ITEM NUMBER 6309-0235 **(T. I. E.)**
27 pages

DATE July 31, 1963

AUTHOR Paul I. Bartholet

TITLE 1401 PROGRAMMERS GUIDE - ELIZABETH OFFICE

SOURCE IBM CORPORATION
570 North Broad Street
Elizabeth, New Jersey

6309 0235

'STRIBUTED BY
THE PROGRAM INFORMATION DEPARTMENT (TIE)
IBM CORP.
112 EAST POST ROAD
WHITE PLAINS, NY

TABLE OF CONTENTS

SECTION I

INTERNAL PROGRAMMING TECHNIQUES

A. Index Registers

1. Timing

Indexed instructions require three to four additional "I" cycles per address indexed and therefore use additional process time. For example the time to move ten characters with and without indexing is as follows:

Without indexing	322.0 us
One address indexed	356.5 us
Both addresses indexed	391.0 us

As can be seen, indexing increases the process time by more than 10% and 20% respectively. While indexing easily outperforms address modification in terms of both time and core, it may not provide the best solution when dealing with fields in a tape I/O area. In this case every instruction referring to the I/O area is indexed and the total process time is substantially increased. The use of a work area would reduce this increase in process time but would use more core storage. If core is available, a work area should be considered especially if iterative routines (e.g. programmed multiplication) are a basic part of the program or if a slight increase in the process time could cause a missed clutch point on the card reader or punch.

2. Testing an Incremented Address

When testing an incremented address, remember that the result of a comparison is in accordance with the 1401 collating sequence. Therefore, a test of the address "965" would indicate that it was higher than "U65" (1465).

3. Incrementing index registers

No circuitry is available in any positions of 1401 memory to allow unit position bit generation on add overflow conditions. Therefore, a straight add instruction will not properly increment an index register equal to or higher than the next 4K multiple over the initialized value. In such cases, the units position zone bits are properly generated only when the Modify Address Instruction is used.

4. Initializing index registers

When initializing an index register, use an "LCA" instruction rather than a "ZA". The ZA causes zoning over the units position and, thus, gives an effective address of a higher module of memory. Once initialized, incrementing with an "A" instruction is correct and will not cause incorrect zoning. A MCW would also be a satisfactory method of initializing an index register, but it is then necessary in most cases to set a work mark through the use of a "SW" instruction or a DCW.

B. Summary of Halt Techniques

Aside from stopping the operation, a programmed halt should provide two things. First, there must be an indication to the operator as to why the halt occurred, and second, after corrective action has been taken, there must be some means of getting to the restart instruction. The following are four common methods of using a programmed halt:

1. H
   Halt. The I address register provides the only means of identifying this halt. Also, the next sequential instruction must be part of the restart procedure.

2. H RESTRT
   Halt and branch to RESTRT. This is the same as #1 above except that when the start button is depressed, the program will branch to the restart procedure.

3. N 0999
   H
   B RESTRT
   Halt and branch to RESTRT. This method identifies the halt by having "999" in the A address register at the time of the halt. A branch to the restart procedure will be executed when the start button is depressed.

4. H 0999 0999
   B RESTRT
   Halt and branch to RESTRT. This is the same as #3 above except the "999" (or whatever code is wanted) will be in both the A and B address registers at the time of the halt.

All of the above four methods provide a means of identifying the halt and provide for restarting the program once corrective action has been taken. Methods #1 and #2 use the I address as a means of identifying the halt and this number would change if the program were changed. From this point of view methods #3 and #4 are preferable.

Under all four methods depressing the start key causes the program to resume running. In the past this has caused some problems due to operator error. This problem can be avoided if the following programmed halt were used:

```
HALT 1  H  RESTRT  0999
        B  HALTI
```

In this case depressing the start key will cause the program to branch back to the halt again. Once corrective action has been taken, the operator gets the address of the restart instruction from the A address register and enters this address using the Manual Address Switches.

