SCRIPT TO RUNOFF CONVERSION PROGRAM

Ly Thien Hoang

Course XV

May 1983

Copy 1
SCRIPT TO RUNOFF CONVERSION PROGRAM

by

LY THIEN HOANG

B.A., Harvard College (1978)

Submitted to the Sloan School of Management in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN MANAGEMENT

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1983

© