I/O work area in core memory to be used for the file being opened in FPS.
3.	ENTERF	A procedure that opens an IPS master for use with FPS library procedures.
4.	ENY	A procedure that determines the current entry position of any file in FPS.
5.	EOF	A procedure that tests for End-of-File in FPS.
6.	EOR	A procedure that tests for End-of-Record in FPS.
7.	EOS	A procedure that tests for End-of-Segment in FPS.
8.	EXIT	A procedure that closes the processing of a file in FPS.
9.	FDES'ENT	A procedure that obtains the entry number in Table FDES for the file designated in the procedure call in FPS.
10.	I'O	A procedure that handles all peripheral device manipulations and actual data transmissions for the procedures of FPS.
11.	LADDL	A procedure that obtains the address plus 100 of the last memory word loaded by the SSEC loader in FPS.
12.	LOST	A procedure that logs the count of unrecoverable parity errors from each FPS file for which the I/O error parameter was specified.

APPENDIX A

IV. FILE PROCESSING SYSTEM PROCEDURES (cont)

13.	READ	A procedure that transfers the next entry of an input file to the first entry of a special table in FPS.
14.	READF	A proceudre that reads a record from an IPS format tape, converts it to FPS format 4, and transfer a set at-a-time to the user program.
15.	REC	A procedure that determines the current record of any file in FPS.
16.	RELEASE	A procedure that is used in conjunction with READ to provide a means of skipping over any remaining entries in a record in FPS.
17.	RWRIT	A procedure that writes the contents of a file's I/O work area onto tape in FPS.
18.	SEG	A procedure that determines the current segment of any file in FPS.
19.	SORT*	A procedure that sorts the records of the input file into either ascending or descending order with respect to key item values in each record in FPS.
20.	TYPE*	A procedure that converts the 50 Hollerith character item designated in the call into its typewriter character equivalent to be outputted at the console in FPS.
21.	WRITE	A procedure that transfers either an entry or a print line image from the first entry of a serial table to a magnetic tape file in FPS.
22.	WSEG	A procedure that writes an End-of-Segment mark in a file in FPS.

APPENDIX A

V. PROGRAM MAINTENANCE AND UTILITY TYPE PROCEDURES

1.	CK'STATS	A procedure that waits for a "file not busy" status. Checks for other non-normal stati, and take appropriate action.
2.	CLEAR	A procedure that clears an area of core, defined by first word address and number of words, to +0 or blanks.
3.	CORE	A procedure that dumps a specified memory area onto file 27 whenever entered.
4.	CRDSTR	A procedure that prestores card images from a 1604 computer to an open ended file.
5.	CRFF	A procedure used with 1604 interruptable (auto) jobs. It senses for a carriage return (typed by the operator) on the 1604 console typewriter and advises the user of an auto job whether his program is to be terminated and control returned to the operator.
6.	CSCAN	A procedure that locates and classifies fields within an unpacked (one character per word) image table that generally represents card columns.
7.	DUMP	A procedure that initiates a dump service request to SSEC for that portion of memory requested in the call.
8.	GET'CARD	A procedure that reads cards (or card images), double buffered, and unpack them as required by CSCAN and the 'nn2nn routines.
9.	INTERRUPT	A procedure that initiates a request to interrogate the interrupt status for internal (arithmetic) faults.
10.	REEL	A function that queries the TINF table and returns the reel number assigned to a given magnetic tape file.
11.	RESTART	A procedure that initiates a request to SSEC to establish restart points.

APPENDIX A

V. PROGRAM MAINTENANCE AND UTILITY TYPE PROCEDURES (cont)

12.	SECURITY	A procedure that initiates an SSEC request to determine the security classification of a file.
13.	SNOCTAL	A procedure that provides the capability of utilizing an unlimited number of octal correction. Core dump cards per computer run.
14.	STAT'CK	A procedure that produces a typewriter comment for all data transmissions resulting in other than a normal or EOF status.
15.	STORE	A procedure that blocks "print" lines, 23 lines per block, and write the lines double buffered. Automatically insert page headings, classifications pagination as needed.
16.	TIME	A procedure that initiates a request to SSEC for the real time.
17.	TPRELSE	A procedure that generates a typewriter message to the operator directing him to save a magnetic tape reel to some given date.
18.	TYPE	A procedure that converts a 50 Hollerith character item to its typewriter character code equivalent for outputting on the console typewriter.
19.	TYPE'MSG*	A procedure that converts 1 to 6 words of OPCON data to its TYPE code equivalent, and type the result on the console typewriter. The original message is untouched.

APPENDIX A

VIII. SEARCH AND SORT PROCEDURES

1. BINSRCH	A procedure that performs a binary search, making use of a single or multiple word key, on a sorted serial table containing one or more words per entry.
2. DDSORT	A procedure that will sort (logically) in ascending order, entries of a single or multiple word per entry serial table via a key field in the entry.
3. DLOAD	A procedure that will read unpacked tape records from tape or cards. Select those meeting preestablished criteria and build a predetermined sort key in core. The records are packed on disc at maximum density.
4. SORT*	Sorts the records of an input file into either ascending or descending order with respect to key item values in each record in FPS.
5. SRC	Locates an identifier in a compool (PPS-111) directory using the binary search technique.

APPENDIX A

* Indicates that this procedure has been crossed referenced under another subject category. For example procedure PCL will be found described under the heading of CONVERSION PROCEDURES and NAVIGATIONAL PROCEDURES.