C. Summary of Negative Zeros

The following five steps or signs of steps will each result in a negative zero.

1. Reset subtract a "+0".

2. Multiply two factors of opposite signs when one of the factors is a zero.

3. Move a "B" zone bit to the units position of a zero field.

4. Subtract using the A field only (S AAA). The sign of the least significant digit will remain. Thus, if the original field was negative, a negative zero will result. This is a common method used for clearing counters, and it should be noted that when a counter sitting at minus zero is edited and printed, the "CR" or "-" will print.

5. Reduce a minus figure by repetitive additions until the figure reaches zero. This will be a minus zero.

D. Arithmetic Overflow

The load program clear storage cards for the SPS and Autocoder Programs cause the arithmetic overflow latch to be set. If a program contains a branch on arithmetic overflow (B XXX Z), the latch should be turned off before entering the main line of the program. The following are two suggested methods of doing this.

1. Include a B XXX Z in the initialization of the program. This will turn the latch off.

2. Change the Autocoder or SPS bootstrap card to turn the latch off. Suggested changes are as follows:

```
Autocoder
,008015,022029,036041,047054,061068B047ZB047
1054041,072040,0010011040
SPS
,008015,022029,056063,036041,049049B049ZB049
1056055,0240671056
```

E. Console Entry

To enter data into a 1401 from the console:

1. Use the address dials to set up three positions of data on each entry.

2. Have a two step routine in your program to Halt and SBR XXX. This will allow the operator to press the "B" address register Key- Light.

If more than three positions are required, you can make several entries. Another method, if the range of numbers is below 15,999 on a system over 4K, is to enter the three digit address and decode it by a program.

This technique is advantageous for clearing registers, initializing invoice numbers or check numbers and resetting totals for reruns of portions of a job.

F. Other

1. Forms Control Instructions

Whenever possible, a forms control instruction should be given prior to the print instruction (skip or space after printing). If a forms control instruction is given after a print instruction (skip or space immediately), the whole system will be interlocked during the print cycle.

This is especially important when the system has print storage. If a print instruction were followed immediately by a CCJ (one space immediately), the whole effect of print storage would be wasted.

2. Editing a Data Field

An edited data field must be of at least two characters.

3. One Position Accumulators

Care should be taken in using a one position accumulator as it will not properly handle the bit overflow on a straight add or reset add.

4. Word Mark Locations

When re-assigning defined fields under a DA, be sure to investigate the effects of new word mark positioning throughout the program.

Example:

```
First       DA      1 X 100
Field 1             1,5
Field 2             6,10

            ORG     First
Second      DA      1 X 100
Field 3             1,6
Field 4             7,10
```

SECTION II

PACKAGE PROGRAMS

A. Autocoder

1. Equate Index Registers

   Always provide for an EQU of X1, X2 and X3 before using them as labels. Autocoder does not assume an address for them.

2. Blocked Records and Indexing

   When processing blocked records using Autocoder DA entries with implicit indexing, be sure to nullify indexing on the tape operations which refer to an indexed label of the DA. For example:

   ```
   RT  1, LABEL + XO
   ```

3. Eight Position Branches

   Autocoder does not presently check for valid instruction length with respect to the instruction OP code. Thus, a BCE or BWZ can assemble as a seven position instruction without an error indication. For example the following could result:

   ```
   B 555 072
   ```

   In this case, a branch to 555 would normally not take place. However, if the B. Address Character is the same as the last character in the branch instruction, then the branch will take place. In the above example if core position 072 contains a Z, then the program will branch to 555. Such occurrences, though rare, can cause debugging problems.

B. SPS

1. Origin Instruction

   An ORG instruction cannot use a Symbolic address in the "A" operand. Actual four position addresses only are valid.

If a symbolic address is used, the SPS program treats the numeric portion of each alpha letter as the actual address. If the numbers exceed core capacity the address will be decremented by the maximum core capacity until a valid address is achieved.

