SENTRY

FST-1, -2 SUBROUTINE LIBRARY MANUAL

FAIRCHILD
SYSTEMS TECHNOLOGY
A DIVISION OF FAIRCHILD CAMERA AND INSTRUMENT CORPORATION

FST-1, -2 SUBROUTINE LIBRARY MANUAL

Manual Part Number: **57000026**
Date Released: June 1977

FAIRCHILD
SYSTEMS TECHNOLOGY
A DIVISION OF FAIRCHILD CAMERA AND INSTRUMENT CORPORATION

PREFACE

This manual describes callable system subroutine that are available to the user. These routines are automatically loaded when called by the user's program. A familiarity with disk files and disk operating system (DOPSY) procedures is assumed. For additional or reference information refer to the following publications:

Publications	Manual Part Number
Sentry VII User Manual	57000013
FST-2 Computer Manual	57000002
Sentry VII Communication Link User's Manual	57000003
FST-1 Assembler Reference Manual	67094951
Register Formats Reference Manual	67095504

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SECTION 1

IOCS PROCEDURES

1.1 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 1.1.

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
```

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.

TABLE 1-1 SUBROUTINE CALLING SEQUENCE OPERATION CODES

Type of Code	Octal Code	Subroutines					
		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	KILL	PRINT BINARY cr-lf*		WRITE BINARY	PRINT WITH-OUT LINE FEED	WRITE BINARY
Write	4		PRINT BCD cr-lf			PRINT WITH LINE FEED	WRITE TAPE MARK
Motion	5						RECORD SKIP FORWARD
Motion	6					SPACE N LINES	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

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 re-entrancy. 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 paragraphs 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 an is, therefore, omitted.

1.2 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.

1.3 TTRIO

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

Calling Sequence:

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

operation - 1 READ keyboard operation - 0 BUSY test
2 READ paper tape 3 KILL pending input

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
```

1.4 CRIO

Purpose: To input a record from the card reader.

TABLE 1-2 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:

```
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 it can be converted to TASCII by the procedure CRASC. Table 1-2 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-dile. This record should be followed by a dummy one if it is the last record in the input stacker.

1.5 LPIO

Purpose: To output a record to the line printer.

Calling Sequence:

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

	operation -	3	PRINT (no line-feed)
		4	PRINT (with line-feed)
		6	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 of 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 require 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.

1.6 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 DISCO 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.

1.7 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 BACKWARD
```

MTIO transmits blocks of data between magnetic tape and core memory, perform error analysis, and executes miscellaneous commands to cause tape movements and writing or 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, XALL+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.

1.8 COMMUNICATION LINK I/O (CLIO)

This section describes the procedures available to users by means of the call directive to communicate with the host using the Communications Link Driver, CLIO.

GENERAL DESCRIPTION

GENERAL FORM

```
        CALL    CLIO
        DATA    operation code
        DATA    dcb address
        BRU     error/Busy/EOF
```

Calling Sequence:

CLIO is the host Data Link I/O procedure name and when used in the operand of a CALL statement will cause the Data Link Driver to be loaded and linked at COREIMAGE CREATE time.

CALL + 1 - Contains the operation code which defines the desired operation.

CALL + 2 - Contains the Data Control Block (DCB) address. The DCB is a three word block that provides the following information;

DCB + Ø - Message length (max)
DCB + 1 - Message first word address (FWA)
DCB + 2 - CLIO status on return

Data may either be binary or TRASCII. The data message is expressed in FST-2 words. Recall that there can be four TRASCII characters, left justified, per FST-2 word. The maximum word count/character count is 28 words or 112 characters respectively for TRASCII or 37 words Binary.

CALL + 3 - is the error exit on all operations and the busy exit on the busy test. This location should contain a branch to a user written error routine for proper recovery. The A-register contains a value indicating the error type.

On all returns from CLIO, the contents of the index registers preserved. The E-register as well as DCB + 2 contains the status of CLIO. The A-Register contains an error number on error return and zero on a good return (except message type 5, the status response).

CALLING PROCEDURES

Command Ø, Driver Status and Message Control

Message Type Ø, Driver Status

```
CALL    CLIO
DATA    B4 + B3
DATA    dcb
BRU     busy/error
        return - not busy
```

Busy is defined as the last initiated write, operation has not completed. This is due to the time it takes to perform the operation and also includes the time it takes for the host to acknowledge the operation. Busy may also be due to failure, either an inoperative interface or a line disconnect (hang-up) condition. If the latter case exists and the driver is called with a message type other then 3 No, a find-out error will occur after approximately 15 seconds.

NOTE

Refer to Figure 1-1 for a diagram of the Opcode format.

Opcode:		=	Ø,	Busy test, if Busy, Rtn = CALL + 3 ELSE CALL + 4
	B3	=	1,	Status in E-REG & DCB + 2, RTN = CALL + 4
	B21	=	0,	Current input character count
		=	1,	Current output character count

Message Type 1, File Request

```
CALL    CLIO
DATA    10000B + B21 + SSA
DATA    dcb
BRU     error
```

This call causes the host to make available (open) the file defined by the header found in memory pointed to by the DCB. When a file request is made of the host, the next operation must be a file transmit read, message type 2. The Header must be TRASCII.

Opcode:	B21	=	1,	Write, a request of the host
		=	Ø,	Read
	SSA	=	Ø-2,	The Sub-system Source Address

Message Type 2, File Transmit

```
CALL    CLIO
DATA    20000B + B21 + B4 + B3 + SSA
DATA    dcb
BRU     error
```

This call causes either the host (write) or the FST-2 (read to create or append to a named space for the subsequent upload or download of data records. On an upload the file identification is specified by the header found in memory defined by the DCB. On a download, the header is an image of the preceding file request message type 1. The header must be TRASCII.

