FST-1

SUBROUTINE LIBRARY MANUAL

TABLE OF CONTENTS

PAGE

SECTION IV FILE PROCESSING PROCEDURES (Continued)

FIGURES

TABLES

SUBROUTINE LIBRARY MANUAL

1.0 INTRODUCTION

This manual describes those systems procedures that are available to users by means of the CALL directive. These routines are automatically loaded with the user's program if he references them. Some aspects of these procedures are not discussed in great detail. In general, great caution should be exercised in getting too intimate with the system since the risk in destroying the system usually outweighs the advantages of trickery. In discussing disc files, familiarity with the DOPSY manual is assumed.

2.0 IOCS PROCEDURES

2.1 INTRODUCTION

All of the I/O routines are very similar with regard to calling sequence and usage. The similarities will be discussed in this section; subsequent sections deal with the particulars for each I/O routine.

An I/O procedure acknowledges two types of calls. The first of these is used to initiate an operation on a device. Its general form is:

```
CALL    ioname
DATA    integer
DATA    <dcb address/integer>
(DATA   name)
```

The name of the I/O procedure occurs in the operand of the CALL statement. The DATA statement immediately following the CALL specifies the operation to be performed. These values and the operations assigned to them are described in Table 2.1.

If the operation is one involving a data transfer, the second location after the CALL will contain the address of the data control block, DCB, where the following information is stored. The first word of the DCB is the number of words to be transferred, the second is the core memory address of the first location to be read/written; the third word of the DCB is required only for disc transfers and is the disc address (in segments) of where the data is to be read or written.

Example:

```
        DATA    48, *+2, 80
        BSS     48
```

TABLE 2.1

SUBROUTINE CALLING SEQUENCE OPERATION CODES

		SUBROUTINES					
TYPE OF CODE	OCTAL CODE	TTRIO	TTPIO	CRIO	DISCIO	LPIO	MTIO
Test	0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY
Read	1	READ TTK		READ BINARY	READ BINARY		READ BINARY
Read	2	READ TTR		READ ALPHA			
Write	3		PRINT BINARY cr-lf*		WRITE BINARY	PRINT WITHOUT LINE FEED	WRITE BINARY
Write	4		PRINT BCD cr-lf			PRINT WITH LINE FEED	WRITE TAPE MARK
Motion	5					SPACE N LINES	RECORD SKIP FORWARD
Motion	6						RECORD SKIP BACK
Motion	7					TOP OF FORM	REWIND
Motion	10						FILE SKIP FORWARD
Motion	11						FILE SKIP BACK
	12						
Write	13		PRINT BINARY no cr-lf				
Write	14		PRINT BCD no cr-lf				

* cr-lf is carriage return - line feed

This DCB can be used for a 48 word transfer between core memory and track one, sector zero on the disc. The DCB and its corresponding buffer area should not be altered until the I/O operation has been successfully completed; this is true of all I/O operations.

For some operations not involving a data transfer, the entry at CALL+2 will contain a count. The SPACE operation of LPIO, for example, uses this count to determine how far to space.

The entry at CALL+3 is not used by all I/O procedures, but when used it must contain the address of a user error routine. This error routine will be entered when either a recoverable error persists after ten attempts to correct it or an error in the DCB has been detected. This error routine is treated as an extension of the I/O interrupt routine and must return by executing a BRU* to its entry point. The A register will have the following format when the error routine is entered:

Bit	Description
19	On if DCB error.
20	On if data overflow.
21	On if parity or validity error.
22	On if end-of-file (EOF).
6-0	Device address.

More than one of the bits 22-19 may be on at a time.

Some general comments on these error conditions:

- DCB errors either result from the memory address exceeding the core available or from an excessive word count.
- Data overflow results when a device needs a memory cycle to empty/fill a buffer and, because of other memory demands, cannot get one.
- Parity/validity error indicates a data transmission error, illegal card codes, etc.

When control returns to the I/O interrupt routine, the operation will be accepted as correct if the A register is non-zero; otherwise, it will be tried again.

As they are currently implemented, the I/O routines, except those for the teletype, use locations 75B-77B as a pre-operative error routine. An illegal operation value or device not ready will cause a halt at 76B to be executed. This can be readily identified by the fact that the program counter is 100B and the A register contains the device number in bits 6-0.

The I/O procedures operate with the interrupt system and automatically overlap I/O with program execution. In order for the user to take advantage of this, a special type of call is provided. Its general format is:

```
CALL     ioname
DATA     0
---            BUSY RETURN
---            NOT BUSY RETURN
```

This "operation" tests to see if the I/O routine has completed processing the previous non-test operation. If the I/O routine is busy, control will return to CALL+2; if it is idle, (data transfer complete), control will return to CALL+3.

No provision is made in the I/O procedure for handling reentrancy. The user should, therefore, be very careful about calling I/O routines from interrupt processors. When an I/O routine is called, it will save and restore any index registers that it requires, but will not save or restore the A and E registers. The interrupt routines, obviously, are not quite so reckless.

If an I/O routine is CALLed by a user program, care must be taken to insure that in case of any software error, control is returned to the Automatic Restart Routine (ARR, location 125B) so that all interrupt entrance locations will be relinked with the proper system routine.

In the following sections the details of each I/O routine are presented. Since the "test" operation is the same for all such routines, further discussion is not required and is, therefore, omitted.

2.2 TTPIO

Purpose: To output a record to the teletype.