C. IOCS

1. Wrong Length Record Test

In order to have the ability to check for a long record on a tape read, the following procedure could be followed:

a. Establish a read in area followed by two group marks with word marks.

b. Before giving a read instruction remove the word mark from the first group mark.

c. Immediately after the read, store the B register and compare the address stored with the correct address if the proper length record had been read.

Note that the second group mark word mark prevents memory being destroyed if a long record is read. Also, if the read-in area is also to be used as a write area, the word mark under the first group mark must be replaced before the write is performed.

2. Relationship Between DTF Entries and Get and Put Formats

a. Input

INDEXREG Entry
N. WORKAREA Entry

IOCS assumes that all processing is being done in the input area defined in the DTF IOAREAS entry." GET TAPE 1" causes records to be read into the input area. "PUT INPUT, TAPE Z" causes records to be written from this area. The output tape DTF would have neither INDEXREG nor WORKAREA entries

No INDEXREG Entry
WORKAREA Entry

IOCS assumes that all processing of input tape records is being done in the work area defined in the DTF. "GET TAPE1" causes one record to be moved to the defined work area. Two forms of PUT can be used for writing this file. "PUT, TAPE2" is used if the same work area is defined in the input and output WORKAREA entries.

PUT WORK1, TAPE2 may be used if neither INDEXREG nor WORKAREA entries are included in the output tape DTF.

No INDEXREG Entry
No WORKAREA Entry

IOCS assumes that all processing will be done in the work areas named in the GET statement. "GET TAPE1, WORK'" reads a record or records and moves one record to the specified work area. The work area or work areas would be set up with DA statements. "PUT WORK1, TAPE2" would be used to write these records. The output tape DTF would have neither INDEXREG nor WORKAREA entries.

b. Output

INDEXREG Entry
No WORKAREA Entry

IOCS assumes that all processing will be done in the output area specified in the IOAREAS DTF entry of the output tape. "PUT, TAPE2" causes a record in the output area to be written. "GET TAPE1, TAPOUT" causes records to be read and moves one record to the output area. The DTF for the input tape should have neither INDEXREG nor WORKAREA entries.

No INDEXREG Entry
WORKAREA Entry

IOCS assumes that all processing will be done in the work area specified in the WORKAREA DTF entry of the output tape. "PUT, TAPE2" moves a record from the work area to the output area and writes the records when the output area has been filled. The input tapes WORKAREA DTF may specify the same work area as the output tape. "GET TAPE 1" would read records and move a record to the work area. The input tape would not have an INDEXREG DTF entry.

No INDEXREG Entry
No WORKAREA Entry

IOCS assumes all processing is being done in the areas named in the PUT statement. "PUT WORK1, TAPE2" causes a record to be moved from the work area specified to the output area and causes a write when that area has been filled. "GET TAPE1, WORK1" would cause a read when necessary and one record to be moved to the work area specified. The input tape would have neither INDEXREG nor WORKAREA DTF entries.

c. Miscellaneous

IOCS sets up tape read and write parameters depending upon the conditions specified in the input and output tape DTF entries. If these entries set up conflicting entries, the results will be in the error. For example, specifying an INDEXREG entry for both input and output tape files tells IOCS that all processing will be done in the input area and in the output area. In this case it would be necessary to move the records from the input area to the output area before giving the PUT. If this were not done , blank tape records would be written.

3. RDLIN Macro

If the RDLIN macro is used, it must be used before any housekeeping word marks are set in positions 1-80. The use of this macro will destroy any word marks in this area.

4. Unblocked, Variable Length Records

Unblocked, variable length records (form 3) do require record marks. The IOCS write-up states that they are optional, but IOCS generates a record mark to control moves from a work area to the output tape area. The generated record mark can overlap the group mark established to control tape writing and, thus, cause a core wrap around or wrong length record check.