Opcode	B21	=	1,	Write, used preceding the upload of a data file
		=	0,	Read, used preceding the download of a data file
	B3	=	1,	Append to an existing file
		=	Ø,	Create a new file
	B4	=	1,	The following file is Binary
		=	Ø,	The following file is TRASCII
	SSA	=	Ø-2,	The Sub-System Source address

Message Type 3, DATA

```
CALL    CLIO
DATA    30000B + B21 + B20 + SSA
DATA    dcb
BRU     error/End of File
```

This call causes a data message to be read or written. The required sequence of file request, status response, and file transmit operations must have first been performed.

Opcode: B21 = 1, Write
= Ø, Read
B20 = 1, Binary message
= Ø, TRASCII message
SSA = Ø-2, The Sub-System service address

Message Type 4, File End

```
CALL    CLIO
DATA    40000B + B21 + B3 + SSA
DATA    dcb
BRU     error
```

This call is executed after the last data message is sent or received and signals the completion of the current upload or download. The open file is then either closed and made available for other processing or it is purged and deleted depending on bit 3.

Opcode: B21 = 1, Write
= Ø, Read
B3 = 1, Purge and delete the file
= Ø, Close and keep the file
SSA = Ø-2, The Sub-System Service address

Message Type 5, Status Message

```
CALL    CLIO
DATA    50000B + B21
DATA    dcb
BRU     error
```

This call transfers a system status message. A status message can occur after any read or write to signal an error, but normally occurs after the file transmit and file end messages signalling successful operation. On a read, Status is supplied in the A-register (binary) and the first 2 characters of the header (TRASCII)/ See Appendix B for the list of Status responses.

NOTE

Refer to Figure 1-2 for a diagram of the Status word format.

Opcode: B21 = 1, Write, Status value in B3-8 of the opcode
= Ø, Read, A-Register contains the status value

Message Type 6, Control Messages

```
CALL    CLIO
DATA    60000B + B21 + B4 + B3
DATA    dcb
BRU     error
```

This call causes control and messages to be communicated between systems. If B5, B4 and B3 are zero, a NOP message results.

There is no difference between types 1, 2, or 3 at the driver level but may take on meaning at the next level of software.

Type 1 - I am about to hang-up.
Type 2 - Normal DOPSY/TOPSY operator messages.
Type 3 - Operator messages that are remote system commands.

Command 1 - Initiate Call-Up

```
CALL    CLIO
DATA    1
DATA    dcb
BRU     error
```

This call initiates the call-up procedure and must be executed prior to performing any other call to CLIO (except status requests).

Opcode: = 1

Command 2 - Initiate Hang-up

```
CALL    CLIO
DATA    2
DATA    dcb
BRU     error
```

This call initiates hang-up such that line disconnection is performed in an orderly
Opcode:= 2

OPCODE FORMAT

BØ-2 CMD, The Command Field
=Ø, Status and Message Control
=1, Initiate Call-Up
=2, Initiate Hang-Up

B3-8 PARM1, A message type specific parameter field

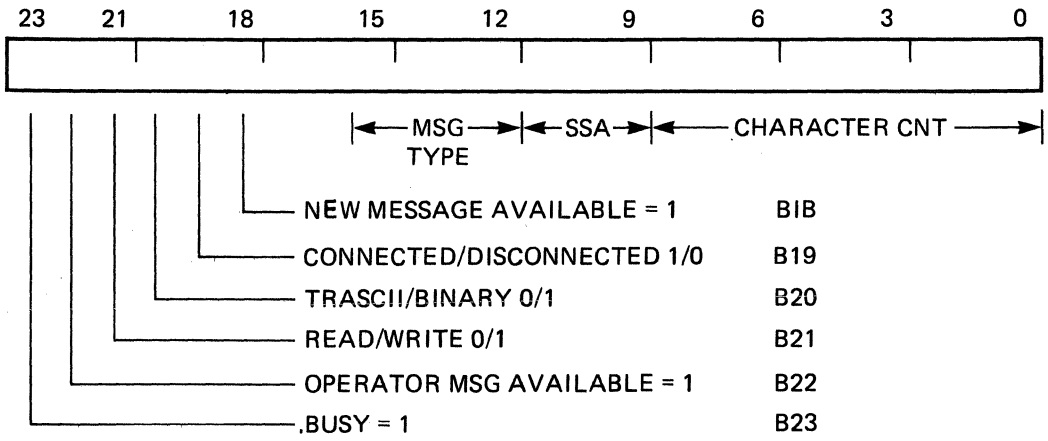
B9-11 SSA, The Source System Sub-Address
=Ø, All non-test station related operations
=1, Station 1
=2, Station 2

B12-15 MSG Type, One of six message types described under command Ø along with CLIO status request.

B19-23 PARM2, A global parameter field.

Figure 1-1, Opcode Format

STATUS



NOTES:

1) When new message available is indicated, the new message type is found in B12-15, else the message type of the current call.

2) The character count is input or output count dependent on B21

3) If an error has occurred the E-register will differ from DCB+2 in that instead of character count, the contents of B0-3 will contain a receive error indication if applicable.

   BØ = 1 = Parity error
   B1 = 1 = Framing error
   B2 = 1 = Overrun error
   B3 = 1 = LRC error

Figure 1-2 Status Word Format

1.9 MESSAGE SEQUENCING

In order to perform a file transfer, upload or download, a prescribed sequence of message type calls must be performed.

Download

Read/Write	Message Type	Function
Write	1	File request
Read	2*	File transmit - open
Write	5	Status - OK, continue
Read	3	Data message - first
Read	4	File end - close
Write	5	Status - successful transfer

Upload

Read/Write	Message Type	Function
Write	2	File transmit - open
Read	5	Status - OK, continue
Write	3**	Data message - first
Write	3	Data message - last
Write	4	file end - close
Read	5	Status - successful transfer

Close - Purge

Read/Write	Message Type	Function
Write	2	file transmit - open
Read	5	Status - OK, continue
Write	4	File end - close with purge
Read	5	Status - successful operation

* If the host cannot honor the file request, this will be a message type 5, status with the appropriate status value.