Calling Sequence:

```
CALL     TTPIO
DATA     operation
DATA     dcb
---              NORMAL RETURN

operation - 3    BINARY print, with CR/LF.
            4    BCD print, with CR/LF.
           13(8) BINARY print, without CR/LF.
           14(8) BCD print, without CR/LF.
```

Description:

TTPIO will output to the teletype the contents of the buffer described by the DCB. The buffer is assumed to contain four TASCII characters per word; see Appendices A and B. TTPIO will output a carriage return and line feed after the last character is printed only if bit 3 of the operation is not set. Because of the 72-character limit to a teletype line, the word count must be less than 19 or the last characters will be truncated. In the BINARY mode, every character in the buffer will be printed. In the BCD mode, trailing blanks will not be printed.

Because the teleprinter is also shared with TTRIO, TTPIO sets a flag in the COMREC so that no keyboard input can be initiated while an output operation is being performed.

2.3 TTRIO

Purpose: To input a record from the teletype keyboard or paper tape reader.

Calling Sequence:

```
CALL      TTRIO
DATA      operation
DATA      dcb
---                   NORMAL RETURN

operation - 1         READ keyboard
            2         READ paper tape
```

Description:

TTRIO will input into the specified buffer until the buffer is full or until a carriage return is encountered; in the latter case, the buffer will be padded with spaces. The TASCII characters are placed in the buffer four per word; see Appendices A and B. When TTRIO is ready for keyboard input, it will output the character obtained from the high-order six bits of the operation entry. If unspecified, it will be a space. This character can be used to uniquely identify the source of the input request, i.e., the monitor's *, etc.

All characters read from the keyboard will be echoed, i.e., sent to the teleprinter; like paper tape input, however, only the printing characters, the TASCII set, are placed in the buffer. Two of the control characters are used by TTRIO to provide limited editing. The characters produced by CTRL B and CTRL L are used to indicate BACKSPACE and LINE DELETE, respectively.

CTRL B will cause the buffer character pointer to be backed up one character position. This is indicated by echoing a '←' if the input is from the keyboard.

Example:

```
//  RENS←AME 'TEST1←2' AS 'TEST3'
//  RENAME 'TEST2' AS 'TEST3'
```

CTRL L will cause the buffer character pointer to be set to zero. This is indicated by echoing carriage return, line feed, and the input request character, if the input is from the keyboard. The same net result, emptying the input buffer, could be obtained by an appropriate number of backspace characters.

Example:

```
*        STA  TABLE+3  <CTRL L>
*DP3J1   STA  TABLE+3
```

2.4 CRIO

Purpose: To input a record from the card reader.

Calling Sequence:

```
        CALL     CRIO
        DATA     operation
        DATA     dcb
        DATA     error
        ---                NORMAL RETURN

        operation - 1      BINARY READ
                    2      BCD READ
```

Description:

CRIO will read a card into the specified buffer. The two types of read, BINARY and BCD, produce twelve and six bits per card column, respectively; the format of the resulting buffer contents is discussed in Appendix B. The maximum word count one can use without producing a DCB error is 20 for BCD mode and 40 for BINARY.

The six bit code produced by the BCD read is not the TASCII code expected by the system, but can be converted to TASCII by the procedure CRASC. Table 2.4 shows the six-bit encoding for the card code produced by the 029 keypunch. This table shows that there are six card codes whose graphic characters do not correspond to any in the TASCII set; the handling of these is discussed in CRASC.

CRIO requires manual intervention on card jams and validity errors and will retry the read when the required intervention has occurred. The validity check occurs when an illegal card code is read; this can happen only in the BCD mode. The BCD character set can be used to produce all 64 possible combinations in the BCD mode.

The user error routine is entered either as a result of DCB errors, data overflow, or an EOF condition. The EOF condition occurs when the card reader goes not ready and the output stacker is full or the input stacker is empty. In either case, the reading of the last card cannot be successful until the card ready goes READY again. The normal mode of operation is to ignore this condition and let the program detect a '//' record for an end-of-file. This record should be followed by a dummy one if it is the last record in the input stacker.

2.5 LPIO

Purpose: To output a record to the line printer.

TABLE 2.4

FST-1 INTERNAL CODES

Char.	TASCII Code	ASCII	029 Code Graphic	Char.	TASCII Code	ASCII	029 Code Graphic
Space	00	240		@	40	300	
!	01	041		A	41	101	
"	02	042		B	42	102	
#	03	243		C	43	303	
$	04	044		D	44	104	
%	05	245		E	45	305	
&	06	246		F	46	306	
'	07	047		G	47	107	
(	10	050		H	50	110	
)	11	251		I	51	311	
*	12	252		J	52	312	
+	13	053		K	53	113	
,	14	254		L	54	314	
-	15	055		M	55	115	
.	16	056		N	56	116	
/	17	257		O	57	317	
0	20	060		P	60	120	
1	21	261		Q	61	321	
2	22	262		R	62	322	
3	23	063		S	63	123	
4	24	264		T	64	324	
5	25	065		U	65	125	
6	26	066		V	66	126	
7	27	267		W	67	327	
8	30	270		X	70	330	
9	31	071		Y	71	131	
:	32	072	0-8-2	Z	72	132	
;	33	273		[	73	333	<
<	34	074	12-0	\	74	134	¬
=	35	275		]	75	335	>
>	36	276	11-0	↑	76	336	|
?	37	077		←	77	137	_
Carriage Return		215		Delete		377	
Line Feed		012					
Bell		207					

Calling Sequence:

```
a)  CALL     LPIO
    DATA     operation
    DATA     dcb/space count
    ---                          NORMAL RETURN

    operation - 3                PRINT
                4                PRINT FORMAT MODE
                5                SPACE N LINES

b)  CALL     LPIO
    DATA     operation
    ---                          NORMAL RETURN

    operation - 7                TOP OF FORM
```

Description:

LPIO will transmit the contents of the specified buffer to the line printer or position the printer paper in a particular way. The buffer is assumed to contain four six-bit TASCII characters per word, see Appendix B.