D. RPG

1. Input Specs

On Input Specs the RPG manual states that, if a record entry is made for more than six (6) record codes, a double line entry can be made with the "resulting condition" and "control fields" as specified on the second line. The manual further states that the Sequence Code in columns 2-3 should be left blank.

This will work for a numeric sequence code but not on alpha sequence code. The alpha sequence code must be repeated on both lines.

2. In the Data Specs, a blank field cannot be specified as being numeric without giving incorrect results. This is due to the fact that the numeric designation causes the field to be re-set added to itself thus generating a sign.

E. FARGO

1. Eject

An "E" in column 55 of the Phase 2 Control Sheets will cause an eject after the total line has been printed provided that columns 52-54 do not contain a space or skip instruction. If these columns do contain a space or skip instruction, this instruction is carried out, and the eject is bypassed.

2. Two or More Lines Within a Total Class

When printing more than one line within a total class, use columns 3-5 of the Phase 2 Control Sheets for any extra spaces to be taken between lines. If you were to use columns 52-54 rather than 3-5, the SP in 52-53 would signal the end of the instructions for that total class and thus give incorrect results.

3. "All" Cards

Columns 2-6 of the Phase 3 Control Sheets MUST be left blank if there is only one type of detail card. The "ALL" designation in columns 3-6 will give incorrect results when used in this case.

F. Auto-Test

1. "Replace" Patch

To delete a tape, RAMAC or typewriter instruction using Auto-Test, use the "Replace" Patch feature rather than the "Delete" patch. The "Delete" patch merely replaces the move Op code with a NOP and the %Ux, %Fx, or %Tx address will cause the machine to hang up with an invalid Op code indication. A "Replace" patch of NØØØØØØØ will obliterate the %xx address and remove the cause of the hang-up.

2. Execute

When Auto-Testing a program which uses an Execute instruction to overlay a portion of the program, any data cards required by the portion to be overlayed must be placed behind the End-Start card for the last portion of the program rather than just behind the Execute card as would be the case in normal testing. This is necessary because Auto-Test first writes all cards in the program to tape and then loads into core from tape. When it reaches the Execute instruction and this portion contains a Read instruction, the proper data card must be in the reader hopper.

3. Snapshot of Core

You may call for a snapshot of core at any time during your program with the following two exceptions:

1) An unconditional branch instruction.
2) Any instruction which you intend to modify during your program.

Auto-Test modifies your instruction making it an unconditional branch to the snapshot routine. It will not restore your program to its original form if you modify the instruction elsewhere in the program.

4. Tape Labels

Any output tape containing a header label that is to be read before data is written must be defined as a tape to be read, not written, in the program control card. Auto-Test is concerned with the first operation to be performed which, in this case, is reading the header label. After the header is read, Auto-Test will shift control to your program, and the tape will be written under control of your program.

If the tape were defined as a tape to be written, Auto-Test would write a series of sentinel records thus wiping out the header label.

5. Sentinel Records

Sentinel records are written by Auto-Test on any output tape which is defined on the tape control card as a "Write" tape and which is not completely written during testing. A tape is not completely written if the tape mark routine is not reached. These sentinel records allow the programmer to tell those records written by his program from those previously on the tape.

6. Save and Pre-create Tapes

Any tape designated as a "save and pre-create" tape is under control of the tape control card. It will be rewound and unloaded during the Auto-Test run.

7. End of Job Halt

The End of Job Halt should be located in a standard location such as 333, for all programs. If this is done, the first control card will never have to be changed. It would have to be changed if the program was reassembled and the address changed.

8. Disk File Generation

Auto-Test RAMAC file generation may be used to place actual data in a 1405 on either a sector or a tract basis. When using auto-test for this purpose, the following cards must be included:

1). Auto-Test control card
2). RFGCTL
3). RAMAC File Data
4). Bootstrap cards
5). Object program. This should be a one or two card program that does nothing except come to an end-of-job halt. Auto-Test needs this program in order to function properly.

