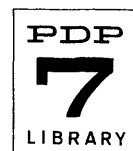


1. IDENTIFICATION
- 1.1 Digital-7-22A-I/O
- 1.2 Bidirectional PDP-7 DEctape Subroutines
- 1.3 January 21, 1966



## 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<sub>g</sub> usable blocks of 400<sub>g</sub> 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<sub>g</sub> locations for the unidirectional, 604<sub>g</sub> 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<sub>g</sub> 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 MMWAI 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.

```

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

```

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

```

X,      0
        DAC ACSAVE      /SAVE THE ACCUMULATOR
        MMEF            /SKIP ON DECTAPE ERROR FLAG
        SKP
        JMP I MMERR
        MMDF            /SKIP ON DECTAPE DATA FLAG
        SKP
        JMP I MMDATA
        MMBF            /SKIP ON DECTAPE BLOCK END FLAG ERROR
        HLT            /UNLESS OTHER OPTIONS ATTACHED
        JMP I MMBLF

```

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:

```

DISMIS = JMP.
        LAC 0
        RAL      /RESTORE LINK
        LAC ACSAVE /RESTORE ACCUMULATOR
        ION      /ENABLE INTERRUPT
        JMP I 0  /RETURN TO MAIN PROGRAM

```

For automatic priority interrupt:

```

DISMIS = JMP.
        LAC X
        RAL      /RESTORE LINK
        LAC ACSAVE /RESTORE ACCUMULATOR
        DBR      /DEBREAK, ENABLE CHANNEL
        JMP I X  /RETURN TO MAIN PROGRAM

```

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<sub>10</sub> 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<sub>8</sub> 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:

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

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		→	LAC MMBLKM /BLOCK SEARCHED FOR
LAM -2	/TO SUBTRACT 3		ADD (1)
TAD I MMWA1	/ADD CURRENT POSITION		
		←	

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:

<u>HLT Location</u>	<u>Meaning</u>	<u>Procedure</u>
MMERRX+1	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)	Correct the calling sequence to provide a JMP instruction for the error return.

5. RESTRICTIONS

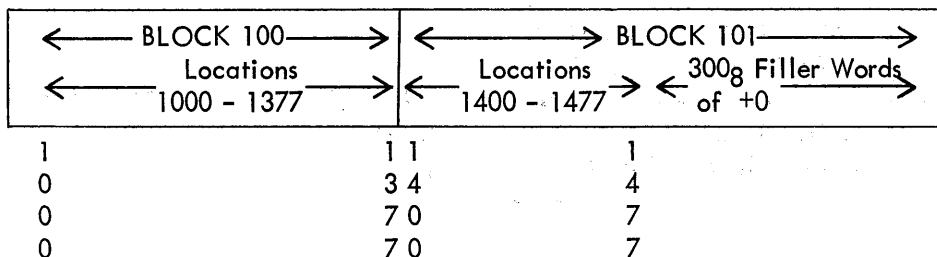
None except those mentioned in the preceding paragraphs. The standard tape format of 1100g usable blocks of 400g words each must be used.

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  $\pm 3$  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:



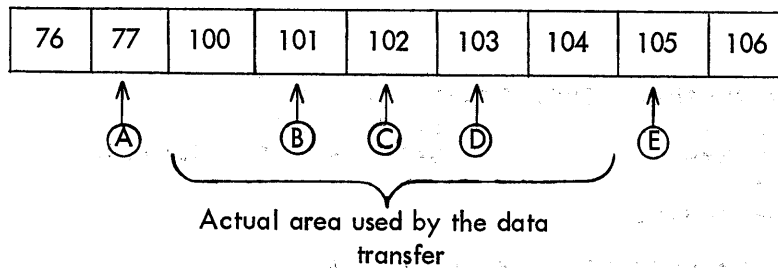
If the data is transferred in the forward direction, the core locations are written first in ascending order followed by 300<sub>8</sub> filler words. If the data is transferred in reverse, 300<sub>8</sub> filler words are written first followed by the core locations in descending order. In either case, 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 200<sub>8</sub> words beginning at block 100:



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

```

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

```

```

/PDP-7 DEC-TAPE SEARCH SUBROUTINE

```

```

MMWR=707504
MMLC=707604
MMSE=707644
MMRS=707612
MMDF=707501
MMBF=707601
MMEF=707541
MMRD=707512
SKP7=703341
ASC=705502
EPI=700044
DBR=705601
LEM=707704
EEM=707702
EMIR=707742

```

```

/LEAVE IN SEARCH REVERSE MODE, START REVERSE
MMSCHR, LAW 41 /SEARCH FWD
DAC MMWA3 /SET CURRENT DIRECTION
CLA
JMP MMSCH8

```

```

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

```

```

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

```

```

/LEAVE IN SEARCH FORWARD MODE, START FORWARD
MMSCH0, LAW 61 /SEARCH REVERSE, USED AS CONSTANT
DAC MMWA3 /SET CURRENT DIRECTION
LAM /LOAD -0, USED AS CONSTANT
MMSCH8, DAC MMSBK
TAD (1)
DAC MMSFK
LAW MMERS /SET UP INTERRUPT RETURN
DAC MMERR
LAW MMDATS
DAC MMDATA
LAC MMBLKM /PICK UP BLOCK NUMBER
ADD MMEK
SMA

```

```

JMP MMSCH5          /FORMAT ERROR
LAM -7
DAC MMSUM          /CHG OF DIRECTION COUNTER

/35 MILLISECOND SELECT DELAY LOOP
MMWAIT,           MMS
                  AND (400)          /SAVE CONTROL TYPE ONLY
                  SZA
                  JMP MMSCH9         /NO DELAY FOR NEW DRIVES
                  LAC MMCHK1+1       /PICK UP SELECT
                  SAD MMSSEL         /PREVIOUS SELECT
                  JMP MMSCH9+1       /SAME SELECT
                  DAC MMSSEL         /SAVE SELECT
MMW2,             CLA                 /USED AS CONSTANT
                  MMSE              /SELECT UNIT ZERO
                  LAM DECIMAL -5000+1 OCTAL
                  DAC MMBLF         /TEMPORARY STORAGE AREA
                  ISZ I .-1         /7 MICROSECONDS PER LOOP
                  JMP .-1           /COUNT 35 MILLISECONDS
MMSCH9,          LAC MMCHK1+1       /UNIT SELECTION
                  MMSE
                  LAC (NOP)
                  DAC MMSAVE
                  LAC MMAPII        /DECTAPE ON API INDICATOR
                  SZA              /DO ION IF NOT ON API
                  JMP MMAPI         /TURN ON API
                  ION

MMTURN,          ISZ MMSUM
                  SKP
                  JMP MMERX2-1      /NOT FOUND
                  LAC MMWA3         /CURRENT SELECT
                  XOR (20)          /COMPLEMENT DIRECTION
                  MMLC             /SEARCH IN CORRECT DIRECTION
                  DAC MMWA3         /SAVE SELECTION
                  SAD MMSCH0        /LAW 61
                  JMP MMREV         /SET UP REVERSE CONSTANTS
                  LAC MMCK3         /SMA, SET TO CONTINUE IN FORWARD DIRECTION
                  DAC MMSCH2
                  LAC MMBLKM
                  TAD MMSFK
                  DAC MMWA2         /BLOCK TO LOOK FOR IN THIS DIRECTION
                  DZM M+MDONE

MMSAVE,          NOP                /OR DISMIS
                  LAC MMRD3B+1     /DISMIS
                  DAC MMSAVE
                  ISZ MMWA         /INDEX POINTER
                  EMIR

MMREV,           JMP I MMWA         /RETURN TO MAIN PROGRAM
                  LAC MMCK2        /SPA, SET TO CONTINUE IN REVERSE DIRECTION
                  DAC MMSCH2
                  LAC MMBLKM
                  TAD MMSBK
                  JMP MMSAVE-2