The line printer operates in two modes, format mode and normal mode. In the format mode, all print and space commands do not consider the space between bottom of form (BOF) and top of form (TOF) as part of the page. That is, the page perforation is skipped automatically. In the normal mode, the region between BOF and TOF can be used for printed output (i.e., where an end of page discontinuity is not desired).

The print commands reqjire a DCB address at CALL+2. The DCB word count should not exceed 33, reflecting the maximum (132 character) line printer print span. If this should occur, characters in excess of 132 will not be printed. In the case of the 80 column printer, if the DCB word count exceeds 20, then 60 characters will be printed in columns 1-60 of the first of a two line pair and the remainder right justified on the second line. For space commands, CALL+2 contains the number of lines to space; only the low-order seven bits are used.

2.6 DISCIO

Purpose: To transmit data between the disc and core memory.

Calling Sequence:

```
CALL     DISCIO
DATA     operation
DATA     dcb
DATA     error
---                  NORMAL RETURN
```

Calling Sequence:

```
a)  CALL     LPIO
    DATA     operation
    DATA     dcb/space count
    ---                          NORMAL RETURN

    operation - 3                PRINT
                4                PRINT FORMAT MODE
                5                SPACE N LINES

b)  CALL     LPIO
    DATA     operation
    ---                          NORMAL RETURN

    operation - 7                TOP OF FORM
```

Description:

LPIO will transmit the contents of the specified buffer to the line printer or position the printer paper in a particular way. The buffer is assumed to contain four six-bit TASCII characters per word, see Appendix B.

The line printer operates in two modes, format mode and normal mode. In the format mode, all print and space commands do not consider the space between bottom of form (BOF) and top of form (TOF) as part of the page. That is, the page perforation is skipped automatically. In the normal mode, the region between BOF and TOF can be used for printed output (i.e., where an end of page discontinuity is not desired).

The print commands reqjire a DCB address at CALL+2. The DCB word count should not exceed 33, reflecting the maximum (132 character) line printer print span. If this should occur, characters in excess of 132 will not be printed. In the case of the 80 column printer, if the DCB word count exceeds 20, then 60 characters will be printed in columns 1-60 of the first of a two line pair and the remainder right justified on the second line. For space commands, CALL+2 contains the number of lines to space; only the low-order seven bits are used.

2.6 DISCIO

Purpose: To transmit data between the disc and core memory.

Calling Sequence:

```
CALL     DISCIO
DATA     operation
DATA     dcb
DATA     error
---                NORMAL RETURN
```

```
operation - 1     BINARY READ
            2     PARITY CHECK
            3     BINARY WRITE
```

Description:

DISCIO will transmit blocks of data between disc and core memory. The maximum size of a block is 16,384 words. Every write operation performed by DISCIO is automatically followed by a parity check to assure that parity was generated properly.

The third word of the DCB required by DISCIO is the disc address of the disc area to be read or written. For this purpose, the disc is treated as a magnetic tape with 16,000 - 48 word records (sectors). The disc address is a binary value in the range of '0' to '15999' of the first such record (sector) involved in the transfer.

2.7 MTIO

Purpose: To transmit data between core memory and magnetic tape.

Calling Sequence:

```
CALL     MTIO
DATA     operation
DATA     dcb or count
DATA     error return
---                    NORMAL RETURN

operation - 0          BUSY TEST
            1          READ
            2          INVALID OP CODE
            3          WRITE
            4          WRITE TAPE MARK
            5          RECORD SKIP FORWARD
            6          RECORD SKIP BACKWARD
            7          REWIND
            8          FILE SKIP FORWARD
            9          FILE SKIP BACKWARD
```

MTIO transmits blocks of data between magnetic tape and core memory, performs error analysis, and executes miscellaneous commands to cause tape movements and writing of tape marks. The minimum block size is six (6) words and the maximum is 16, 384 words.

If the operation specified is either a record skip or file skip, CALL+2 should specify the number of records or files to be skipped.

The DCB used by MTIO is a standard 2-word DCB containing word count and core buffer address in that order.

3.0 CONVERSION PROCEDURES

3.1 INTRODUCTION

This section describes the conversion procedures that are available for translating from one character set to another or one number base to another.

3.2 CRASC

Purpose: To convert to TASCII the six-bit character code produced by the card reader.

Calling Sequence:

```
CALL    CRASC
DATA    dcb
---           NORMAL RETURN
```

Description:

The buffer is assumed to be of the format produced by a BCD read in CRIO. Each six-bit character is replaced by its TASCII counterpart. Registers affected: A, E.

3.3 ASCBIN

Purpose: To convert six-bit TASCII characters to 12 bit column-binary characters.

Calling Sequence:

```
CALL    ASCBIN
DATA    dcb
---           NORMAL RETURN
```

Description:

ASCBIN is used primarily for producing BCD card output. The number of words converted is determined from the word count entry in the DCB. Since each six-bit character is replaced by twelve-bits, the buffer area must be at least twice as large as indicated by the DCB. At the completion of the operation, the user's DCB word count will be doubled to reflect the increased size.

REGISTERS AFFECTED: A, E

3.4 BINDEC

Purpose: To convert a binary number to decimal.

Calling Sequence:

```
        LDA     binary
        CALL    BINDEC
        ---                 NORMAL RETURN
```

Description:

BINDEC will return six four-bit characters in the A register. These characters provide the decimal equivalent of the binary value. Note that if the binary value exceeds 999,999 the conversion will be incorrect.