When generating full track data, the sectors should run continuously on the record data cards. The 1000 positions will appear on fifteen cards as follows:

1). Cards 1-14 - 70 positions each in card columns 11-80
2). Card 15 - 20 positions in card columns 11-30.

G. Sort II

1. User Programming Area

Errors presently exist in the formulas for calculating user programmer areas in Phase II and Multi-phase of Sort 2. The correct formulae are:

Phase II

3500 + (M+1) (BL+2) = Area

If area is less than 6300 the user must restrict himself to starting at 6300. Sort 2 uses 001 through 6299.

Multi-Phase

3558 + (4) (BL) = Area

6500 is the lowest core position available

M - Merge Order BL = Block Length

SECTION III

RANDOM ACCESS

A. 1405 Testing

1. Dumping File Areas onto Tape

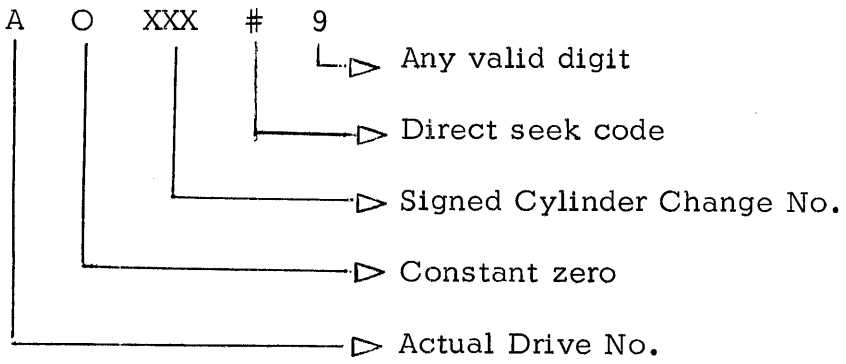
When testing 1405 RAMAC programs, it is usually necessary to set up the areas being used in RAMAC. When the test shots are short, with elapsed time in between them, time can be saved by dumping the file areas being used onto tape using the RAMAC to Tape Utility Programs. Dumping this tape back to the file provides a fast means of re-establishing the RAMAC areas for the succeeding test shots . This is especially useful as the detail of testing increases.

B. 1311 Direct Seek

1. Direct Seek Hints

When using the Di rect Seek feature a disk control field is necessary for the seek operation. The format of this field is different from that used in normal seeking, reading and writing. Therefore, if the disk control field for seeking is located in the normal core positions immediately before the I/O area, continual replacement is necessary. Because of this it may be advisable to locate the direct seek disk control field elsewhere in core while the read/write disk control field (alternate code, core sector address and sector count) occupies the normal positions.

In this case this direct seek disk control field takes the following format:



The three digit sector indication may be left out.

Note that the actual drive number must be used in the direct seek control field rather than the asterisk alternate code. The alternate code will not define the arm (drive) because the control field does not specify a core sector address.

Note also that an additional constant zero must be included behind the drive code.

The three digit signed cylinder change number is expressed as twice the actual cylinder change to be made; i.e., to change from one cylinder to the next adjacent cylinder this field would contain .002. A plus sign will direct the move into the file (to a higher numbered cylinder) while a minus sign will move the arm out of the file (to a lower numbered cylinder).

Once the direct seek control field is set up in this manner, it is not automatically altered by disk operations. Therefore, if the cylinder change code were initially set to 002 no modification would be necessary to process sequentially through an entire disk pack. When processing is not sequential, this three digit signed cylinder change code must be calculated.

2. Direct Seek Routine

The linkages necessary to enter the routine are:

1). Move the address of the record to be sought to the routine (labeled NEW)

2). Branch to the routine (labeled DIRECT)

At the conclusion of the routine the signed number is located at a location labeled NEW. The routine saves the address of the sector being sought now for use on the next direct seek calculation (starting address must be initialized) and automatically returns to the next step of the users program. Note that this routine is geared for a one pack file.