```

/INSTRUCTIONS FOR AUTOMATIC PRIORITY INTERRUPT

```

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

/ROUTINES TO ANSWER INTERRUPT SEQUENCE

```

MMERS,      MMRS
            AND (40000)    /CHECK EOT BIT
            SZA
            JMP MMTURN    /EOT, TURN AROUND
            LAW 300      /NON-EOT ERROR DURING SEARCH
            JMP MMERX2
MMDATS,     MMRD
            DAC I M-MWA1
            SAD MMWA2
            JMP MMSCH3
            CMA
            ADD MMWA2
MMSCH2,     SMA          /OR SPA FOR REVERSE
            JMP MMSAVE-1 /KEEP GOING
            JMP MMTURN   /TURN AROUND
MMSCH3,     SAD MMRLKM
            JMP I MMCHK  /EXIT TO READ OR WRITE ROUTINES
            JMP MMTURN
MMSCH5,     LAW 100
            JMP MMERX2   /FORMAT ERROR
MMEK,       DECIMAL -576 OCTAL
    
```

/ERROR LOOP

```

MMERX2,     LAW 200      /NOT FOUND
            DAC MMBLF    /STORAGE AREA
            MMRS
            DAC MMRSA
            JMS MMRULL   /ADD ROLL CONSTANT
            LAC MMERRX
            AND (20000)
            SZA          /JMP
            EMIR        /JMP I
            CLC
            DAC MMDONE
            LAC MMBLF    /STORAGE AREA
            MMLC
MMERRX,     JMP .
            HLT
MMSEL,      0           /ERROR EXIT WAS NOT A JMP INSTRUCTION
MMERR,      0           /SAVE SELECTION
MMDATA,     0           /ERROR RETURN
MMBLF,      0           /DATA RETURN
MMAPIC,     10000      /BLOCK FLAG RETURN
MMWAX,      3           /NORMAL DECTAPE CHANNEL=3
            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          /USED FOR MMSUM
           LEM        /LEAVE EXTEND MODE
           MMRS       /GET STATUS
           AND (4000) /CHECK GO BIT
           SZA        /NOT GOING?
           JMP .-3    /WAIT
           JMP I MMITLK /SYSTEM AVAILABLE

/LOOP TO ADD ROLL CONSTANT TO CURRENT ADDRESS
MMROLL,    0          /USED AS WORK AREA, MMSBK
           LAC MMWA3  /LAST SEARCH COMMAND
           AND (20)   /SAVE DIRECTION BIT
           SZA        /GOING FORWARD
           LAM -5     /TO CREATE LAM -2
           ADD (3)    /ROLL CONSTANT
           TAD I MMWA1 /ADD CURRENT LOCATION
           DAC I MMWA1 /STORE CURRENT LOCATION
           JMP I MMROLL

/COMMON ROUTINE FOR PICKING UP CONSTANTS AND SEARCHING
/PICK UP PARAMETERS
MMCHK,     0
           XCT I MMWA /BLOCK NUMBER
           AND (7777)
           DAC MMRLKM /SAVE BLOCK NUMBER
           ISZ MMWA   /INDEX POINTER TO ERROR RETURN
           LAC I MMWA /ERROR RETURN
           DAC MMERRX /IN SEARCH EXIT
           ISZ MMWA   /INDEX POINTER TO UNIT
           LAC I MMWA /UNIT
           AND (170000) /KEEP UNIT ONLY
           DAC MMCHK1+1 /IN CALLING SEQUENCE
           RCL        /CLEAR AND ROTATE LINK
           RTL        /PUT UNIT NUMBER IN L.O.POSITION
           RTL
           RTL
           ADD (MMWAX-1)
           DAC MMWA1  /ADDRESS OF POSITION POINTER
                       /FOR THIS UNIT
           ISZ MMWA   /INDEX POINTER TO STARTING ADDRESS
           LAC I MMWA /STARTING ADDRESS
           AND (77777) /15 BIT ADDRESS
           DAC MMADDR /LOCATION POINTER
           ISZ MMWA   /INDEX POINTER TO ENDING ADDRESS

/CALCULATE NUMBER OF DATA AND FILLER WORDS
           LAC I MMWA /FINAL ADDRESS
           AND (77777) /15 BIT ADDRESS
           CMA
           ADD MMADDR /STARTING ADDRESS
           SMA
           JMP MMSCH5  /ILLEGAL FORMAT
           DAC MMWDC  /-NO. OF DATA WORDS+1

```

```

AND (377)           /LOW ORDER 8 BITS
XOR (777400)       /MAKE NUMBER NEGATIVE
TAD (377)
CMA
DAC MM2CN          /-NO OF FILLER WORDS+1
LAM -1
DAC MMFILC        /SECTION COUNTER
/CALCULATE THE DIRECTION TO SEARCH
LAC MM→BLKM       /BLOCK DESIRED
SNA
JMP MMSCH5        /BLOCK 0, FORMAT ERROR
CMA
MMCK2, ADD I MMWA1  /CURRENT POSITION
SPA              /CURRENT POSITION HIGHER THAN DESIRED BLOCK
JMP MMGF         /SEARCH AND TRANSFER DATA FORWARD
DAC MMWA5        /DISTANCE TO START BLOCK
LAC MM→WDC       /-NUMBER OF DATA WORDS+1
CMAVCLL         /NUMBER OF DATA WORDS-1
AND (777400)     /KEEP NUMBER OF BLOCKS=1
RTR              /DIVIDE BY 400 OCTAL
RTR
RTR
RTR

ADD MMBLKM       /STARTING BLOCK
DAC MMWA7        /LAST BLOCK
CMA
MMCK3, ADD I MMWA1  /CURRENT POSITION
SMA              /CURRENT POSITION IS WITHIN TRANSFER SECTION
JMP MMGR2        /SEARCH AND TRANSFER DATA IN REVERSE
ADD MMWA5        /DISTANCE TO START BLOCK
SMA              /START IN REVERSE, TRANSFER DATA FORWARD
JMP MMGR         /START FWD, TRANSFER DATA IN REVERSE