** If the nost cannot honor this write, a message type 5, Status, will result.

SECTION 2

CONVERSION PROCEDURES

2.1 INTRODUCTION

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

2.2 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.

2.3 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 -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

SECTION 3

FILE PROCESSING ROUTINES

3.1 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 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 3-1.

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 (see Figure 3-1) 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.

The area of main memory that the file is segmented into is defined by PS and PL (see Figure 3-1). 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 3-1 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 of 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 dic before reading in a new one.

TABLE 3-1 PMF HEADER FORMAT

Word	Bit	Description
1	23-0	CP - Current Pointer - word/char relative to FS
2	23-0	WS - Window Start - words relative to FS
3	23-0	WL - Window Length - words
4	23-0	FS - File Start - Disc Address in words
5	23-0	FL - File Length - words
6	23-0	PS - Page Start - Memory address in words
7	23-0	PL - Page Length - Sectors
8	23-0	First 4 characters of file name
9	23-12	Last 2 characters of file name
9	4	Factor DOF File Olen
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 window contents are altered.

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 initalize the entries of a dummy PMF so that the WINDOW completely encompasses the entire buffer (file). In particular, WS=), 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.

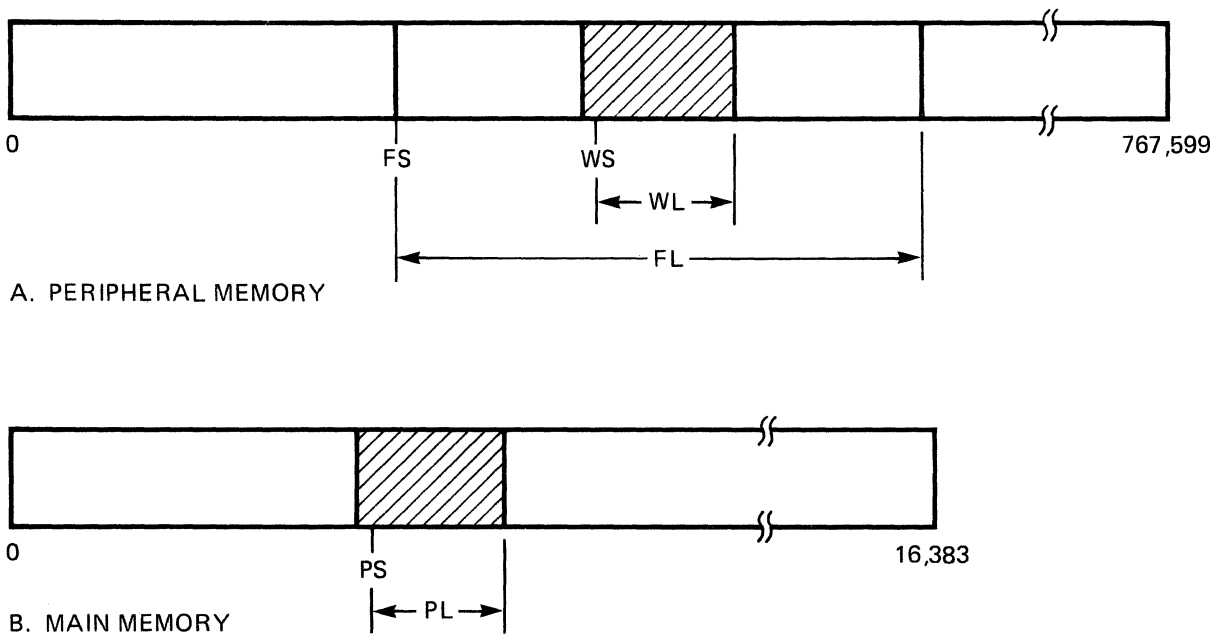


Figure 3-1 Peripheral and Main Memory Header

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.

3.2 OPEN

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

Calling Sequence:

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

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, paragraph 3-1 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

3.3 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

3.4 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_ 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 DISCO 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

3.5 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:

The 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 allowage range the EOF return is taken.

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

REGISTERS AFFECTED: A,E

3.6 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 allowage range, the EOF return is taken.

REGISTERS AFFECTED: A
E on normal return

3.7 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 _ 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

3.8 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 _ 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.

T he 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

3.9 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_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

3.10 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 is assumed. 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 resiters 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 thise 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.

3.11 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 a CALL=1. The EOF return is taken whenever an WOF 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 form 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 would be as large as the largest possible record (10 words).

REGISTER AFFECTED: A, E, Index Register 7

3.12 PFREC

Purpose: To move a record into a PMF.

Calling Sequence:

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

See 3.11 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

3.13 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.

3.14 OUTREC

Purpose: To place a record in an output file.

