1. IDENTIFICATION
1.1 Digital-7-22A-I/O
1.2 Bidirectional PDP-7 DECTape Subroutines
1.3 January 21, 1966
2. ABSTRACT

The Bidirectional PDP-7 DECTape Subroutines allow the programmer to transfer variable length records to or from DECTape in either direction depending on the current position of the tape. The only requirement is that the standard DECTape format be used (1100 usable blocks of 400 words each). Mainly the subroutines minimize access time to the DECTape, and allow program overlap with data transfers using the program interrupt or automatic priority interrupt facilities. With three exceptions,* the subroutines are completely compatible with the unidirectional subroutines described in Digital-7-22-I/O (which are not obsolete). Information written with either set can be read with the other. In addition, data is written as if it were in the forward direction; so that the record need not be read in the same direction as it was written. The amount of data transferred need not be an integral number of blocks. Though the routines themselves are loaded into the first 8K of core, data transfers can address normal or extended memories. These new subroutines make more efficient use of the DECTape; however, as they are significantly larger than the basic set (450 locations for the unidirectional, 604 locations for the bidirectional), the user must choose the set most applicable to the job.

3. REQUIREMENTS

3.1 Storage

The subroutines occupy approximately 604 locations including variable registers and literals, and must be loaded into the first 8K of memory.

3.3 Equipment

The subroutines function with a 555 or TU55 DECTape drive and a 550 or 550A DECTape control interfaced to a PDP-7. The subroutines will not run on a PDP-4.

4. USAGE

4.1 Loading

The subroutines are in ASCII format designed to be assembled as part of the user's program. The tapes contain no origin, no starting address, and no undefined symbols. In order to link with the user's program, however, the following items are required as part of the main program.

4.1.1 In order to correctly use the program interrupt or automatic priority interrupt facilities, the main program must include the following coding. Reference should be made to the description of the interrupt facilities in the PDP-7 Users Handbook (F-75).

For the standard program interrupt control, the main program should store a JMP X instruction at location 1 where X must include the following instructions:

*1) The Search subroutine can no longer be used as a separate independent subroutine.
2) The register MMWA1 no longer holds the next block to be transferred (or the next free block). See Section 4.2.7, page 5.
3) Starting and ending core addresses for the Read and Write Subroutines must be 15-bit addresses and can no longer be LAW instructions.
For the automatic priority interrupt, the program should store a JMS X instruction at location 43 (assuming DCTape is attached to channel 3) where X must include the following instructions:

\[
\begin{align*}
X, & \\
& X, 0 \\
& DAC ACSAVE /SAVE THE ACCUMULATOR \\
& MMEF /SKIP ON DECTAPE ERROR FLAG \\
& SKP \\
& JMP I MMERR /SKIP ON DECTAPE DATA FLAG \\
& MMDF \\
& SKP \\
& JMP I MMDATA \\
& MMBF /SKIP ON DECTAPE BLOCK END FLAG OR HLT \\
& SKP \\
& JMP I MMBLF /IF NO OTHER OPTIONS ATTACHED \\
& (Any additional options attached to the interrupt are checked here)
\end{align*}
\]

For the automatic priority interrupt, the program should store a JMS X instruction at location 43 (assuming DCTape is attached to channel 3) where X must include the following instructions:

\[
\begin{align*}
X, & \\
& 0 \\
& DAC ACSAVE /SAVE THE ACCUMULATOR \\
& MMEF /SKIP ON DECTAPE ERROR FLAG \\
& SKP \\
& JMP I MMERR /SKIP ON DECTAPE DATA FLAG \\
& MMDF \\
& SKP \\
& JMP I MMDATA \\
& MMBF /SKIP ON DECTAPE BLOCK END FLAG ERROR \\
& HLT /UNLESS OTHER OPTIONS ATTACHED \\
& JMP I MMBLF
\end{align*}
\]

4.1.2 The tag "DISMIS" must be defined in the main program as a jump to the instructions which restore the link bit and accumulator and reenable the interrupt or channel. (The examples assume the interrupt routines were coded as in Section 4.1.1 above.)