Program _____

Programmed by _____

Date _____

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM 1401, 1410 AND 1440 DATA PROCESSING SYSTEMS

AUTOCODER CODING SHEET

Identification _____ (76 80)

Page No. 1 of 2

Form X24-13
Printed in U

Line	Label	Operation	OPERAND
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60 65 70
01	Direct Seek Routine		
02			Result of Direct Seek Routine
03			
04			
05		MCW	CHK-2,NEW#3 MOVING 3 HI ORDER DIGITS
06		B	DIRECT
07			
08			
09	Direct	Seek Routine	
10			
11	DIRECT	SBR	RETURN,R SAVING RETURN ADDRESS
12		MCW	NUMRLD#3,RLD TEST
13		BCE	MAKEYE,NEW,1 ERR
14		BCE	MAKEYE,NEW,3 RDR
15		BCE	MAKEYE,NEW,5 NUMRLS
16		BCE	MAKEYE,NEW,7
17		BCE	MAKEYE,NEW,9
18	MOVE	MCW	NEW,NUMRLD SAVING THIS ADDRESS
19		S	RLD,NEW SIGNED DIFFERENCE
20	RETURN	B	0
21	MAKEYE	S	0,0,NEW
22		B	MOVE
23			
24			
25		MCW	NEW,ADDR SETTING UP ADDRESS IN XXXX?

C. 1311 - Other

1. Avoiding Unnecessary Seeks

When processing on the 1311 it is often advantageous to determine whether or not the next record to be handled is located on the same cylinder as the one on which the arm is currently located. Two methods of making this determination have been published and are as follows:

a. In each case attempt to read the next record without seeking it. If the following address compare is unequal, then a seek is required. This method is very slow.

b. Divide the first three digits of the actual address by 2. The result is the cylinder number which can then be used to determine whether or not a given record is on the same cylinder as the previous record. This division can be accomplished by multiplying by .5 or by repeti tive addition. (See 1440 System Installation Topics, Page 47). The following are examples:

Cyl.	Address Range From	To	Sample Address Within Range	Result of Division
00	00000	00199	00050	000 20
01	00200	00399	00375	001 87
02	00400	00599	00401	002 01
03	00600	00799	00700	003 50
04	00800	00999	00850	004 25

2. Disk Control Field During Reading and Writing

In normal 1311 reading and writing the Disk Control Field is automatically altered. The core sector address is incremented with each successful transfer of data provided that the sector count is not 000. The sector count is decremented with each successful address compare.

00001 1st Sector 00002 2nd. Sector 00003 3rd Sector

Disk Record

1 2 3 4 5 6 7

Core Sector Address	Sector Count	Function
1. 00001	003	Start of READ/WRITE
2. 00001	002	1st Add'r Compare
3. 00002	002	End of First Transfer
4. 00002	001	2nd Add'r Compare
5. 00003	001	End of Second Transfer
6. 00003	000	3rd Add'r Compare
7. 00003	000	Completion of READ/WRITE

Because the address compare precedes the transfer of data, the sector count will reach 000 prior to the last transfer of data when reading or writing multiple sectors. When reading or writing a single sector this will occur prior to the read of that sector.

Note that this automatic subtraction does not generate a sign over the Sector Count portion of the Disk Control Field.

As stated above the sector count of 000 prevents the incrementing of the core sector address. Therefore at the termination of a read or write the sector count will stand at 000 while the core sector address will be the address of the last sector read or written. So, another read or write issued against this disk control field will re-read/write this last sector. To avoid this the core sector address must be incremented by program.

3. Starting Disk Address

Prior to initiating a read or a write, the starting address should be saved (moved to some area other than the Disk Control Field) because the read or write destroys this address in the Disk Control Field. This address will be needed in the following instances:

a. In the case of reading or writing, it is necessary for re-entry in the event of an error.