/START IN REVERSE, TRANSFER DATA FORWARD
LAW MMSFR
SKP

/START AND TRANSFER DATA FORWARD
MMGF, LAW MMSCH0
DAC MMWA5        /SET UP SEARCH ENTRANCE
LAC (1)
DAC MMDK         /FOR INCREMENTING ADDRESS
MMGF2, LAC (DAC I MMADDR) /SET READ ROUTINE
DAC MMRD3
LAC (LAC I MMADDR) /SET WRITE ROUTINE
DAC MMWR3+1

/START SEARCH
MMCHK1, JMP I MMWA5  /TO SEARCH
0          /UNIT AND WORK AREA MMWA2

```

```

/START FORWARD, TRANSFER DATA IN REVERSE
MMGR,      LAW MMSRF
           SKP

```

```

/START AND TRANSFER DATA IN REVERSE
MMGR2,     LAW MMSCHR
           DAC MMWA5           /SET UP SEARCH ENTRANCE
           LAM
           DAC MMDK           /TO DECREMENT ADDRESS
           LAC MM2CN          /FILLER COUNTER
           SZAVCA            /+0 IF THERE ARE NO FILLERS
           ADD (1)
           ADD I MMWA         /ENDING ADDRESS
           DAC MMADDR        /DATA LOCATION POINTER
           LAC MMWA7         /SEARCH FOR LAST BLOCK
           DAC MMRLKM
           LAC MM2CN
           SNA
           JMP MMGR3         /NO FILLERS
           DAC MMWA6         /EXCHANGE MMWDC AND MM2CN
           LAC MMWDC
           DAC MM2CN
           LAC MMWA6
           DAC MMWDC
           LAC (NOP)
           DAC MMRD3         /FOR FILLERS IN RD ROUTINE
           LAC MMW2 /FOR FILLERS IN WRITE ROUTINE, CLA
           JMP MMCHK1-1

```

```

MMGR3,     ISZ MMFILC
           JMP MMGF2
/DECTAPE SUBROUTINE, READ PDP-7
/FORMAT    JMS MMRDS
/          LAW B             /OR LAC (B), BLOCK NUMBER
/          JMP X             /ERROR RETURN
/          ZZ0000           /UNIT SELECTION
/          C1               /15-BIT CORE STARTING ADDRESS
/          C2               /15-BIT CORE ENDING ADDRESS, INCLUSIVE
/          MULTI-PROGRAM RETURN

```

```

MMRDS,     0
           JMS MMITLK        /CHECK IF SYSTEM IS FREE
           LAC MMRDS
           DAC MMWA         /STORE POINTER TO ARGUMENTS
           JMS MMCHK        /GET ARGUMENTS AND SEARCH

```

```

/RETURN FROM SEARCH WITH BLOCK FOUND
           LAW MMRD1        /SET UP INTERRUPT RETURNS
           DAC MMERR
           LAW MMRD4
           DAC MMBLF

```

```

MMRD0,     XCT MMWA3        /SEARCH COMMAND
           ADD (1)         /MAKE READ COMMAND
           MMLC
           LAW MMRD1A
           DAC MMDATA
           DZM MMDONE
           DISMIS

```

```

MMRD1,      LAW 400              /ERROR FLAG DURING READING
             JMP MMERY2
MMRD1A,     MMRD                /READ REVERSE CHECKSUM
             DAC MMSUM
             LAW MMRD2
             DAC MMDATA
             DISMIS
MMRD2,     MMRD                /READ DATA
             EEM                /ENABLE EXTENDED MEMORY
MMRD3,     DAC I MM+ADDR        /OR NOP
             LEM                /DISABLE EXTENDED MEMORY
             JMS MMRD6          /CALCULATE CHECKSUM
MMRD3A,    ISZ MM+FILC         /SECTION COUNTER
             JMP MMRD5          /SET UP FOR SECTION
             LAC (NOP)
MMRD3B,    DAC MMRD3           /DO NOT STORE REMAINDER OF BLOCK
             DISMIS           /USED AS CONSTANT
MMRD4,     MMRD                /READ FORWARD CHECKSUM
             ADD MMSUM
             SAD MMSCH8-1      /-2
             JMP .+3
             LAW 500           /SUM CHECK READING
             JMP MMERY2
             JMS MMBLC        /CHECK NEXT BLOCK NUMBER
             JMP MMRD0        /READ NEXT BLOCK
MMRD5,     LAC MM+20N          /2ND SECTION COUNTER
             SNA
             JMP MMRD3A        /NO FILLERS
             DAC MMWDC        /SET UP WORD COUNTER
             LAC (DAC I MMADDR)
             SAD MMRD3
             JMP MMRD3B-1      /STORE NOP
             JMP MMRD3B       /STORE DAC INSTRUCTION
/ADD TO CHECKSUM AND INCREMENT OR DECREMENT ADDRESS
MMRD6,     0                    /USED AS WORK AREA, MMWA5
             ADD MMSUM         /PREVIOUS CALCULATION
             DAC MMSUM         /STORE NEW RESULT
             LAC MMADDR        /CURRENT ADDRESS
             TAD MMDK          /+1 OR -1
             DAC MMADDR        /NEW ADDRESS
             ISZ MMWDC         /WORD COUNTER
             DISMIS
             JMP I MMRD6       /CHECK FOR FILLERS ETC.
/CHECK NEXT BLOCK MARK NUMBER
MMBLC,     0                    /USED AS WORK AREA, MMSFK
             LAW MMBLC2
             DAC MMDATA        /SET DATA FLAG RETURN
             XCT MMWA3         /SEARCH COMMAND
             MMLC
MMBLC2,    DISMIS
             LAC I MMWA1       /CURRENT BLOCK NUMBER
             TAD MM+DK         /+1 OR -1
             DAC I MMWA1       /NEW BLOCK NUMBER
             MMRD

```

```

SAD I MMWA1          /COMPARE TO CORRECT NUMBER
JMP .+3
LAW 700             /BLOCK MARK ERROR
JMP MMERX2
LAC MMFILC         /SECTION COUNTER
SZA
JMP I MMBLC        /RETURN FOR NEXT BLOCK
MMLC              /STOP THE TAPE
JMS MMRLL         /ADD ROLL CONSTANT
CLC
DAC MMDONE        /SFT DONE SWITCH
DISMIS
/DEC-TAPE WRITE SUBROUTINE, PDP-7
/FORMAT           JMS MMWRS
/                LAW B              /OR LAC (B), BLOCK NUMBER
/                JMP X              /ERROR RETURN
/                ZZ0000            /UNIT SELECTION
/                C1                /15-BIT CORE STARTING ADDRESS
/                C2                /15-BIT ENDING ADDRESS, INCLUSIVE
/                MULTI-PROGRAM RETURN