Registers Affected: A,E

4.0 FILE PROCESSING ROUTINES

4.1 INTRODUCTION

The files residing on the disc are called Peripheral Memory Files (PMF). This section deals with the procedures that are available for processing these files. Some of these procedures are necessarily involved with housekeeping, but most are involved with input/output on the files. In the latter group are procedures for doing word I/O either sequentially or randomly and record or character I/O sequentially. These are discussed in detail in subsequent sections.

All of the procedures discussed in this section are associated with a PMF header. This PMF header contains enough information to permit an arbitrarily large disc file to be processed in pieces as small as 48 words, one sector. The PMF header is nine words in length; its format is described in Table 4.1.

In the discussion that follows, all pointers/addresses use '0' origin referencing; the first word/character has an address of '0', the second '1', etc.

The entries FS and FL are used to define that portion of the disc addressable by the file I/O procedures. For output files, this will be the entire space allocated to the file. For input files, it is only that portion of the file that has been written. FS is a word address relative to the beginning of the disc and FL is the number of words that can be referenced.

At any point in time, a certain portion of the file will be present in main memory. This section of the file is called the WINDOW and is defined by WS and WL. WS is a word address relative to FS and WL is the number of words currently in main memory. WL generally is equal to 48*PL; the only time this is not true is when 48*PL would force the WINDOW to include part of the next file.

WORD	BITS	DESCRIPTION
1	23-0	CP - Current Pointer
2	23-0	WS - Window Start
3	23-0	WL - Window Length
4	23-0	FS - File Start
5	23-0	FL - File Length
6	23-0	PS - Page Start
7	23-0	PL - Page Length
8	23-0	First 4 characters of file name
9	23-12	Last 2 characters of file name
9	3	I/O Flag 1 = Input 0 = Output
9	2-1	File Type 0 = STRING 1 = DATA 2 = OBJECT 3 = CORE IMAGE
9	0	Modify flag. Set if wondow contents are altered.

PMF HEADER FORMAT

TABLE 4.1

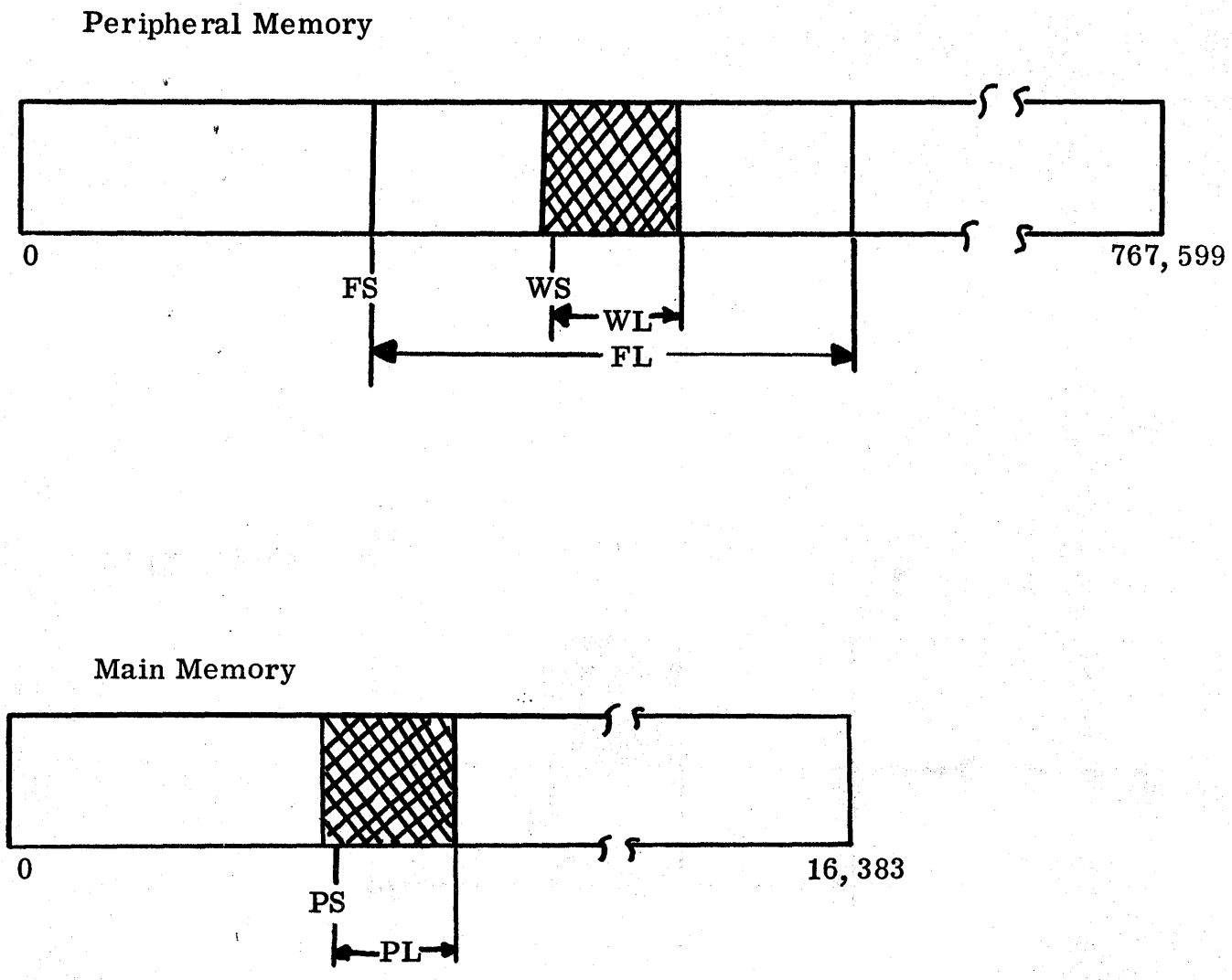

Peripheral Memory
0
767, 599
FS
WS
WL
FL
Main Memory
0
16, 383
PS
PL

Figure 4.1

The area of main memory that the file is segmented into is defined by PS and PL. PS is the main memory address of the buffer area and PL is the number of sectors available for this buffer. PS and PL must be assigned values by the user; PMFH entries in words 2-5 are initialized and maintained by the file procedures.