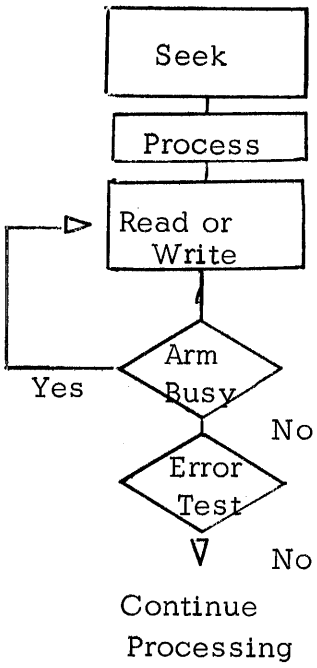
b. In the case of writing, it is needed for the subsequent write-check instruction.

c. In the case of a typical updating run (read, process, write, write-check) it is needed for the write and write-check instructions.

4. "Arm Busy" Latch

A read or write disk instruction given after a seek but prior to the arm coming to rest will be treated as a NOP. Program execution will continue without the new record having entered core. There is no automatic interlock as in former RAMAC Systems (305,655,1405).

However, the seek instruction sets on internal "latch" (arm busy). This "latch" remains ON until the successful execution of a read or write. Therefore, whenever a read or write has been <u>preceded by a seek</u> it should be followed by an arm busy interrogation. Note that the "d" modifier for the arm busy instruction is an apostrophe (0-6-8). This character does not print.



5. Scan Disk

Scanning of the 1311 disk file can stop for any one of the following reasons:

a. Section count of 000.

b. Satisfying the Search Argument.

c. Overflowing the cylinder

The following table indicates the status of the Disk Control field when the scan operation has stopped:

Reason	Core Sector Address	Sector Count
1. Sector Count 000	Last Sector Scanned	000
2. Satisfying Search Condition *	Sector containing Search Condition	Number of Sectors Remaining to be scanned

	Reason	Core Sector Address	Sector Count
3.	End of Cylinder ** Sector Count not 000	First sector of next cylinder	Number of sectors remaining to be scanned.

* A second scan disk instruction issued after a hit will re-scan the sector on which the hit occurred. This can be avoided by incrementing the core sector address.

** This condition results in an address compare error since the core sector address is not on the cylinder where the arm is located.

SECTION IV

MAGNETIC TAPE

A. Reading and Writing Tape

The 1401 Move Tape operation consists of an op code, L or M; an A Operand of % U x, a B-Operand of the address of data field; and a d-Modifier to specify reading or writing. Move tape is only another form of data transmission even though it refers to external units and not core to core transmission.

The SPS mnemonic for the move tape op code is MCW, the same as for a regular move. If a programmer is inattentive he may write it as MCM. This mnemonic is a move op code also but refers to a move of data from left to right up to and including a recordmark or groupmark/wordmark in the data record. Written in a tape instruction, in SPS, the op code P will be generated and the correct A and B operands and the d-Modifier. This instruction, P/% U 1/930/W will be executed on the machine, but control of the reading or writing will be determined by the first recordmark/wordmark encountered as transmission proceeds from left to right.

As long as the data record does not contain any record mark characters, the equivalent of a correct read or write tape will take place. If a recordmark does appear it will write out only up to and including the recordmark character.

Since the SPS assembly will not catch this type of coding error, the programmer must be alert to it on his initial debugging if he runs into problems on the size and contents of his tape records. This will be true also on a production program that has apparently been running fine but suddenly develops tape trouble.

B. Tape Marks

When writing Tape Marks, it is always advisable to either check to see that a valid tape mark has been written or to write a series of tape marks. A suggested routine is as follows:

```
WTM     WTM         3
        BSP         3
        RT          3, Input
        BEF         EOJ
        B           WTM
```