Calling Sequence:

```
a)  LDA     FILEIDENT
    CALL    OUTREC
    DATA    1
    DATA    DCB
    ---             EOF FILE RETRUN
    ---             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	MTW
	5	DOF
	6	CLO

INREC	0	PID
	1	TTK
	2	CR
	3	TTR
	4	MTR
	5	DIF
	6	CLI

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 work 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 obtained from the record itself for object records and for string records is obtained 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.

3.15 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, DIF. This means that the limited editing available in TTRIO is available for records obtained from the teletype.

NOTE

Records beginning with '$$' in columns 1 and 2 coming from DIF are treated as '//'

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.

3.16 SRCH (ENTREN, 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 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.

3.17 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 3-2, below shows the name/number correspondence of the various files.

TABLE 3-2 OUTFILES, INFILES, NAME/NUMBER CORRESPONDENCE

	File (Device) Name	File Number	
Out Files	POD	0	
	TTP	1	TT Printer
	CP	2	Card Punch (if available)
	LP	3	Line Printer
	MTW	4	Magnetic Tape Output
	DOF	5	Disc Output File
	CLO	6	Data Link Output
	FDOF	7	
	SDOF	8	
In Files	PID	0	
	TTK	1	TT Keyboard
	CR	2	Card Reader
	TTR	3	TT Paper Tape Reader
	MTR	4	Magnetic Tape Input
	DIF	5	Disc Input File
	CLI	6	Data Link Input
	FDIF	7	
	SDIF	8	

The file name 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 MICTRL.

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 and FST-2 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-1.

TABLE A-1 TASCCI CODE

		BIT POSITIONS 5* & 4			
		00	00	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
BIT	0110	&	6	F	V
POSITIONS	0111	'	7	G	W
3 - 0	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

Not all I/O devices supported on the FST-1 and FST-2 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 paragraph 2.0.

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 Figure B-1, which assumes the buffer starts at location 500.

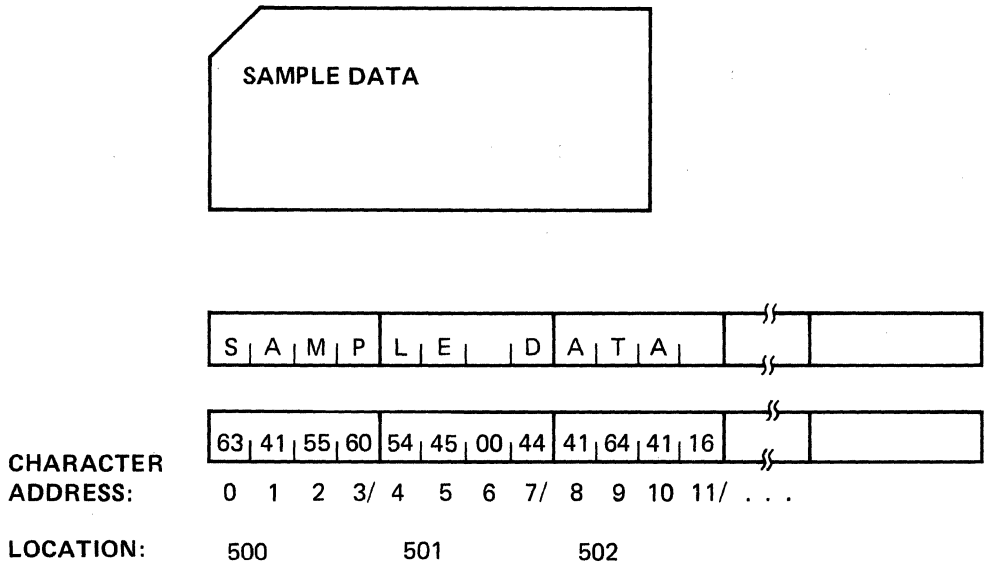


Figure B-1 Four, Six-Bit Characters Per Word Format

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 location 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 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. Diagram in Figure B-2 illustrates the format of this buffer.

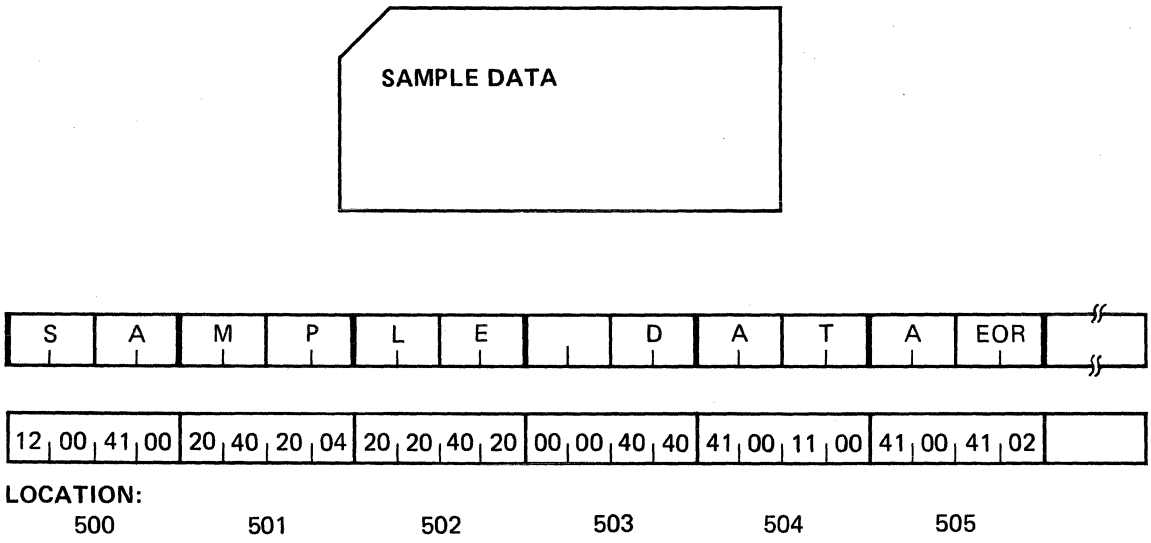


Figure B-2 Two, Twelve-Bit Characters Per Word Format

APPENDIX C

DRIVER ERROR MESSAGE

ERROR #	DESCRIPTION
1	DATA LINK interrupt and no RS-232 status indicator is set. An interrupt has been executed, but BG - DATA set change or BL - Receiver done or BBC - Transmit done are not true.
2	Data Set Change (DSC) interrupt has occurred and either no RS-232 bit or more than one RS-232 bit has changed since the last data set change.
3	Lost Data Set Ready (DSR) and not a formal hang-up. The host has probably timed out and hung-up due to an uncompleted sequence while there was a request in the driver.
4	Lost Clear-To-Send (CTS) and transmission was incomplete.
5	Lost Receiver Line Signal Detect (RLSD) during a receive procedure.
6	The Host has hung-up.
7	The last message was cancelled. This is due to an illegal sequence of line protocal characters.
10	Transmit error, 10 successive, attempts at writing a message have frailed. Disconnection occurs.
11	A message read or write call has been made and the line is not connected. A call-up is requested to establish line connection.
12	DCB error, a calling parameter is in error a) Command in the opcode is not Ø, 1 or 2 b) Message type in the opcode is greater than 6 c) PARMI is in error in a Driver Status call d) A call for an operator message has been made and none exists e) PARMI is in error on a message type 6 write.

13 DCB error, error in the Buffer or message length supplied.

a) Read an operator message and the buffer provided is insufficient (the partial message is past)
b) The message supplied the driver is larger than allowed
c) Read a message and the buffer provided is insufficient.

14 Time out error, the following situtations occur for greater than 15 seconds

a) The driver is hung in a busy state
b) The host retains line ownership and a write attempts exists
c) A call for a read is made and does not complete.

15 Message sequence error (normal error at times)

a) A read of a message type 'n' made but message type 'm' is found.

e.g.,	Write	Message type 1	
	Read	Message type 2	but a message type 5 exists with host status.
or	Read	Message type 3	but a message type 4 exists indicating end of file.

APPENDIX D

HOST STATUS MESSAGES

ERROR #	DESCRIPTION
0	The requested operation was performed without error
	-UPLOAD ERRORS-
1	N/A
2	N/A
3	N/A
4	A file open was attempted and there is a file already open. A file transmit message was sent and the subchannel is already in use.
5	Host RTE error Directory search/add error in response to a file transmit message (e.g., Disc full).
6	Duplicate file, an attempt to create a file which already exists.
7	Host RTE error An open error in response to a file transmit - oppend operation.
8	FST-2 System error Attempt to add, with a data message, to a closed file.
9	Host RTE error A disc write error while oppending file data to an open file.
10	FST-2 System error Attempt to close an already closed file.
11	Host RTE error Error encountered while attempting to close a file.
12	Host RTE error Error encountered while attempting to purge a file.

13	Host RTE error Unable to find the directory entry on a close-purge operation.
14	Host RTE error A Directory write error during a close-purge operation.
15	N/A
16	N/A
17	N/A
18	N/A
19	FST-2 System error Attempting to close-purge a file and the file ID's are not the same.

-DOWN LOAD ERRORS-

50	N/A
51	A down load request was made and a download is already in progress.
52	Any file error encountered during a download procedure e.g., the requested file was not found, RTE file error.

APPENDIX E

HOST-FST-2 COMM LINK I/O EXAMPLES