MMWRS,           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 MMRLF
MMWR1,          LAW MMWR3
                DAC MMDATA
                DZM MMDONE
                CLC
                DAC MMSUM         /START CHECKSUM
                XCT MMWA3         /SEARCH COMMAND
                ADD (2)          /CREATE WRITE COMMAND
                MMLC
                DISMIS
MMWR2,          LAW 600           /ERROR FLAG DURING WRITING
                JMP MMERX2
MMWR3,          EEM              /ENABLE EXTENDED MODE
                LAC I MMADDR      /OR CLA
                LEM              /DISABLE EXTEND MODE
                MMWR
                JMS MMRD6         /CALCULATE CHECKSUM
MMWR3A,         ISZ MMFILC       /SECTION COUNTER
                JMP MMWR6         /SFT UP FOR 2ND SECTION
                LAC MMW2          /CIA
MMWR3B,         DAC MMWR3+1
                DISMIS
MMWR4,          LAC MMSUM        /WRITE CHECKSUM
                CMA
                MMWR
                JMS MMRLL        /CHECK NEXT BLOCK NUMBER
                JMP MMWR1

```



```

MMWR6,      LAC MM2CN
             SNA
             JMP MMWR3A
             DAC MMWDC
             LAC (LAC I MMADDR)
             SAD MMWR3+1
             JMP MMWR3B-1
             JMP MMWR3B
    
```

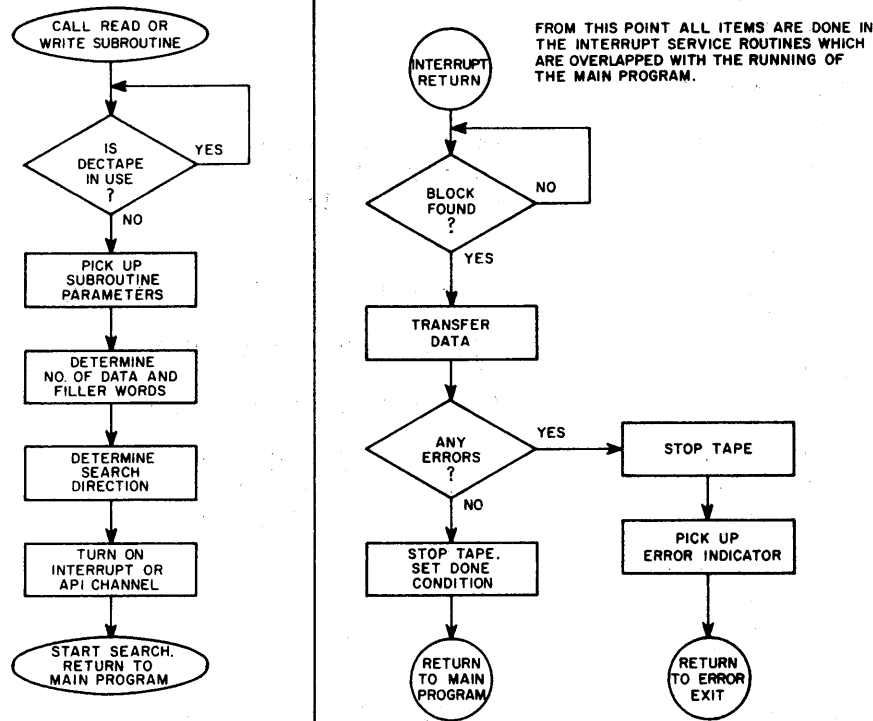
```

MMWA6=MMSAVE
MMWA5=MMRD6
MMWA3=MMRDS
MMSFK=MMPLC
MMWA2=MMCHK1+1
MMSBK=MMROLL
MMWA=MMWRS
MMSUM=MMITLK
MMWA7=MMYSUM
MMSA=MMERR
    