For the standard program interrupt control:

\[
\begin{align*}
& DISMIS = JMP. \\
& LAC 0 /RESTORE LINK \\
& RAL /RESTORE ACCUMULATOR \\
& LAC ACSAVE /ENABLE INTERRUPT \\
& ION \\
& JMP I 0 /RETURN TO MAIN PROGRAM
\end{align*}
\]

For automatic priority interrupt:

\[
\begin{align*}
& DISMIS = JMP. \\
& LAC X /RESTORE LINK \\
& RAL /RESTORE ACCUMULATOR \\
& LAC ACSAVE /DEBREAK, ENABLE CHANNEL \\
& DBR \\
& JMP I X /RETURN TO MAIN PROGRAM
\end{align*}
\]
4.1.3 In order to differentiate between programs using the program interrupt and automatic priority interrupt facilities, the main program must contain a register named "MMAPII" containing a + 0 if the program interrupt is used, and any nonzero word if the automatic priority interrupt is used. Since it is not destroyed or changed MMAPII may be defined as equal to any other register which always contains the zero or nonzero word as required.

4.1.4 The subroutines assume that DECtape is attached to channel 3 if the automatic priority interrupt is used. If attached to any other channel, the register named "MMAPIC", within the subroutines themselves, must be modified to contain a 1 bit in one of the bits 2-17 (representing channels 0-15 respectively) to indicate the channel.

For example:
- If MMAPIC contains "1" channel 15 is used
- If MMAPIC contains "100000" channel 0 is used
- If MMAPIC contains "40" channel 10 is used

MMAPIC need not be changed if the program interrupt, or channel 3 of the automatic priority interrupt, is used.

4.1.5 The program interrupt or automatic priority interrupt (and channel) will be enabled by the subroutines themselves whether or not they were enabled by the user previously. The main program must guarantee that no flags can come up (or be up) from devices which are not checked by the user's interrupt service routine (as outlined in Section 4.1.1).

4.2 Calling Sequence
To transfer information the following calling sequence must be used:

- JMS MMRDS /OR JMS MMWRS  See Section 4.2.1
- LAC BLOCK /BLOCK NUMBER  See Section 4.2.2
- JMP XX /ERROR RETURN  See Section 4.2.3
- ZZ0000 /UNIT  See Section 4.2.4
- C1 /FIRST ADDRESS  See Section 4.2.5
- C2 /LAST ADDRESS  See Section 4.2.6
- RETURN /MULTIPROGRAMMING RETURN  See Section 4.2.7

4.2.1 The JMS MMRDS instruction is used for reading; the JMS MMWRS instruction is used for writing.

4.2.2 The DECtape block number on which the information transfer is to begin can be loaded into the accumulator with a LAC instruction as shown (where BLOCK is any register containing the correct block number), or with a LAW instruction containing the correct block number. The user should always assume the information is being transferred in the forward direction. As the instruction is executed, the location cannot contain just the block number itself. The low order twelve bits of the block number are examined, however, only block numbers 1 through 1100\text{B} are acceptable to the subroutines.

4.2.3 The JMP XX instruction is the instruction executed should any type of error occur. The accumulator contains a code indicating the type of error which occurred and location MMRSA contains the status of the DECtape system (obtained by means of an MMRS instruction) at the time of the error. The error may be detected in either the main program level or interrupt level of the program and, therefore, the interrupt system or the particular channel used will be disabled when this instruction is executed.
NOTE: If the main program is normally in extend mode while the DECtape is running, the error return must be a JMP I (XX+400000) so that the extend mode will be restored if an error occurs.

At the time the instruction is executed, the contents of the accumulator can be interpreted as follows:

<table>
<thead>
<tr>
<th>Contents of Accumulator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAW 100</td>
<td>Illegal format. Block number or core locations requested were illegal.</td>
</tr>
<tr>
<td>LAW 200</td>
<td>Block requested cannot be found.*</td>
</tr>
<tr>
<td>LAW 300</td>
<td>The DECtape error flag was raised while searching for other than an end-of-tape condition.*</td>
</tr>
<tr>
<td>LAW 400</td>
<td>The DECtape error flag was raised while reading.**</td>
</tr>
<tr>
<td>LAW 500</td>
<td>The calculated checksum does not agree with the checksum read from tape.**</td>
</tr>
<tr>
<td>LAW 600</td>
<td>The DECtape error flag was raised while writing.**</td>
</tr>
<tr>
<td>LAW 700</td>
<td>The block number read was not the block mark number predicted, while reading or writing.**</td>
</tr>
</tbody>
</table>

At the present the DECtape error flag can only be raised by end-of-tape, a timing error (the program did handle data fast enough), or a mark-track error.

4.2.4 ZZ represents the unit number (1-10) which must be placed in bits 2 through 5 of the register. Only those bits are examined.