```
                    PAGE
                *
                *
                *
00000 00000007      OBJ   7
                *
                *
                *
                *       HOST - FST2 COMM LINK I/O EXAMPLES
                *
                *
                *   CLIO CALLING SEQUENCE:
                *
                *
                *       ENT:  N/A
                *       EXT:  A-REG = ERROR # ON ERROR, ELSE 0
                *             E-REG = CLIO STATUS
                *             X-REG'S ARE PRESERVED
                *
                *       CALL: CALL    CLIO
                *             DATA    OPCODE
                *             DATA    DCB
                *             BRU     ERROR/BUSY/EOF - ROUTINE
                *                     RTN - NORMAL
                *
                *       DCB:  DATA    LENGTH (WORDS)
                *             DATA    BUFFER/MESSAGE FIRST WORD ADDR
                *             DATA    0 (STATUS)
                *
                *
                *
00000 00000000 HOSTIO  PROG  0
00001 01000002         BPU   BEGIN
```

```
                        PAGE
                    *
                    *
                    *       PROGRAM EQUATES FOR ERROR FREE LISTING
                    *
                    *
          00000002 BEGIN    EQU  *          PROGRAM STARTING POINT
          00000002 DONE     EQU  *          PROGRAM ENDING POINT
          00000002 ABORT    EQU  *          PROGRAM ERROR EXIT
          00000002 LIST     EQU  *          OPERATOR MESSAGE LIST ROUTINE
          00000002 ELIST1   DQU  *          CONVERT & LIST DRIVER ERROR #
          00000002 FLIST2   EQU  *          CONVERT & LIST HOST ERROR # & MSG
                    *
                    *
                    *       CONSTANTS & TEMP STORAGE
                    *
                    *
00002 00000000  D0        DATA 0
00003 00000004  D4        DATA 4
00004 00000005  D5        DATA 5
00005 00000016  D13       DATA 13
00006 00000017  D17       DATA 17B
                    *
00007 20000000  B22       DATA 20000000B
00010 01000000  B18       DATA 01000000B
                    *
00011 01000000  WMT5      DATA 10050000B MSG TYPE 5, WRITE, OPCODE
                    *
00012 00000000  TIMER     DATA 0         PREVENT 'HUNG' CONDITION
00013 00023420  T15S      DATA 10000     INITIAL TIMER VALUE
```

```
                          PAGE
                    *
                    *
                    *       DATA LINK I/O DCB'S
                    *
                    *
      00000014 OPIDCB     EQU  *          OPR MESSAGE INPUT DCB
00014 00000024 OPILEN     DATA 20
00015 00000041            DATA OPIBFF
00016 00000000            DATA 0
                    *
      00000017 OPODCB     EQU  *          OPR MESSAGE OUTPUT DCB
00017 00000023 OPOLEN     DATA 19
00020 00000065            DATA OPOBFF
00021 00000000            DATA 0
                    *
      00000022 HIDCB      EQU  *          40 CHAR HEADER INPUT DCB
00022 00000013            DATA 11               (+ 1 FOR ONTRL INFO)
00023 00000110            DATA HIBFF
00024 00000000            DATA 0
                    *
      00000025 HODCB      EQU  *          40 CHAR HEADER OUTPUT DCB
00025 00000012            DATA 10
00026 00000123            DATA HOBFF
00027 00000000            DATA 0
                    *
      00000030 DLIDCB     EQU  *          DATA (MSG TYPE3) INPUT DCB
00030 00030052 DLILFN     DATA 42
00031 00000135            DATA DLIBFF
00032 00000000            DATA 0
                    *
      00000033 DLODCB     EQU  *          DATA (MSG TYPE 3) OUTPUT DCB
00033 00000052 DLOLEN     DATA 42
00034 00000211            DATA DLOBFF
00035 00000000            DATA 0
                    *
      00000033 DLODCB     EQU  *          DATA (MSG TYPE 3) OUTPUT DCB
00033 00000052 DLOLEN     DATA 42
00034 00000211            DATA DLOBFF
00035 00000000            DATA 0
                    *
      00000036 BSYDCB     EQU  *          BUSY TEST DCB
00036 00000001            DATA 1
00037 00000025            DATA HODCB
00040 00000000            DATA 0
                    *
                    *
                    *       DATA LINK I/O BUFFERS
                    *
                    *
00041 00000024 OPIBFF     BSS  20         OPR MSG IN FROM HOST BFFR
00065 00000023 OPOBFF     BSS  19         OPR MSG OUT TO HOST BFFR
00113 00000013 HIBFF      BSS  11         HEADER INPUT BFFR
00123 00000012 HOBFF      BSS  10         HEADER OUTPUT BFFR
00135 00000054 DLIBFF     BSS  44         DATA INPUT BFFR
00211 00000054 DLOBFF     BSS  44         DATA OUTPUT BFFR
```