```

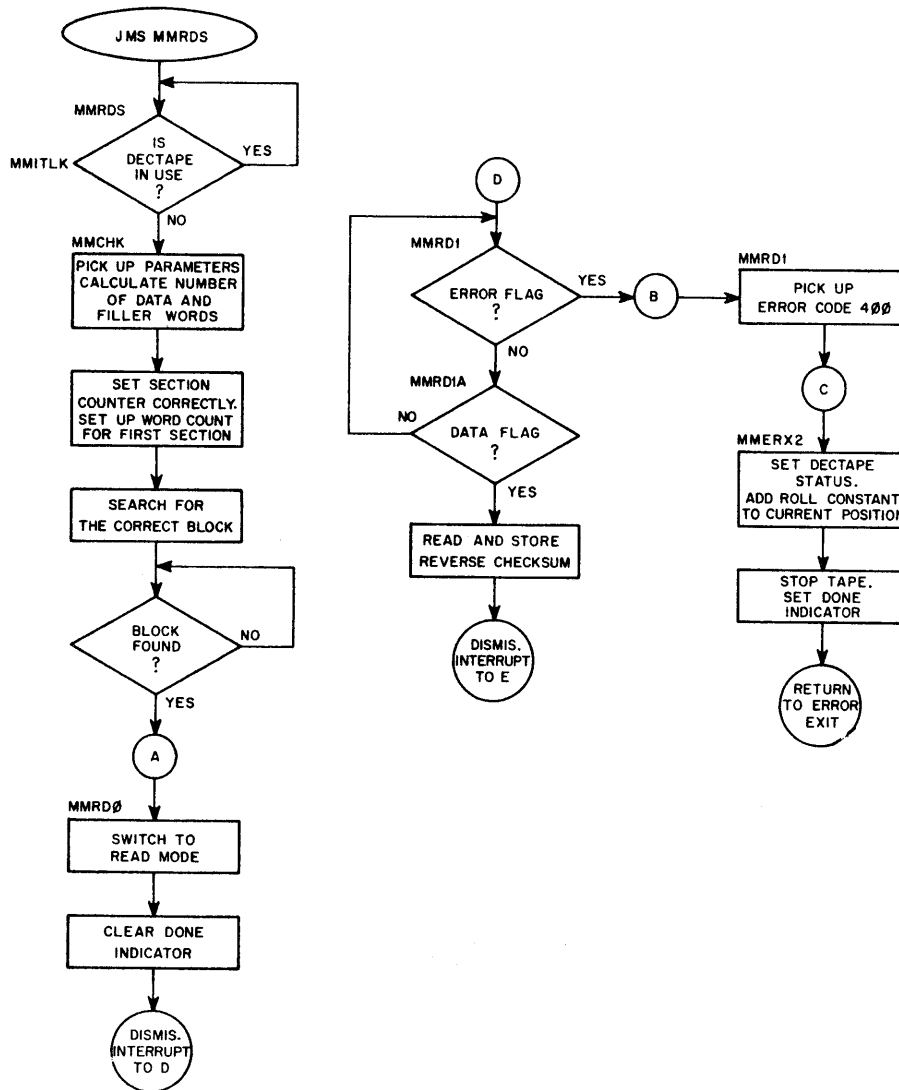
START

11. DIAGRAMS

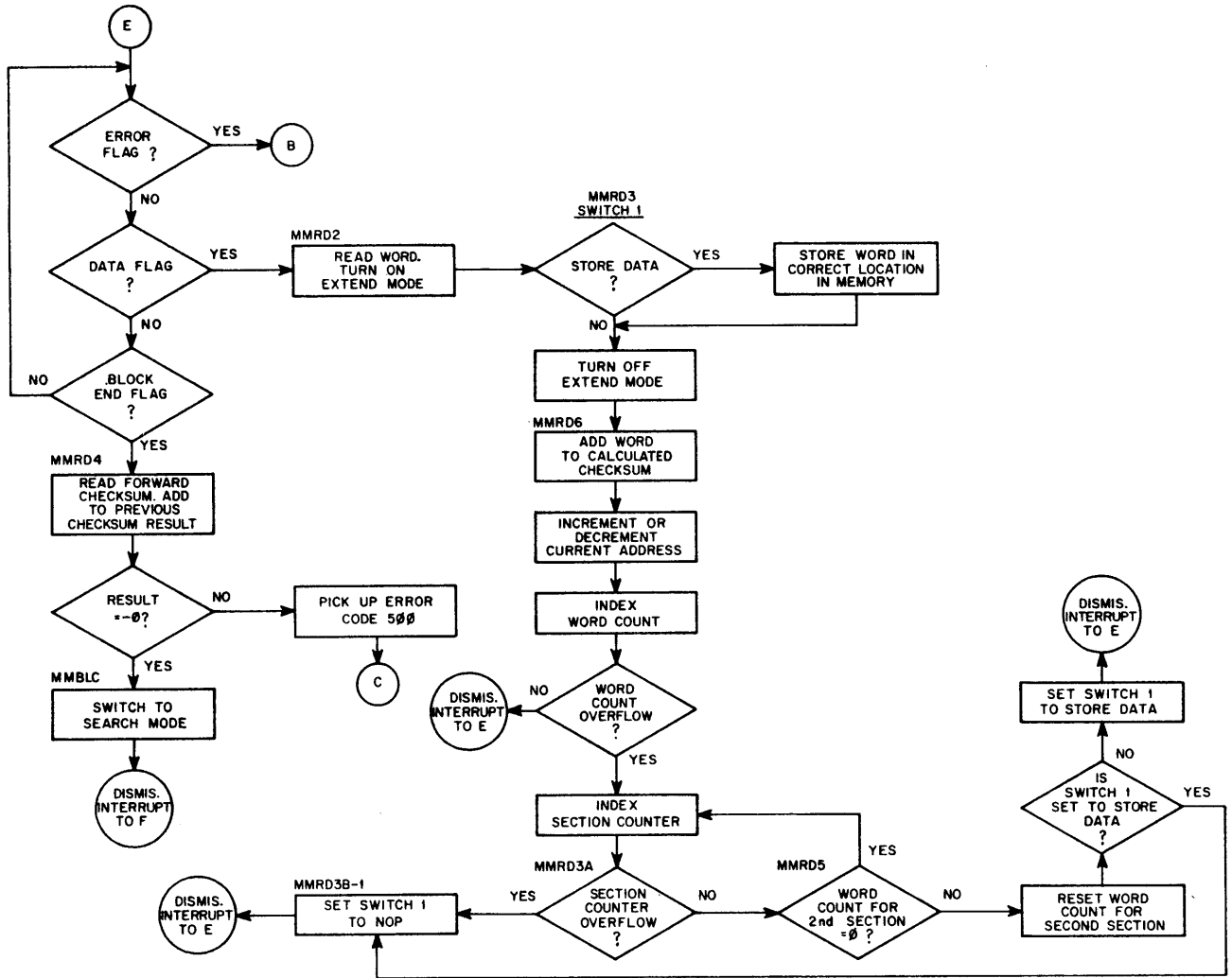
11.1 Flow Charts



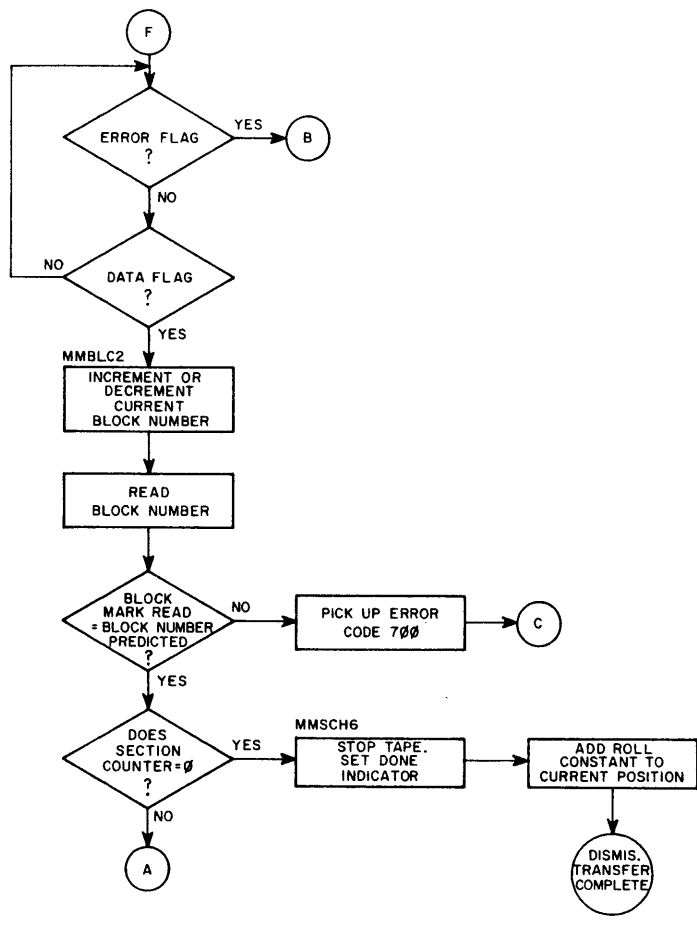
Gross DEctape Subroutine