4.2.5 C1 represents the 15-bit address of the first core location to be read into or written from (always assuming the data is transferred in the forward direction). It can be any address in normal or extended memory. (Only 15 bits are examined.)

4.2.6 C2 represents the 15-bit address of the last core location (inclusive) to be read into or written from (again assuming the data is transferred in the forward direction). C2 must be equal to or greater than C1. (Only 15 bits are examined.) The area transferred should not normally include the subroutines themselves or location 0. The subroutines are not designed to read over themselves.

Since each block written contains its own checksum, the area read need not be the same as that written. For example, if the user requested that locations 1000-3777 be written beginning with block 100, he could at some future time request that locations 2000-2777 be read beginning with block 102.

---

*The number of the block being searched for can be found in the register called MMBLKM. The block mark number last read can be found in the location whose address is contained in MMWA1, (i.e., it has been stored with a DAC I MMWA1 instruction).

**The block number last read can be found in the location whose address is contained in MMWA1, (i.e., it has been stored with a DAC I MMWA1 instruction).
Any number of words may be transferred. If a nonintegral number of blocks is specified, the following takes place:

If reading, the correct number of words will be deposited in memory and the remainder of the last block will be read but not deposited in order to verify the checksum.

If writing, the remainder of the last block will be filled with +0's and a correct checksum written.

4.2.7 As soon as searching starts, the subroutines return to the register marked "return" with the interrupt enabled. If necessary, this allows the programmer to continue processing while both the searching and data transfer takes place. In terms of usable programming time, the user has approximately 200 msec + 53 msec per block searched + 35 msec per block transferred which can be used after the subroutines are initially called.

The register named MMDONE is set to a +0 after each block mark is passed and to -0 when the data transfer is complete or if an error occurs. This allows the user three possible ways of determining when the transfer has been completed:

1. ISZ MMDONE
   JMP ? /NOT DONE

2. LAM
   SAD MMDONE
   JMP ? /DONE

3. LAC
   SNA
   JMP ? /NOT DONE ETC.

Method 1 has the advantage of not destroying the accumulator. However, if for any reason the DECTape data flag did not occur as it should, the ISZ would skip incorrectly after approximately 1.4 sec.

If the user should call the DECTape subroutines before a previous DECTape transfer has been completed, the subroutines will remain in a wait loop and not return to the main program until the first transfer has been completed and the second has begun.

It sometimes is necessary to determine what is the next forward block number on the tape, after the information just transferred. The following sequence of instructions places the correct block number in the accumulator:

LAW 61 /SEARCH BACK COMMAND
SAD MMWA3 /LAST COMMAND ISSUED
JMP
LAM -2 /TO SUBTRACT 3
TAD I MMWA1 /ADD CURRENT POSITION

LAC MMBLKM /BLOCK SEARCHED FOR
ADD (1)

4.3 Switch Settings
None
4.4 Start Up and/or Entry

See Section 4.2, Calling Sequence

4.5 Errors in Usage

Only one HLT exists in the subroutines:

<table>
<thead>
<tr>
<th>HLT Location</th>
<th>Meaning</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMERRX+1</td>
<td>Error return parameter was not a JMP instruction and an error occurred. Type of error is indicated by the number in the accumulator. (See Section 4.2.3)</td>
<td>Correct the calling sequence to provide a JMP instruction for the error return.</td>
</tr>
</tbody>
</table>

5. RESTRICTIONS

None except those mentioned in the preceding paragraphs. The standard tape format of 1100 usable blocks of 400 words each must be used.

6. DESCRIPTION

6.1 Discussion

The subroutines attempt to make variable length DECTape data transfers as easy and efficient as possible. They are completely self-contained, include only one possible error halt, indicate all possible errors which can occur, and allow fastest access to the DECTape itself. The last is accomplished by keeping track of the current position of each DECTape drive being used, calculating the effective starting and ending block numbers of the transfer requested, and starting the search in the direction causing the least number of turnarounds. Thus the direction of the data transfer is predetermined before the tape is started and does not depend on the first block actually read during searching. If the tape is currently sitting within the area to be used by the data transfer, the ultimate transfer direction will be determined by which end of the DECTape area is nearest. The current position of the tape is always assumed to be the last block number read ± 3 blocks depending on the direction of the last transfer. Initially all tapes are assumed to be sitting at block number 3.