Table 4.1 illustrates the PMF header entries. The last two words of the PMF header contain the name of the file that is being referenced and some flags required for housekeeping. Working storage is 'treated' as a disc file and has the special name ' ' (four blanks inside the quotes). The use of the flags in bits 3, 2, and 1 or word 9 is described in more detail in the routines OPEN and CLOSE. The flag in bit 0 is set whenever the contents of the WINDOW are altered; this will always cause the current WINDOW to be written back to the disc before reading in a new one.

The remaining entry, CP, is a word/character address relative to FS of the next word/character to be affected by the sequential I/O procedures. Due to the dual interpretation of the CP, it is not advisable to do both word and character I/O on a file at the same time.

The procedures concerned with character I/O, viz: SCAN, GET and PUT, can be used for character processing on string buffers that are completely contained in main memory and not associated with a disc file. In order to do this, it is necessary for the user to initialize the entries of a dummy PMF so that the WINDOW completely encompasses the entire buffer (file). In particular, WS=0, FL and WL are set to the length of the buffer and FS and PS reference the first location in the buffer. CP should be set to the first character position to be affected; the remaining entries should be set to zero.

To see how this works, assume that there is a 20 word buffer into which a card has been read. To retrieve characters from this buffer one at a time in sequence from column one to eighty, procedure GET could be used. The following assembly language statements would define the PMF header and buffer.

```
PMFHEADR    DATA    0,0,20,BUFFER,20,BUFFER,0,0,0
BUFFER      BSS     20
```

Whenever a new card is read, the CP would need to be reset to zero. Setting the entries in this fashion forces the character processing routines to produce an EOF return whenever they address beyond the WINDOW, i.e., BUFFER. This prevents them from doing any disc operations. If WL<FL, a disc operation will be caused if the associated WINDOW is ever altered.

In the descriptions that follow, the word "pmfheader" is assumed to be the label on the CP; i.e., index register 7 contains the address of the PMF header.

4.2 OPEN

Purpose: To initialize a PMF header for processing a disc file.

Calling Sequence:

```
        LDX     7,pmfheader
        LDA     openflag
        CALL    OPEN
        ---                 ERROR RETURN
        ---                 NORMAL RETURN


openflag 0          OUTPUT
         1          INPUT
```

Description:

The only function performed by OPEN is filling in the values of CP, FS, FL, WS, WL and the flags so that the associated file may be referenced properly. FL will be set to reference the entire space allocated to the file if 'openflag' is an '0', otherwise, it is set to address only that portion of the file previously written. CP and WS are set to '0', WL to a '-1', and the flags, except for the I/O bit, are set to '0'. The I/O bit takes on the value of 'openflag' so that CLOSE can determine what action must be taken when the user is through processing the file.

The entries PS and PL must be initialized by the user. See the introduction, Section 4.1, for a description of these entries.

The normal return is taken if the file was opened successfully. The value returned in the A register is the value of the CP the last time the file was closed as an output file, i.e., the next available slot in the file. This value can be used to append new information to an old file by opening the file as an output file and storing the A register into the CP entry. Subsequent sequential output operations will continue from the end of the old file.

The error return is taken if the file cannot be located in the file directory.

REGISTERS AFFECTED: Index Register 6, A, E

4.3 CLOSE

Purpose: To terminate processing of a disc file.

Calling Sequence:

```
        LDX     7,pmfheader
        CALL    CLOSE
        ---                 ERROR RETURN
        ---                 NORMAL RETURN
```

Description:

CLOSE should be called when a file is opened for output and may be called when the file is opened for input. In either case, the first function performed by CLOSE is to write the WINDOW back to disc if it has been altered, since the altering of the WINDOW contents is independent of how the file was opened.

If the file has been opened as an output file, the directory entry for the file will be updated to reflect its new size and type. The type is determined from the file type field in the PMF header flags and the size of the file is determined from the CP, which is interpreted as a character count if the file type is STRING and a word count if the type is anything else. The file type is assumed to be STRING when a file is opened and is changed to type DATA by the PUTW procedure, so that output files of type STRING or DATA take care of themselves if the sequential I/O procedures are used.

The error return is taken if the directory entry for an output file cannot be located.

REGISTERS AND STATE SWITCHES AFFECTED: A, E
Index Register 6 if output file is closed.
State Switch 9

4.4 READ

Purpose: To obtain the contents of a specified PMF location.

Calling Sequence:

```
        LDX     7,pmfheader
        LDA     pmfaddress
        CALL    READ
        ---                 EOF RETURN
        ---                 NORMAL RETURN
```

Description:

If $0 \leq$ pmfaddress < FL,READ will return in the A register the contents of the PMF location specified by 'pmfaddress'. If the address is not in the allowable range, the EOF return is taken.

READ and WRITE are the basic procedures used directly or indirectly by all other file processing procedures. READ and WRITE call a common subprocedure ADRXLATE that uses DISCIO to read in new pages. Altered pages are written back to disc by means of the subprocedure SWAPOUT. Both of these subprocedures halt in the disc error routine when DISCIO cannot perform the required operation successfully. Pressing start allows the operation to be retried another ten times.