```
                    PAGE
               *
               *
               *       UPLOAD EXAMPLE, BINARY FILE
               *
               *
      00000265 UPLOAD    EQU   *
00265 12000545           BSM   WRMT2B   WRITE FILE TRANSMIT - OPEN
00265 01000343           BRU   UPERR1     ERROR - DRIVER
               *
00267 12000511           BSM   RDMT5    READ HOST'S STATUS RESPONSE
00274 01000343           BRU   UPPER1     ERROR - DRIVER
00271 025000345          BNEZ  UPPER2     ERROR - HOST
               *
00272 12000572           BSM   WRMT3B   WRITE FIRST DATA MESSAGE
00273 01000347           BRU   UPPER3     ERROR DRIVER/HOST
               *
00274 12000424           BSM   DLWAIT   WAIT NOT BUSY, SAMPLING   FOR OPR MSG
00275 01000343           BRU   UPERR1     ERROR - DRIVER
               *
               *         *      *
               *         *      *     UPDATE THE DCB & CONTINUE . . . OR
               *         *      *
               *
00276 12000572           BSM   WRMT3B   WRITE THE LAST DATA MESSAGE
00277 01000347           BRU   UPERR3     ERROR - DRIVER/HOST
               *
00300 12000601           BSM   WRMT4    WRITE FILE END - CLOSE
00301 01000347           BRU   UPERR3
               *
00302 12000511           BSM   BDMT5    WAS TRANSFER SUCCESSFUL?
00303 01000343           BRU   UPERR1
00304 02500345           BNEZ  UPERR2     NO
               *
00305 01000002           BRU   DONE       YES, COMPLETE
```

```
                    PAGE
               *
               *
               *       DOWN LOAD EXAMPLE, BINARY FILE
               *
               *
     00000306 DNLOAD    EQU   *
00306 12000527          BSM   WRMT1    WRITE FILE REQUEST
00307 01000357          BRU   DNERR1     ERROR - DRIVER
               *
00310 12000455          BSM   RDMT2    READ FILE TRANSMIT
00311 01000361          BRU   DNERR2     ERROR - DRIVER/HOST
               *
00312 24000002          LDA   DO       WRITE - STATUS = OK
00313 12000617          BSM   WRMT5
00314 01000357          BRU   DNERR1
               *
     00000315 DNLO1     EQU   *
00315 12000473          BSM   RDMT3B   READ N'TH DATA MESSAGE
00316 01000361          BRU   ONERR2     ERROR - DRIVER OR NORMAL EOF
               *
00317 12000424          BSM   DLWAIT   SAMPLE FOR OPERATOR MSG'S
00320 01000367     BRU  ONERR1
               *
               *         *       *
               *         *       *    UPDATE DCB & CONTINUE
               *         *       *
               *
00321 01000315          BRU   DNLD1
               *
               *
     00000322 DNLD2     EQU   *
00322 12000502          BSM   RDMT4    READ FILE END - CLOSE
00323 01000361          BRU   DNERR2
               *
00324 24000002          LDA   DO
00325 12000617          BSM   WRMT5    WRITE STATUS = OK
00326 01000361          BPU   DNERR2
               *
00327 01000002          BRU   DONE     COMPLETE
```

```
                    PAGE
                *
                *
                *       CLOSE - PURGE EXAMPLE
                *
                *
      00000330 CLOSEF   EQU   *
00330 12000536          BSM   WRMT2    WRITE FILE TRANSMIT - OPEN
00331 01240343          BRU   UPERR1
                *
00332 12000511          BSM   RDMT5    WAS OPERATION SUCCESSFUL?
00333 01000343          BRU   UPERR1
00334 02500345          BNEZ  UPERR2     NO
                *
00335 12000610          BSM   WRMT4P   WRITE FILE END WITH PURGE
00336 01000343          BRU   UPERR1
                *
00337 12000511          BSM   RDMT5    WAS OPERATION SUCCESSFUL?
00340 01000343          BRU   UPERR1
00341 02500345          BNEZ  UPERR2     NO
                *
00342 01000002          BRU   DONE       YES, COMPLETE
```

HOST-FST-2 COMM LINK I/O EXAMPLES (Continued)