The main thing to remember is that the user need never worry about the actual direction-transfer of the data since data always appears in memory or on tape as if it were transferred in the forward direction. For example, assuming the user has requested that locations 1000 through 1477 be written beginning at block 100, the tape appears as follows irrespective of the direction in which it was written:

```
<table>
<thead>
<tr>
<th>BLOCK 100</th>
<th>BLOCK 101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locations</td>
<td>Locations</td>
</tr>
<tr>
<td>100 - 1377</td>
<td>1400 - 1477</td>
</tr>
<tr>
<td></td>
<td>300 filler words</td>
</tr>
</tbody>
</table>
```

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>
If the data is transferred in the forward direction, the core locations are written first in ascending order followed by 300 filler words. If the data is transferred in reverse, 300 filler words are written first followed by the core locations in descending order. In either case, the end result is the same, and the technique is applicable to both reading and writing.*

6.2 Examples
The search routine has been rewritten to allow four different entrances:

1. Start the tape in the forward direction and exit when the correct block is found in the forward direction.
2. Start the tape in the reverse direction and exit when the correct block is found in the reverse direction.
3. Start the tape in the forward direction and exit when the correct block is found in the reverse direction.
4. Start the tape in the reverse direction and exit when the correct block is found in the forward direction.

In the following example assume the tape is sitting at the locations indicated by the letters shown, and a request is made to transfer 20018 words beginning at block 100:

<table>
<thead>
<tr>
<th>76</th>
<th>77</th>
<th>100</th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
<th>105</th>
<th>106</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Actual area used by the data transfer

Case A  Block 77 or less  The program will use entrance 1
Case B  Closest to initial block  The program will use entrance 4
Case C  Equidistant from start or end  The program will use entrance 4
Case D  Closest to final block  The program will use entrance 3
Case E  Block 105 or more  The program will use entrance 2

7. METHODS
See Description, Section 6.

8. FORMAT (Not Applicable)

9. EXECUTION TIME (Not Applicable)

*Should it ever become necessary to determine which way a block on tape was actually written, the following method can be used. If written in the forward direction, a checksum of -0 will appear at the front of the block (the end near the preceding lower-numbered block). If written in reverse, the -0 checksum will be at the end of the block near the next higher-numbered block.
10. PROGRAM

10.4. Program Listing

/DIRECTIONAL PDP-7 DECTAPE SUBROUTINES
/ASSUMES STANDARD 400 (OCTAL) WORD BLOCKS
/JANUARY 21, 1966
/DIS=IS MUST BE DEFINED AS JMP TO DISMISS INTERRUPT ROUTINE

/PDP-7 DECTAPE SEARCH SUBROUTINE

MMWR=707504
MMRC=707604
MMSE=707644
MMRS=707612
MMDF=707661
MMBF=707681
MMEF=707541
MMRD=707512
SKP7=703341
ASC=705502
API=700044
DVR=705601
LEQ=707704
EEM=707702
EMRE=707742

/LEAVE IN SEARCH REVERSE MODE, START REVERSE
MMSCHR,  LAW 41  /SEARCH FWD
    DAC MMWA3  /SET CURRENT DIRECTION
    CLA
    JMP MMSCHR8

/LEAVE IN SEARCH REVERSE MODE, START FORWARD
MMSER,  LAW 61  /SEARCH REVERSE
    JMP MMSCHR+1

/LEAVE IN SEARCH FORWARD MODE, START REVERSE
MMSER,  LAW 41  /SEARCH FORWARD
    SKP

/LEAVE IN SEARCH FORWARD MODE, START FORWARD
MMSCHR0,  LAW 61  /SEARCH REVERSE, USED AS CONSTANT
    DAC MMWA3  /SET CURRENT DIRECTION
    LAM
    DAC MMSBK
    TAD (1)
    DAC MMSFK
    LAW MMERS
    DAC MMERR
    LAW MMDATS
    DAC MMDATA
    LAC MMRLKM
    ADD MMEK
    SMA

MMSCHR8,
JMP MMSC#5  "FORMAT ERROR"
LAM -7  "CHG OF DIRECTION COUNTER"
DAC MMSSUM

/35 MILLISECOND SELECT DELAY LOOP
MMWAIT,
MRS
AND (400)  "SAVE CONTROL TYPE ONLY"
SZA
JMP MMSC#9  "NO DELAY FOR NEW DRIVES"
LAC MMCHK1+1  "PICK UP SELECT"
SAD MMSEL  "PREVIOUS SELECT"
JMP MMSC#9+1  "SAME SELECT"
DAC MMSEL  "SAVE SELECT"