REGISTERS AFFECTED: A
E on normal return

4.5 WRITE

Purpose: To store a value into a specified PMF location.

Calling Sequence:

```
          LDX     7,pmfheader
          LDA     pmfaddress
          LDE     value
          CALL    WRITE
          ---               EOF RETURN
          ---               NORMAL RETURN
```

Description:

If 0 ≤ pmfaddress < FL,WRITE will store the contents of the E register into the PMF location specified by 'pmfaddress'. If the address is not in the allowable range the EOF return is taken.

See READ (section 4.4) for comments on disc usage.

REGISTERS AFFECTED: A, E

4.6 GETW

Purpose: To obtain the contents of the current word from a PMF.

Calling Sequence:

```
          LDX     7,pmfheader
          CALL    GETW
          ---               EOF RETURN
          ---               NORMAL RETURN
```

Description:

If 0 ≤ CP<FL,GETW will use CP as the PMF address and perform the same function as READ. In addition it will increment CP by one so the next call for GETW will obtain the next word. If CP is out of the allowable range, the EOF return is taken.

REGISTERS AFFECTED: A
E on normal return

4.7 PUTW

Purpose: To replace the contents of the current word in a PMF.

Calling Sequence:

```
        LDX     7,pmfheader
        LDA     value
        CALL    PUTW
        ---                 EOF RETURN
        ---                 NORMAL RETURN
```

Description:

If $0 \leq CP < FL$, PUTW will use CP as the PMF address and perform the same function as WRITE. In addition, CP is advanced by one so that the next call for PUTW will store into the next word. If CP is out of the allowable range, the EOF return is taken.

REGISTERS AFFECTED: A, E

4.8 GET

Purpose: To obtain the current character from a PMF.

Calling Sequence:

```
        LDX     7,pmfheader
        CALL    GET
        ---                 LETTER RETURN
        ---                 DIGIT RETURN
        ---                 OTHER
```

Description:

GET interprets CP as a character address. If $0 \leq CP/4 < FL$, GET will return in the low order portion of the A register the character addressed by CP. CP will also be advanced by one to make the following character the current one.

The letter return is taken if the character is one of the characters $, A, B, C, ..., Z. The digit return is taken for any of the characters 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Control returns to CALL+3 for anything else with an EOF indicated by the value 177B.

REGISTERS AFFECTED: A, E

4.9 PUT

Purpose: To replace the contents of the current character in a PMF.

Calling Sequence:

```
        LDX     7,pmfheader
        LDA     character
        CALL    PUT
        ---             EOF RETURN
        ---             NORMAL RETURN
```

Description:

PUT interprets CP as a character address. If $0 \leq CP/4 < FL$,PUT will store the character in the A register into the character position addressed by CP. CP is then advanced by one to make the next character the current one. If CP is out of the allowable range, the EOF return is taken.

Registers Affected: A,E

4.10 SCAN

Purpose: To obtain the next syntactical entity from a PMF.

Calling Sequence:

```
        LDX     7,pmfheader
        CALL    SCAN
        ---             IDENTIFIER RETURN
        ---             NUMBER RETURN
        ---             STRING RETURN
        ---             CHARACTER RETURN
```

Description:

SCAN uses GET to obtain characters from the PMF to form identifiers, strings and numbers.

An identifier is a sequence of letters (including $) or digits, the first of which must be a letter. Only the first six characters of such sequences are retained and they are returned left justified in the A and E registers to CALL+1.

Examples:

```
        NAME1
        $TEST
        A
        B2
```

A number is a sequence of digits; if the terminating character is a 'B' the base is assumed to be octal, otherwise, decimal. The low order 24 bits are returned to the E register to CALL+2. The terminating character, e.g., blank or comma, will be returned in the A register.

Examples:

```
10
77B     Equivalent to 63
16000
```

A string is a sequence of characters enclosed in single quotes. Like identifiers, only the first six characters are retained. These are returned left justified in the A and E registers to CALL+3.

Examples:

```
'A-B #'
'A'
'   A'
```

The address of the location containing the terminating character for these entities is returned in index register 7.

Single character operators/terminators are returned in the low order portion of the A register. In this case index register 7 still points to the PMF header.

REGISTERS AFFECTED: A, E and index register 7 for returns 1, 2, 3;
A and E for return 4.

4.11 GFREC

Purpose: To obtain the current record from a PMF.

Calling Sequence:

```
LDA     fileid
CALL    GFREC
DATA    dcb
---           EOF RETURN
---           NORMAL RETURN

fileid  BITS 23-21    FILE TYPE

                        0    STRING
                        1    DATA
                        2    OBJECT
                        3    COREIMAGE

              13-0    PMF header address
```

Description:

GFREC (using GETW) will move the current record from the PMF to the buffer defined by the DCB whose address is at CALL+1. The EOF return is taken whenever an EOF is returned by GETW; the contents of the record obtained are not predictable in this case.

For STRING files this move will terminate only when a 77B character is read or GET indicates an EOF. In the former case, the buffer will be padded out with blanks and the normal return is taken. If the buffer is smaller than the record, the trailing part of the record is lost.

The amount of information transmitted from COREIMAGE or DATA files is determined by the DCB word count. It is the responsibility of the calling program to set this correctly.

The amount of information transmitted from an OBJECT file is a function of the first word of the record. The buffer whould be as large as the largest possible record (10 words).

REGISTERS AFFECTED: A, E, Index Register 7

4.12 PFREC

Purpose: To move a record into a PMF.

Calling Sequence:

```
        LDA     fileid
        CALL    PFREC
        DATA    dcb
        ---             EOF RETURN
        ---             NORMAL RETURN
```

See 4.11 for a description of fileid.