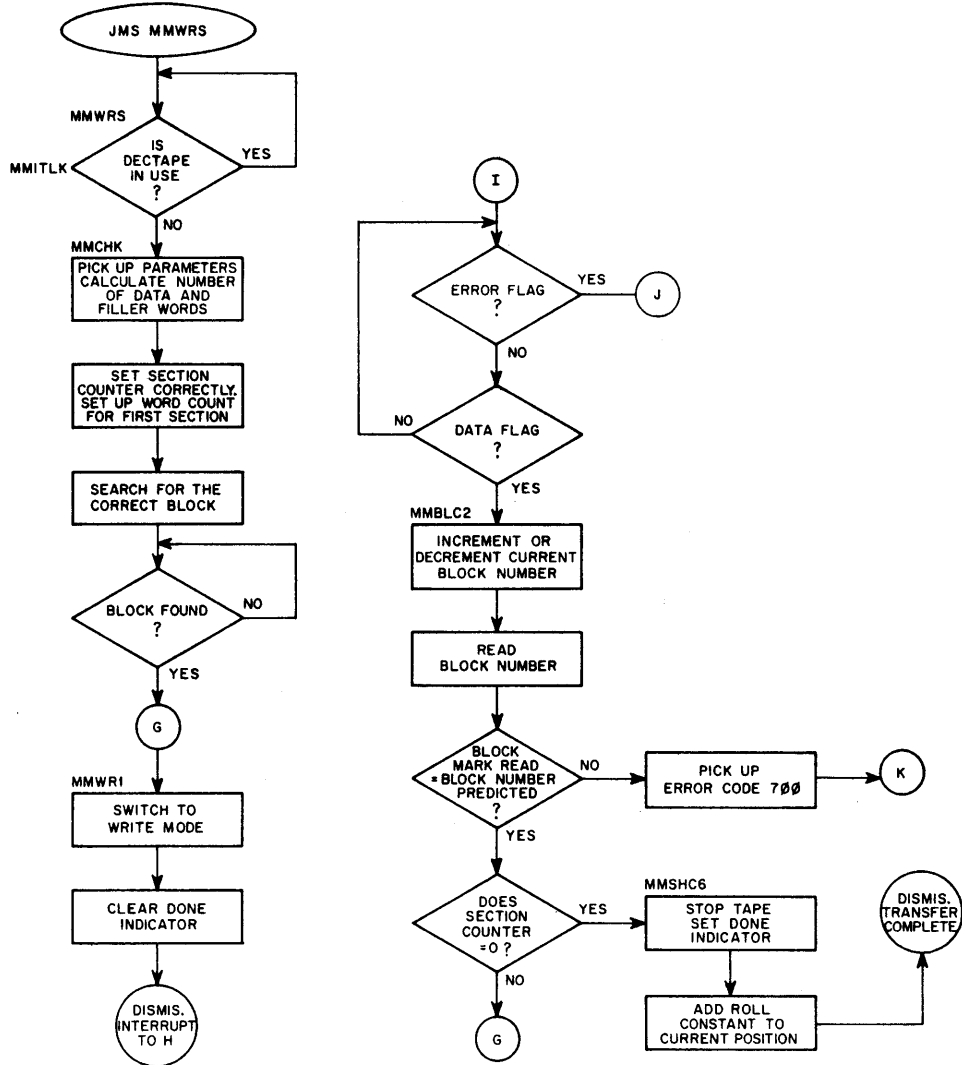
Read Routine



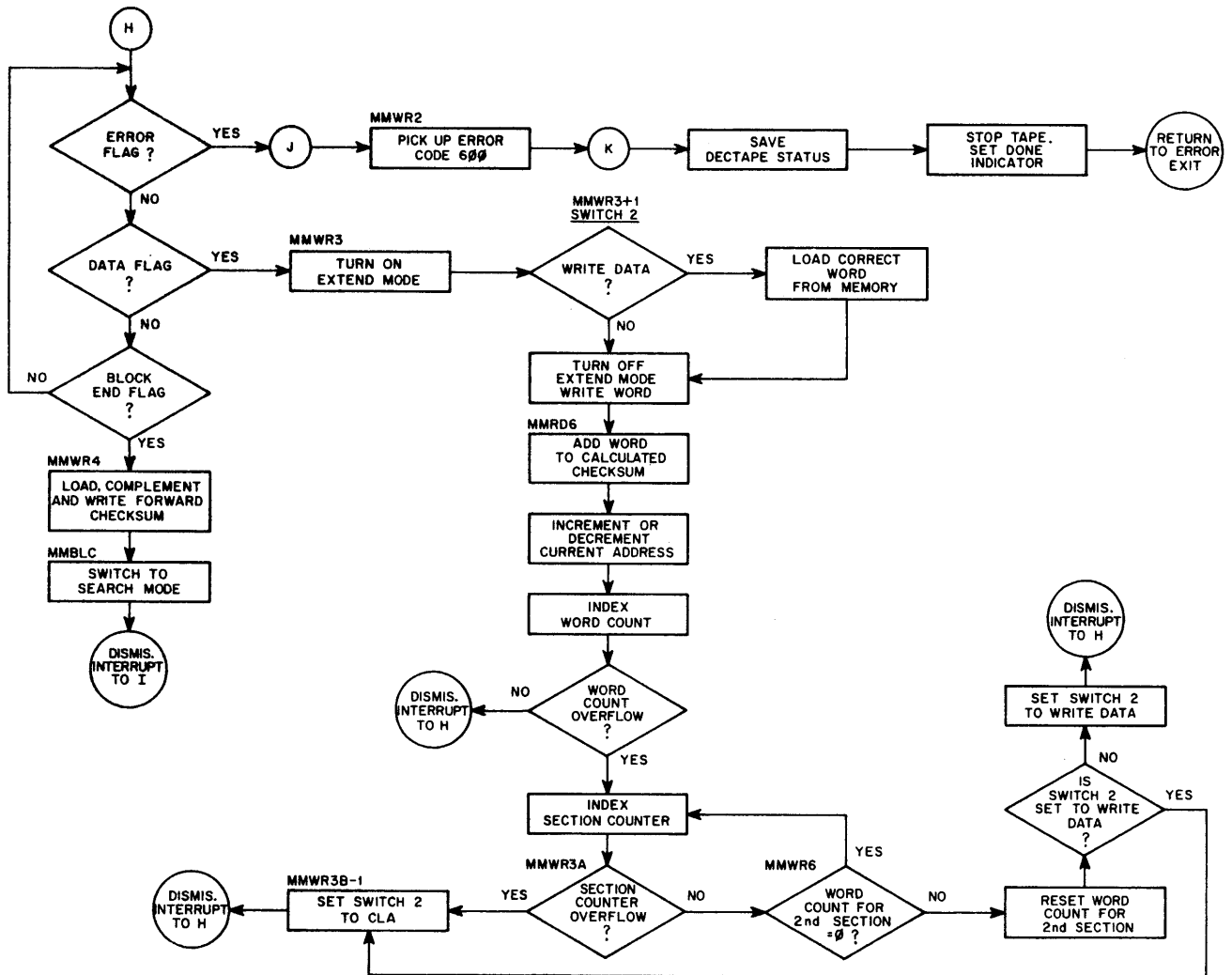
Read Routine (continued)



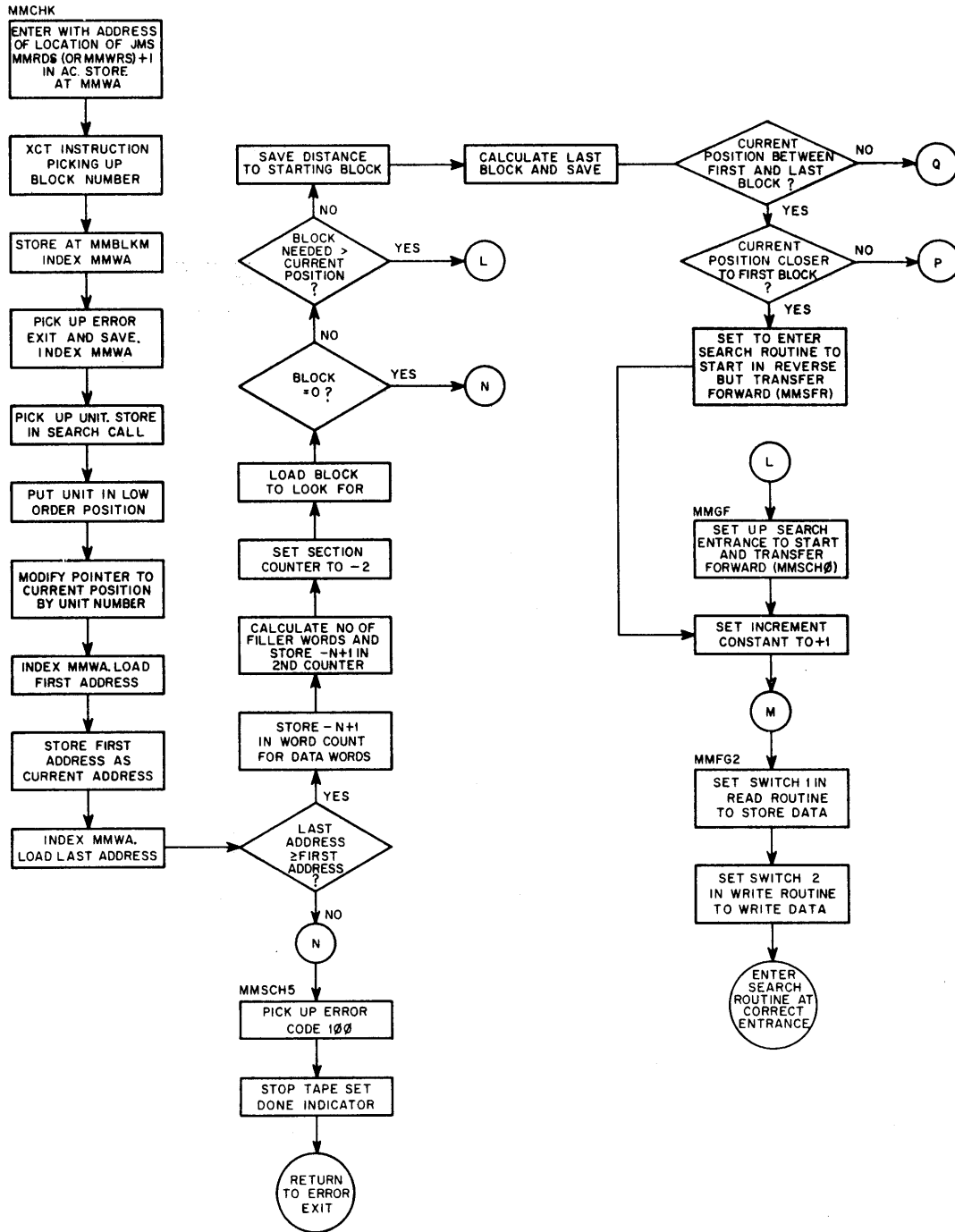