MM2,
CLA
MMSE
LAM DECIMAL -5000+1 OCTAL
DAC MMBLF  "TEMPORARY STORAGE AREA"
IS7 1 -.1  "7 MICROSECONDS PER LOOP"
JMP -.1  "COUNT 35 MILLISECONDS"
LAC MMCHK1+1  "UNIT SELECTION"
MMSE
LAC (NOP)
DAC MMSAVE
LAC MMAPI I  "DECTAPE ON API INDICATOR"
SZA
JMP MMAPI  "DN ION IF NOT ON API"
ION
MMTURN,
ISZ MMSSUM
SKP
JMP MMERX2=1  "NOT FOUND"
LAC MMWA3  "CURRENT SELECT"
XOR (20)  "COMPLEMENT DIRECTION"
MMLC
DAC MMWA3  "SEARCH IN CORRECT DIRECTION"
SAD MMSC#0  "SAVE SELECTION"
JMP MMREV  "SET UP REVERSE CONSTANTS"
LAC MMCK3  "SHA, SET TO CONTINUE IN FORWARD DIRECTION"
DAC MMSC#2
LAC MMBLKM
TAD MMSFK
DAC MMWA2
NZM MMMDONE
MMSAVE,
NOP
LAC MMRD3B+1  "OR DISMIS"
DAC MMSAVE
ISZ MMWA
EMIR
JMP 1 MMWA  "DISMIS"
LAC MMCK2  "INDEX POINTER"
DAC MMSC#2
LAC MMBLKM
TAD MMSFK
JMP MMSAVE-2  "RETURN TO MAIN PROGRAM"
/MMREV,
/INSTRUCTIONS FOR AUTOMATIC PRIORITY INTERRUPT

LAC MMAPIC /DECTAPE CHANNEL NUMBER
EPI /ENABLE API
ASC /ENABLE DECTAPE CHANNEL
JMP MMTURN

/ROUTINES TO ANSWER INTERRUPT SEQUENCE

MMERS,

MMPRS, AND (40000) /CHECK END BIT
SZA
JMP MMTURN /ENT TURN AROUND
LAW 300 /NON-END ERROR DURING SEARCH
JMP MMERRx2
MMDATS,

MMRD
DAC I M-MWA1
SAD MMWA2
JMP MMSUH3
CMA
ADD MMWA2
SMA
JMP MMSAVE-1 /KEEP GOING
JMP MMTURN /TURN AROUND
SAD MMRKLM
JMP I MMSCHK
JMP MMTURN

MMSCH2,

LAW 100 /EXIT TO READ OR WRITE ROUTINES
JMP MMERRx2
MMSCH5,

INDEX MMREX2
MMEK, DECIMAL -576 OCTAL

/ERROR LOOP

LAW 200 /NOT FOUND
DAC MMRLF /STORAGE AREA
MMRS
DAC MMRSA
JMP MMRULL
LAC MMERRx AND (20000)
SZA /JMP
EMIR /JMP I
CLC
DAC MMDONE
LAC MMRLBF
MMLC

MMERRx, JMP . /ERROR EXIT WAS NOT A JMP INSTRUCTION
MMSL,
MSEL, /SAVE SELECTION
0 /ERROR RETURN
MMEERR, /DATA RETURN
MMDATA, 0 /BLOCK FLAG RETURN
MMLBF, 0 /NORMAL DECTAPE CHANNEL=3
MMAPIC, 10000
MMWAX, 3 /POSITION OF UNIT 1
3 /POSITION OF UNIT 2
3 /POSITION OF UNIT 3
3 /POSITION OF UNIT 4
3 /POSITION OF UNIT 5
3 /POSITION OF UNIT 6
3

/POSITION OF UNIT 7
3

/POSITION OF UNIT 10

/INTERLOCK LOOP, HANGS UP MAIN PROGRAM UNTIL GO=0
MMITLK, 0
LEH /USED FOR MMSUM
MRMR /LEAVE EXTEND MODE
AND (4000) /GET STATUS
SZA /CHECK GO BIT
JMP -3 /NOT GOING?
JMP I MMITLK /WAIT

/LOOP TO ADD ROLL CONSTANT TO CURRENT ADDRESS
MMROLL, 0
LAC MMWAX /USED AS WORK AREA, MMSBK
AND (20) /LAST SEARCH COMMAND
SZA /SAVE DIRECTION BIT
LAM -5 /GOING FORWARD
ADD (3) /ROLL CONSTANT
TAD I MMWAX /AND CURRENT LOCATION
DAC I MMWAX /STORE CURRENT LOCATION
JMP I MMROLL