```
                          PAGE
                *
                *
                *       ERROR ROUTINES
                *
                *
                *         UPLOAD DRIVER ERRORS
                *
                *
      00000343 UPERR1     EQU   *
00343 10000002            BSM   ENLIST1  LIST DRIVER ERROR #
00344 01000002            BRU   ABORT
                *
                *         HOST RESPONSES
                *
      00000345 UPERR2     EQU   *
00345 12000002            BSM   ENLIST2  LIST HOST ERROR # AND MESSAGE
00346 01000002            BRU   ABORT
                *
                *         POSSIBLE NORMAL SEQUENCE
                *
      00000347 UPERR3     EQU   *
00347 23000005            CAM   D13      SEQUENCE ERROR ?
00350 03500343            BNE   UPERR1     NO, DRIVER ERROR
00351 07010000            EXC
00352 07022014            LS    12         YES, GET STATUS IN E-REG
00353 26000006            AND   017
00354 23000004            CAM   D5       MESSAGE TYPE 5 ?
00355 03200345            BE    UPERR2     YES, LIST IT
00356 01000343            BRU   UPERR1
                *
                *
                *         DOWN LOAD DRIVER ERRORS
                *
                *
      00000357 DNERR1     EQU   *
00357 12000002            BSM   ELIST1   LIST DRIVER ERROR #
00360 01000002            BRU   ABORT
                *
                *         POSSIBLE NORMAL SEQUENCE
                *
      00000361 DNERR2     EQU   *
00361 23000005            CAM   D13      SEQUENCE ERROR ?
00362 03500357            BNE   DNERR1     NO, DRIVER ERROR
00363 07010000            EXC
00364 07022214            LS    12         YES, GET STATUS FROM E-REG
00365 26000006            AND   017
02366 23000003            CAM   D4       FILE END MESSAGE ?
00367 03200322            BE    DNLD2      YES, NORMAL, GO READ IT
00370 23000004            CAM   D5         NO, HOST STATUS ?
00371 03200373            BE    DNERR3     YES, LIST IT
00372 01000357            BRU   DNERP1
                *
                *
                *
      00000373 DNERR3     EQU   *
00373 12000511            BSM   RDMT5    READ HOST RESPONSE
00374 01000357            BRU   DNERR1
                *
00375 12000002            BSM   ELIST2   LIST HOST ERROR # AND MESSAGE
00376 01000002            BRU   ABORT
```

```
                        PAGE
                *
                *
                *       CLIO PROCEDURE CALLS
                *
                *         ENT:      N/A
                *         EXT:      N/A
                *         CALL:     BSM       DL(X)
                *                             RTN - ERROR, A-REG - #
                *                             RTN - OK
                *
                *
                *         DLCALL - DATA LINK CALLUP PROCEDURE
                *
                *
00377 00000000 DLCALL     PZE   0
00400 12000000            CALL  CLIO
00401 00000001            DATA  1
00402 00000036            DATA  BSYDCB
00403 01040377            BRU*  DLCALL
00404 36000377            AOM   DLCALL
00405 01640377            BRU*  DLCALL
                *
                *         DLHANG - DATA LINK HANG-UP PROCEDURE
                *
00406 00000000 DLHANG     PZE   0
00407 12000000            CALL  CLIO
00410 00000002            DATA  2
00411 00000036            DATA  BSYDCB
00412 01040406            BRU*  DLHANG
00413 36000406            AOM   DLHANG
00414 01040406            BRU*  DLHANG
                *
                *         DLSTAT - GET DATALINK DRIVER STATUS
                *
00415 00000000 DLSTAT     PZE   0
00416 12000000            CALL  CLIO
00417 00000010            DATA  10B
00420 00000035            DATA  BSYDCB
00421 01040415            BRU*  DLSTAT
00422 36000415            AOM   DLSTAT
00423 01040015            BRU*  DLSTAT
```

```
                        PAGE
                *
                *
                *       DLWAIT - WAIT FOR DATA LINKNOT BUSY
                *                TEST FOR & LIST ANY OPR MSG'S
                *
                *       ENT:     N/A
                *       EXT:     N/A
                *       CALL:    BSM  DLWAIT
                *                     RTN - ERROR, A- REG = #
                *                     TRN - NORMAL
                *
                *
00424 00000000 DLWAIT   PZE   0
00425 24000013         LDA   T15S          SET 15 SECOND TIME-OUT
00426 14000012         STA   TIMER
                *
      00000427 DLWAT1   EQU   *
00427 24000012         LDA   TIMER         ARE WE HUNG ?
00430 37000012         SOM   TIMER
00431 02200441         BZ    DLWAT2         YES, ERROR 00, ABORT
                *
00432 12000415         BSM   DLSTAT        STATUS TEST
00433 01000441         BRU   DLWAT2         ERROR - DRIVER
                *
00434 07010000         EXC
00435 23000007         CAM   B22           BUSY OR READY, OPR MSG AVAILABLE ?
00436 03040442         BBC   DLWAT3         YES
00437 02100427         BN    DLWAT1         NO, JUST BUSY ?
00440 36000424         AOM   DLWAIT         RDY - CONTINUE
                *
      00000441 DLWAT2   EQU   *
00441 01000424         BRU*  DLWAIT
                *
      00000442 DLWAT3   EQU   *
00442 12000520         BSM   RDMT6         READ THE OPERATOR MESSAGE
00443 01000441         BRU   DLWAT2
00444 12000002         BSM   LIST           & LIST IT
00445 01000427         BRU   DLWAT1         CONTINUE
```

```
                    PAGE
                *
                *
                *       DATA LINK MESSAGE READ & WRITE ROUTINES
                *
                *         ENT:      N/A  (EXCEPT WRMT5)
                *         EXT:      N/A
                *         CALL:     BSM       RD/WR-MT(X)
                                              RTN - ERROR
                                              RTN - OK
                *
                *
                *         READ MSG TYPE 1 - FILE REQUEST
                *
                *
00446 00000000 RDMT1      PZE   0
00447 12000000            CALL  CLIO
00450 00010000            DATA  00010000B
00451 00000022            DATA  HIDCB
00452 01040446            BRU*  RDMT1
00453 36000446            AOM   RDMT1
00454 01040446            BRU*  RDMT1
                *
                *         READ MSG TYPE 2 - FILE TRANSMIT
                *
00455 00000000 RDMT2      PZE   0
00456 12000000            CALL  CLIO
00457 00020000            DATA  000200008
00460 00000022            DATA  HIDCB
00461 01040455            BRU*  RDMT2
00462 36000455            AOM   RDMT2
00463 01040455            BRU*  RDMT2
                *
                *         READ MSG TYPE 3, ASCII - DATA
                *
00464 00000000 RDMT3A     PZE   0
00465 12000000            CALL  CLIO
00466 00030000            DATA  000300008
00467 00000030            DATA  DLIDCB
00470 01040464            BRU*  RDMT3A
00471 36000464            AOM   RDMT3A
00472 01040464            BRU*  RDMT3A
                *
                *         READ MSG TYPE 3, BINARY - DATA
                *
00473 00000003 RDMT3B     PZE   0
00474 12000000            CALL  CLIO
00475 04030000            DATA  040300008
00476 00000030            DATA  DLIDCB
00477 01040473            BRU*  RDMT3B
00500 36000473            AOM   RDMT3B
00501 01040473            BRU*  RDMT3B
                *
                *         READ MSG TYPE 4 - FILE END
                *
00502 00000000 RDMT4      PZE   0
00503 12000000            CALL  CLIO
00504 00040000            DATA  000400008
00505 00000022            DATA  HIDCB
00506 01040502            BRU*  RDMT4
00507 36000502            AOM   RDMT4
00510 01040502            BRU*  RDMT4
```