Description:

The DCB whose address is at CALL+1 describes the buffer that contains the record to be moved. Except for OBJECT files, the entire buffer area is moved using PUTW or PUT. For OBJECT files the number of words moved is determined from the first word of the record, or the length of the buffer area, whichever is smaller. If the file is type STRING, the 77B character is also placed in the file after the record.

The EOF return is taken whenever such a return is given by PUTW or PUT.

REGISTERS AFFECTED: A, E, Index Register 7

4.13 FIND

Purpose: To locate a file on the disc.

Calling Sequence:

```
        LDA     'SYMB'
        LDE     'OL'
        BSM     FIND
        ---             NOT FOUND RETURN
        ---             NORMAL RETURN
```

Description:

FIND searches the file directory for the specified disc file. The search begins at the beginning of the directory and continues until a match is found or the end-of-directory is reached. If a match is found, the main memory address of the found entry is placed in index register 7, the binary disc address of the corresponding file is placed in the A register and control returns to CALL+2. If no match is found, control returns to CALL+1 and the index and A register reference the last directory entry and working storage, respectively.

The index register address is that of the first word of the entry. The other five words are at the next five higher memory addresses.

If bit '0' of the E register is '1', the system job number is assumed. Otherwise, the current job number is used.

NOTE: The label FIND must be 'EQU'ed to 356B.

REGISTERS AND STATE SWITCHES AFFECTED: A, E, Index Registers 6 and 7
State Switch 7

4.14 OUTREC

Purpose: To place a record in an output file.

Calling Sequence:

```
a)  LDA     FILEIDENT
    CALL    OUTREC
    DATA    1
    DATA    DCB
    ---             EOF FILE RETURN
    ---             NORMAL RETURN

b)  LDA     FILEIDENT
    CALL    OUTREC
    DATA    0
    ---             BUSY RETURN
    ---             NOT BUSY RETURN
```

FILEIDENT (for non-disc files):

BITS	22-21	FILE TYPE
	13-0	FILE NUMBER
FILE NUMBER	0	POD
	1	TTP
	2	CP (when available)
	3	LP
	4	MT

FILEIDENT (for disc files):

BITS	22-21	FILE TYPE (as above)
	13-0	PMFH
FILE TYPE	0	STRING
	1	DATA
	2	OBJECT
	3	COREIMAGE

Description:

OUTREC utilizes the IOCS procedures MTIO, CPIO, LPIO and TTPIO along with PFREC to place the record in the file. OUTREC works much like the IOCS procedures in that it automatically overlaps record output with program execution. This can be synchronized by doing a 'test'. The buffer should not be altered however, and is complete when control returns to the calling sequence. A 'test' may be performed on an output to a disc file and the NOT BUSY return will always be taken.

The buffer for object and string records should be large enough to accommodate the largest record. The number of words actually sent to the file is gotten from the record itself for object records and for string records is gotten from the DCB and decremented to suppress trailing blanks.

If the record is sent to the card punch (when available), it is first edited to provide BCD output. The buffer must be large enough to accommodate the edited record; it must be twice as large as the DCB word count specifies.

4.15 INREC

Purpose: To obtain a record from an input file.

Calling Sequence:

```
        LDA     FILEIDENT
        CALL    INREC
        NOP     DCB
        ---             EOF/MONITOR REC RETURN
        ---             NORMAL RETURN

        FILEIDENT:      SEE OUTREC FOR FILEIDENT SPECIFICATION
```

Description:

This procedure will obtain the next record from the specified file (device) and place it in the buffer described by the DCB. The input record is always available in this buffer when control returns to the calling sequence, i.e., INREC waits until the record has been obtained before returning. INREC obtains the record by calling the IOCS procedures MTIO, CRIO, TTRIO and GFREC. This means that the limited editing available in TTRIO is available for records obtained from the teletype.

when using GFREC to obtain records from a PMF, the PMF header describes the buffer actually used for disc transfers while the DCB describes the buffer which will finally contain the record.

For files containing variable length records, i.e., string and object files, the buffer must be large enough to accommodate the largest record. The EOF return is taken if the buffer is too small for a particular record. In retrieving records from string files, INREC pads the record with blanks if it is smaller than the buffer. This is not done with object the first word of the record enables its exact size to be determined.

4.16 SRCH (ENTRFN, WRITDS)

Purpose: To locate files on the disc.

Calling Sequence:

```
        LDA     'SYMB'
        LDE     'OL'
        CALL    SRCH
        ---             NOT FOUND RETURN
                        FOUND RETURN
```

Description:

SRCH performs the same function as FIND. The main differences are SRCH uses DISCIO and is relocatable, while FIND uses its own simple I/O and is not relocatable. Another difference is that SRCH has two subprocedures which can be used to maintain the file directory. These are WRITDS and ENTRFN. WRITDS will write the sector of the directory that is currently in main memory back to the directory. ENTRFN must be used after a SRCH failure to enter the name of the 'new' file into the directory. In the new entry, the 'last entry' bit is set and all other words are set to zero. It is the responsibility of the calling program to fill these entries in correctly. Index register 7 points to the new entry. A call for ENTRFN should be followed by one for WRITDS when the entries have been completed.

The calls for these subprocedures are:

```
        CALL    WRITDS
        CALL    ENTRFN
```

The latter enters the routine from the previous CALL SRCH and uses parameters set up by SRCH.

4.17 DFILEN

Purpose: To determine the file number of the file whose name is in the A register.

Calling Sequence:

```
        LDA     file name
        CALL    DFILEN
        ---             INPUT FILE
        ---             OUTPUT FILE
        ---             NOT A FILE
```

Description:

The table below shows the name/number correspondence of the various files.