/COMMON ROUTINE FOR PICKING UP CONSTANTS AND SEARCHING
/PICK UP PARAMETERS
MMCHK, 0
XCT I MMWAX /BLOCK NUMBER
AND (7777) /SAVE BLOCK NUMBER
DAC MMRLKM /INDEX POINTER TO ERROR RETURN
ISZ MMWAX /ERROR RETURN
LAC I MMWAX /IN SEARCH EXIT
DAC MMHXR /INDEX POINTER TO UNIT
ISZ MMWAX /UNIT
LAC I MMWAX /KEEP UNIT ONLY
AND (17000) /IN CALLING SEQUENCE
DAC MMCHK+1
RCL /CLEAR AND ROTATE LINK
RTL /PUT UNIT NUMBER IN L.O.S.POINTE R
RTL
ADD (MMWAX+1)
DAC MMWAX /POSITION POINTER
LAC I MMWAX /FOR THIS UNIT
AND (77777) /INDEX POINTER TO STARTING ADDRESS
DAC MMADDR /ERROR ADDRESS
LAC I MMWAX /STARTING ADDRESS
ISZ MMWAX
AND (77777) /15 BIT ADDRESS
DAC MMADDR /LOCATION POINTER
ISZ MMWAX
DAC MMWDC /INDEX POINTER TO ENDING ADDRESS

/_CALCULATE NUMBER OF DATA AND FILLER WORDS
LAC I MMWAX /FINAL ADDRESS
AND (77777) /15 BIT ADDRESS
CMA
ADD MMADDR
SMA
JMP MMSCH5
DAC MMWDC /11 LEGAL FORMAT
-NO. OF DATA WORDS+1
AND (377) /LOW ORDER 8 BITS
XOR (777400) /MAKE NUMBER NEGATIVE
TAN (377)
CMA
DAC MM2CN /-NO OF FILLER WORDS+1
LAM -1
DAC MMF1LC /SECTION COUNTER

/CALCULATE THE DIRECTION TO SEARCH
LAC MM-BLKM /BLOCK DESIRED
SNA
JMP MMSCCH5 /BLOCK 0, FORMAT ERROR
CMA
ADD I MMWA1 /CURRENT POSITION
SPA
JMP MM4F /CURRENT POSITION HIGHER THAN DESIRED BLOCK
DAC MMWA5 /SEARCH AND TRANSFER DATA FORWARD
LAC MM-WOC /-NUMBER OF DATA WORDS+1
CMA CALL
AND (777400) /KEEP NUMBER OF BLOCKS-1
RTR
RTR
RTR
RTR

ADD MMRLKM /STARTING BLOCK
DAC MMWA7 /LAST BLOCK
CMA
ADD I MMWA1 /CURRENT POSITION
SMA
JMP MMGH2 /SEARCH AND TRANSFER DATA IN REVERSE
ADD MMWA5 /DISTANCE TO START BLOCK
SMA /START IN REVERSE, TRANSFER DATA FORWARD
JMP MMGH /START FWD, TRANSFER DATA IN REVERSE

/START IN REVERSE, TRANSFER DATA FORWARD
LAW MM5FR
SKP

/START AND TRANSFER DATA FORWARD
MMGF,
LAW MMSCCH5
DAC MMWA5 /SET UP SEARCH ENTRANCE
LAC (1)
DAC MMDK /FOR INCREMENTING ADDRESS
MMGF2,
LAC (LAC I MMADDR) /SET READ ROUTINE
DAC MMRD3
LAC (LAC I MMADDR) /SET WRITE ROUTINE

/START SEARCH
MMGTK1,
JMP I MMWA5 /TO SEARCH
? /UNIT AND WORK AREA MMWA2
/START FORWARD, TRANSFER DATA IN REVERSE
MMGR,      LAW MMSRF
SKP