Read Routine (continued)



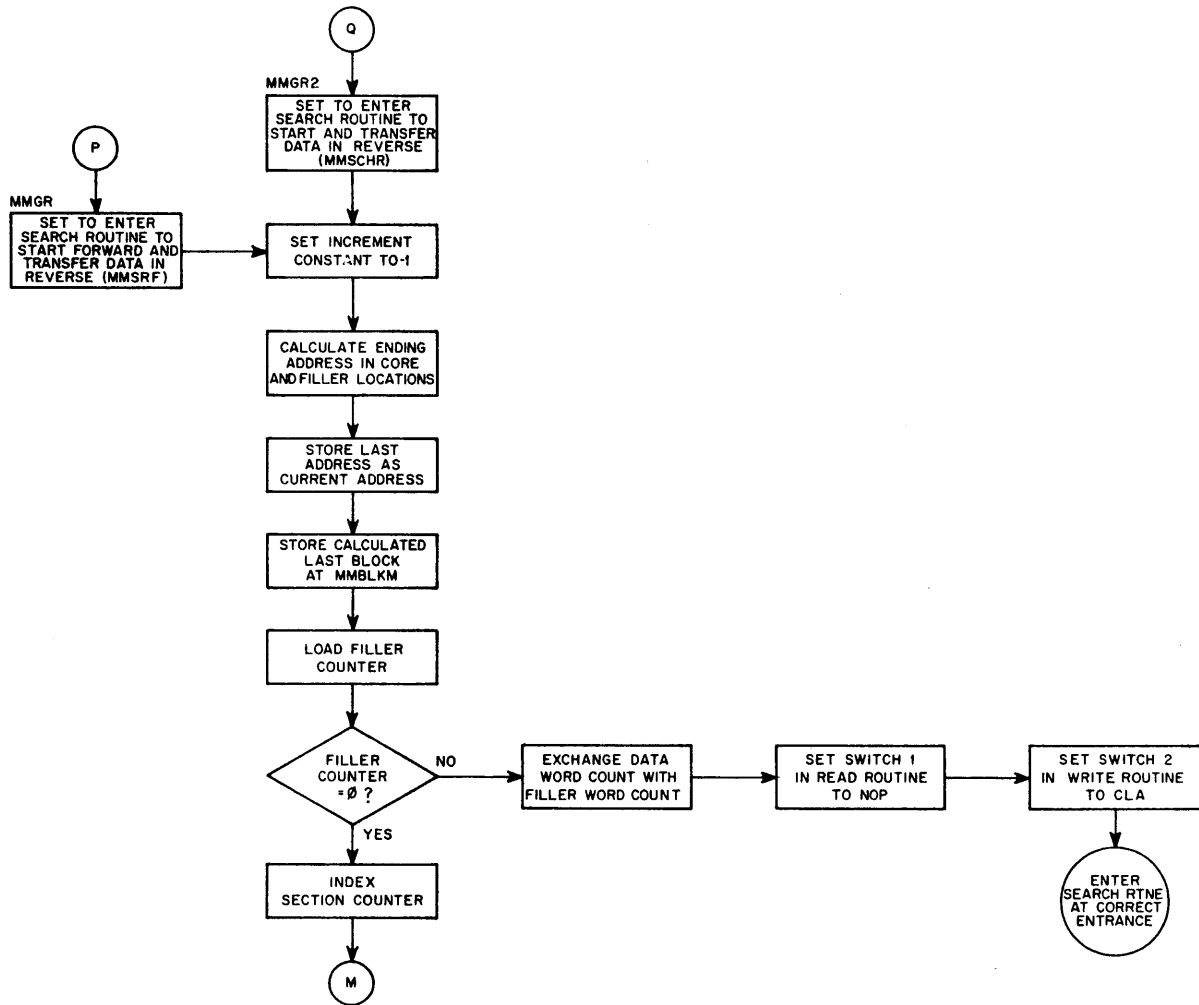
Write Routine



Write Routine (continued)

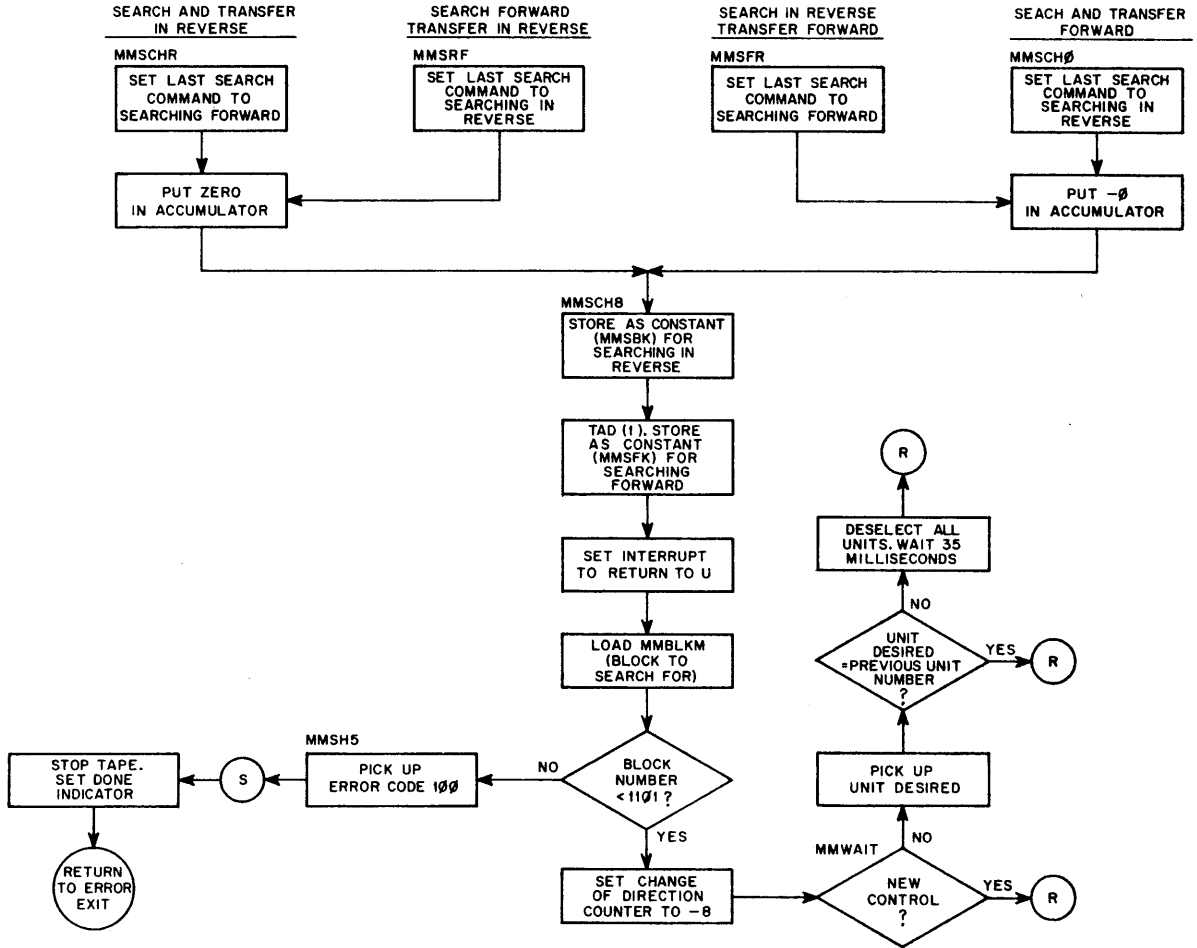


Common Routine to Pick Up Parameters and Initiate Searching