```
                        PAGE
                *
                *       READ MSG TYPE 5 - HOST STATUS
                *
00511 00000000 ROM15    PZE   0
00512 12000000          CALL  CLIO
00513 00050000          DATA  00050000B
00514 00000022          DATA  HIDCB
00515 01040511          BRU*  RDMT5
00516 36000511          AOM   RDMT5
00517 01040511          BRU*  RDMT5
                *
                *       READ MSG TYPE 6 - OPERATOR MESSAGE
                *
00520 00000000 RDMT6    PZE   0
00521 12000000          CALL  CLIO
00522 00060000          DATA  00060000B
00523 00000014          DATA  OPIDCB
00524 01040520          BRU*  RDMT6
00525 36000520          AOM   RDMT6
00526 01040520          BRU*  RDMT6
                *
                *       WRITE - MSG TYPE 1 - FILE RQUEST
                *
00527 00000000 WRMT1    PZE   0
00530 12000000          CALL  CLIO
00531 10010000          DATA  10010000B
00532 00000025          DATA  HODCB
00533 01040527          BRU*  WRMT1
00534 36000527          AOM   WRMT1
00535 01040527          BRU*  WRMT1
                *
                *       WRITE MSG TYPE 2 - FILE TRANSMIT
                *
00536 00000000 WRMT2    PZE   0
00537 12000000          CALL  CLIO
00540 10020000          DATA  10020000B
00541 00000025          DATA  HODCB
00542 01040536          BRU*  WRMT2
00543 36000536          AOM   WRMT2
00544 01040535          BRU*  WRMT2
                *
                *       WRITE - MSG TYPE 2 - FILE TRANSMIT, BINARY
                *
00545 00000000 WRMT2B   PZE   0
00546 12000000          CALL  CLIO
00547 10020020          DATA  10020020B
00550 00000025          DATA  HODCB
00551 01040545          BRU*  WRMT2B
00552 36000545          AOM   WRMT2B
00553 01040545          BRU*  WRMT2B
                *
                *       WRITE - MSG TYPE 2 - FILE TRANSMIT + APPEND
                *
00554 00000000 WRMT2P   PZE   0
00555 12000000          CALL  CLIO
00556 10020010          DATA  10020010B
00557 00000025          DATA  HODCB
00560 01040554          BRU*  WRMT2P
00561 36000554          AOM   WRMT2P
00562 01040554          BRU*  WRMT2P
```

```
                        PAGE
                *
                *       WRITE MSG TYPE 3, ASCII - DATA
                *
00563 00000000 WRMT3A   PZE   0
00564 12000000          CALL  CLIO
00565 10030000          DATA  10030000B
00566 00000033          DATA  DLODCB
00567 01040563          BRU*  WRMT34
00570 36000563          AOM   WRMT34
00571 01040563          BRU*  WRMT3A
                *
                *       WRITE MSG TYPE 3, BINARY - DATA
                *
00572 00000000 WRMT3B   PZE   0
00573 12000000          CALL  CLIO
00574 14000000          DATA  14030000B
00575 00000033          DATA  DLODCB
00576 01040572          BRU*  WRMT3B
00577 36000572          AOM   WRMT3B
00600 01040572          BRU*  WRMT3B
                *
                *       WRITE MSG TYPE 4 - FILE END
                *
00601 00000000 WRMT4    PZE   0
00602 12000000          CALL  CLIO
00603 10040000          DATA  10040000B
00604 00000025          DATA  HODCB
00605 01040601          BRU*  WRMT4
00606 36000601          AOM   WRMT4
00607 01040601          BRU*  WRMT4
                *
                *       WRITE MSG TYPE 4 - FILE END W/PURGE
                *
00610 00000000 WRMT4P   PZE   0
00611 12000000          CALL  CLIO
00612 10040010          DATA  10040010B
00613 00000025          DATA  HODCB
00614 01040610          BRU*  WRMT4P
00615 36000610          AOM   WRMT4P
00616 01040610          BRU*  WRMT4P
```

```
                        PAGE
                *
                *       WRITE MSG TYPE 5 - STATUS MSG
                *
00617 00000000 WRMT5    PZE   0
00520 07026003          SL    3         MERGE IN STATUS ERROR #
00621 27000011          OR    WMT5
00622 14000624          STA   *+2
00623 12000000          CALL  CLIO
00624 10050000          DATA  10050000B
00625 00000025          DATA  HODCB
00625 01040617          BRU*  WRMT5
00627 36000617          AOM   WRMT5
00630 01040617          BRU*  WRMT5
                *
                *       WRITE MSG TYPE 6 - OPERATOR MESSAGE
                *
00631 00000000 WRMT6    PZE   0
00632 12000000          CALL  CLIO
00633 10060020          DATA  10060020B
00634 00000017          DATA  OPODCB
00635 01040631          BRU*  WRMT6
00636 36000631          AOM   WRMT6
00637 01040631          BRU*  WRMT6
                *
                *
                *
00637 00000000          END
```

INDEX