/START AND TRANSFER DATA IN REVERSE
MMGR2,     LAW MMSCHM
DAC MMWA5  /SET UP SEARCH ENTRANCE
LAM
DAC MMOK   /TO DECREMENT ADDRESS
LAC MM2CN  /FILLER COUNTER
SZAVCMF    /*IF THERE ARE NO FILLERS
ADD (1)
ADD I MMWA /ENDING ADDRESS
DAC MMADDH /DATA LOCATION POINTER
LAC MMWA7  /SEARCH FOR LAST BLOCK
DAC MMRLKM
LAC MM2CN
SNA
JMP MMR3  /NO FILLERS
DAC MMWA6  /EXCHANGE MMWC AND MM2CN
LAC MMWDC
DAC MM2CN
LAC MMWA6
DAC MMWDC
LAC (NOP)
DAC MMRD3 /FOR FILLERS IN RD ROUTINE
LAC MM22 /FOR FILLERS IN WRITE ROUTINE, CLA
JMP MMRH1-1

MMGR3,     ISZ MMFILC
JMP MMGF2

/DIAG COMP SUBROUTINE, READ PDP-7
/FORMAT        JMS MMRF5
    /  JMS MMRF5
    /  JMP X  /ERROR RETURN
    /  ZZ0000  /UNIT SELECTION
    /  CI  /16-BIT CORE STARTING ADDRESS
    /  C2  /16-BIT CORE ENDING ADDRESS, INCLUSIVE
    /  MULTI-PROGRAM RETURN

MMRF5,      0
    JMS MMTILK  /CHECK IF SYSTEM IS FREE
    LAC MMRF5
DAC MMWA  /STORE POINTER TO ARGUMENTS
    JMS MMRF6  /GET ARGUMENTS AND SEARCH

/RETURN FROM SEARCH WITH BLOCK FOUND
    LAW MMRU1 /SET UP INTERRUPT RETURNS
    DAC MMERR
    LAW MMRF4
DAC MMHALF
XCT MMWA3 /SEARCH COMMAND
ADD (1)  /MAKE READ COMMAND
    MMRF5
LAW MMRF1A
DAC MMDATA
NZM MMDONE
DISMIS
Digital-7-22A-1/O
Page 14

MMRD1:
LAW 40H
JMP MMEM0X2
/MMRD
DAC MMSUM
LAW MMMU2
DAC MMDATA
DISMIS
/MMRD
/LA,

MMRD2:
/MMRD
EEM
/READ DATA
LEM
/MMRD
/LA,

MMRD3:
/DAC
/MMRD
/EEM
/MEM
/ENABLE EXTENDED MEMORY
/JMP
/READ REVERSE CHECKSUM
/LAC (NOP)
/DISABLE EXTENDED MEMORY

MMRD3A:
/IST
/MMRD
/CALCULATE CHECKSUM
/JMP
/SFCTION COUNTER
/LAC (NOP)

MMRD3B:
/DAC
/MMRD
/NOT STORE REMAINDER OF BLOCK
/DISMIS
/USED AS CONSTANT
/MMRD
/FORWARD CHECKSUM

MMRD4:
ADD MMSUM
SAD MMSCHB-1
/JMP +3
/READ FORWARD CHECKSUM
/LAW 50H
/JMP MMEM0X2
/LMS
/MMRD
/LMS
/MMRD
/SUM CHECK READING
/MMRD
/NEXT BLOCK NUMBER
/MMRD
/READ NEXT BLOCK

MMRD5:
/LAC MM-20H
/SNA
/JMP MMRD3A
/DAC MMWUC
/NOM FillERS
/LAC (DAC I MMADDR)
/SF'T UP WORD COUNTER
/SAD MMRD3
/NOT STORE NOP
/MMRD
/STORE DAC INSTRUCTION

/MMRD6:
/ADD TO CHECKSUM AND INCREMENT OR
/DISCRETE ADDRESS
/MMRD6:
/SAD MMSUM
/MMRD6:
/PREVIOUS CALCULATION
/DAC MMSUM
/MMRD6:
/STORE NEW RESULT
/LAC MMADV
/MMRD6:
/CURRENT ADDRESS
/TAN MMADK
/MMRD6:
/+1 OR -1
/DAC MMADV
/MMRD6:
/NEW ADDRESS
/IST MMDUC
/MMRD6:
/WORD COUNTER
/DISMIS
/MMRD
/ointments FOR FillERS ETC.