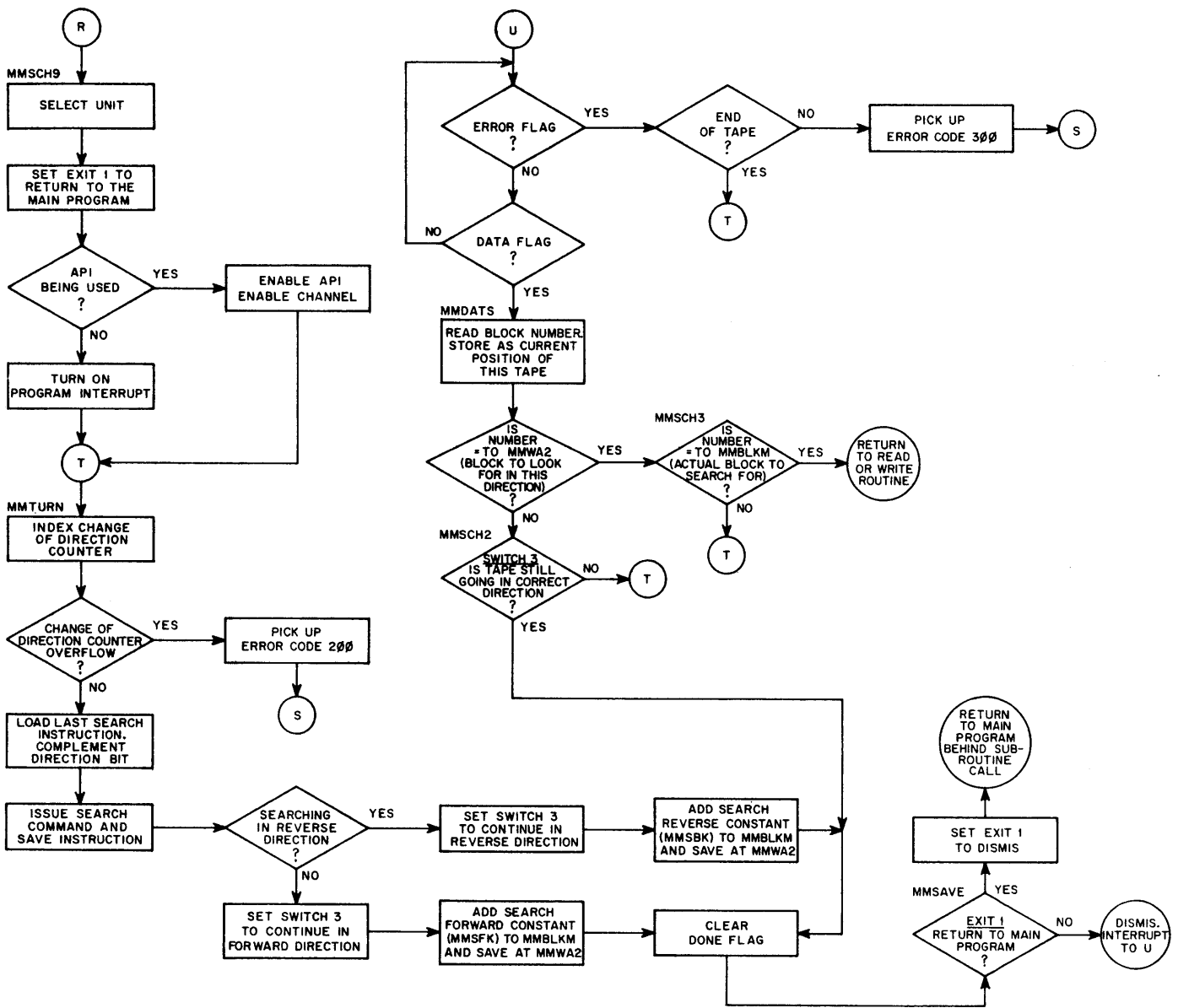


Common Routine to Pick Up Parameters  
and Initiate Searching (continued)





Search Routine



Search Routine (continued)