	File (Device) Name	File Number	
OUT FILES:	POD	0	TT Printer
	TTP	1	Card Punch (if available)
	CP	2	Line Printer
	LP	3	Magnetic Tape Output
IN FILES:	PID	0	
	TTK	1	TT Keyboard
	CR	2	Card Reader
	TTR	3	TT Paper Tape Reader
	MTR	4	Magnetic Tape Input

The file names must be left justified in the A register. The error return is taken if the file cannot be found in the table.

If PID or POD is input, the number of the appropriate file will be obtained from M1CTRL.

If the file is found, the appropriate normal return is taken and the file number is placed in the A register; otherwise, the error return (CALL+3) is taken.

REGISTERS AFFECTED: Index Registers 7 and 6, A

APPENDIX A

CHARACTER SET

The internal character set used by the FST-1 software packages is six-bit trimmed ASCII, TASCII. TASCII characters are obtained from their seven-bit counterparts, parity level excluded, by subtracting 40B. The resulting character set is shown in TABLE A.

BIT POSITIONS 3 - 0 *	BIT POSITIONS 5 - 4			
	00	01	10	11
0000	SPACE	0	@	P
0001	!	1	A	Q
0010	"	2	B	R
0011	#	3	C	S
0100	$	4	D	T
0101	%	5	E	U
0110	&	6	F	V
0111	'	7	G	W
1000	(	8	H	X
1001	)	9	I	Y
1010	*	:	J	Z
1011	+	;	K	[
1100	,	<	L	\
1101	-	=	M	]
1110	.	>	N	↑
1111	/	?	O	←

* Bit 5 is the high-order bit position

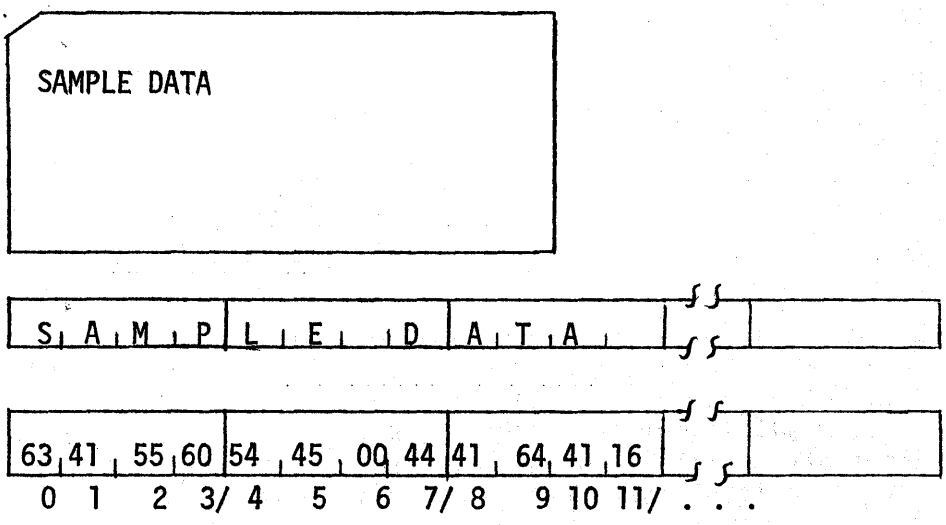
TABLE A - TASCII CODE

Not all I/O devices supported on the FST-1 accept/produce this code. There are, however, conversion procedures for making the proper transformations from one character set to another. A description of these procedures can be found in Section 3.0.

APPENDIX B

BUFFER FORMAT

The character buffers used by the FST-1 software packages are of two kinds. The more common of the two has four six-bit characters per word. The first word of the buffer contains the first four characters of the corresponding I/O record; the first character is in the high order position of the word. The second buffer word contains the next four and so on. This is illustrated in the following diagram, which assumes the buffer starts at location 500.

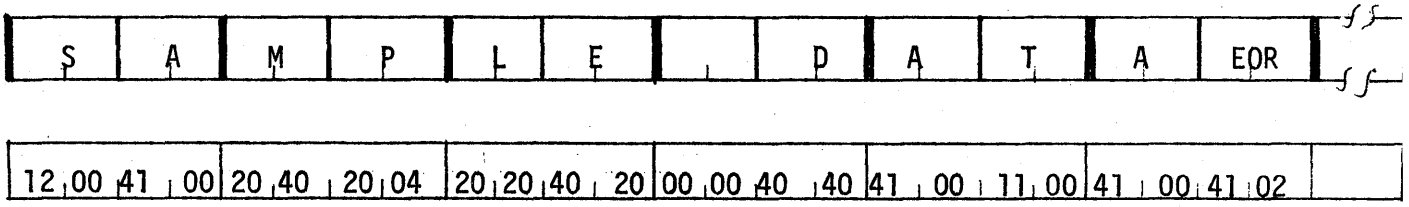


The second part of the illustration shows the relative position of each character in the buffer. The high-order bit of each character position in a word is occupied by the high order bit of the character residing there. This can be seen by part three (3) which gives the octal value of each buffer location after the card code has been converted to trimmed ASCII.

The other format is used by the card I/O devices and has two twelve bit characters per word. These twelve bit characters are a result of a column binary operation. That is, there is a one-to-one correspondence between the punches in a card column and the one bit in the corresponding twelve bit character. The high and low order bits correspond to rows twelve and nine, respectively.

This kind of a buffer is used by the card reader in the BINARY mode. The following diagram illustrates the format of this buffer.

SAMPLE DATA


S A M P L E , D A T A EOR
12 00 41 00
20 40 20 04
20 20 40 20
00 00 40 40
41 00 11 00
41 00 41 02
LOCATION:
500
501
502
503
504
505