/MMRD10:
/JMP I MMRD6
/MMRD10:
/SUM OF WORK AREA, MMSFK

/MMRD12:
/LAV MM3LC02
/MMRD12:
/SF'T DATA FLAG RETURN
/DAC MMDATA
/MMRD12:
/SEARCH COMMAND
/XCT MMW4A3
/MMRDL

/MMRDL:
/LAC I MMWA1
/MMRDL:
/CURRENT BLOCK NUMBER
/LAD MMJK
/MMRDL:
/+1 OR -1
/DAC I MMWA1
/MMRDL
/MMRDL:
/Nw BLOCK NUMBER
SAD I MMWA1 /COMPARE TO CORRECT NUMBER
JMP .+3 /BLOCK MARK ERROR
LAW 700 /SECTION COUNTER
JMP MMEX2
LAC MMFILC
SZA
JMP I MMRLC /RETURN FOR NEXT BLOCK
MMC /STOP THE TAPE
JMS MMROLL /AND ROLL CONSTANT
CLC
DAC MMDONE /SFT DONE SWITCH

DEC-TAPE WRITE SUBROUTINE, PDP-7

/FORMAT
JMS MMWS
/-law R /OR LAC (8), BLOCK NUMBER
/ JMP X /ERROR RETURN
/ 200000 /UNIT SELECTION
/ C1 /16-BIT CORE STARTING ADDRESS
/ C2 /16-BIT ENDING ADDRESS, INCLUSIVE
/MULTI-PROGRAM RETURN

MMWS,
0
JMS MMITLK /CHECK IF SYSTEM IS FREE
JMS MMCHK /PICK UP ARGUMENTS AND SEARCH

/RETURN FROM SEARCH WITH BLOCK FOUND
LAW MMWR2 /SFT UP INTERRUPT RETURNS
DAC MMERR
LAW MMWR4
DAC MMRF
LAW MMWR3
DAC MMDATA
D2M MMDONE
CLC
DAC MMSCM /START CHECKSUM
XCT MMWA3 /SEARCH COMMAND
ADD (2) /CREATE WRITE COMMAND
MMC
DISMIS

MMWR2,
LAW 600 /ERROR FLAG DURING WRITING
JMP MMEX2

MMWR3,
EH
LAC I MMADDK /FOR CLA
LEM /DISABLE EXTENDED MODE
MMWR
JMS MMRD6 /CALCULATE CHECKSUM
ISZ MMFILC /SECTION COUNTER
JMP MMWR4 /SFT UP FOR 2ND SECTION
LAC MMU2 /CLA
MMWR3A
DAC MMWR3+1 /DISMIS

MMWR4,
LAC MMSCM /WRITE CHECKSUM
CMA
MMWR
JMS MMRLC
JMP MMWR1 /CHECK NEXT BLOCK NUMBER
Digital-7-22A-I/O
Page 16

START

11. DIAGRAMS

11.1 Flow Charts

Gross DECTape Subroutine
Read Routine
Read Routine (continued)
Read Routine (continued)
Write Routine
Write Routine (continued)

Digital-7-22A-I/O
Page 21
Common Routine to Pick Up Parameters and Initiate Searching
Common Routine to Pick Up Parameters and Initiate Searching (continued)
Search Routine

SEARCH AND TRANSFER
IN REVERSE

MMSCHR

SET LAST SEARCH
COMMAND TO
SEARCHING FORWARD

PUT ZERO
IN ACCUMULATOR

SEARCH FORWARD
TRANSFER IN REVERSE

MMSRF

SET LAST SEARCH
COMMAND TO
SEARCHING IN
REVERSE

SEARCH IN REVERSE
TRANSFER FORWARD

MMSFR

SET LAST SEARCH
COMMAND TO
SEARCHING FORWARD

PUT -8
IN ACCUMULATOR

SEARCH AND TRANSFER
FORWARD

MMSCHF

SET LAST SEARCH
COMMAND TO
SEARCHING IN
REVERSE

STORE AS CONSTANT
(MMSKR) FOR
SEARCHING IN
REVERSE

TAD (1), STORE
AS CONSTANT
(MMSFR) FOR
SEARCHING FORWARD

SET INTERRUPT
to RETURN TO U

LOAD MMBLMN
(BLOCK TO
SEARCH FOR)

DESELECT ALL
UNITS, WAIT 35
MILLISECONDS

UNIT DESIRED
= PREVIOUS UNIT
NUMBER?

NO

BLOCK NUMBER
< 1101?

NO

RETURN TO ERROR EXIT

NO

PICK UP
ERROR CODE 108

YES

SET CHANGE
OF DIRECTION
COUNTER TO -8

YES

NEW CONTROL?

NO

MMSH5

STOP TAPE, SET DONE
INDICATOR

R

R

R

R

R

NO

YES

YES

NO

YES

YES

R

R

R

R

R
Search Routine (continued)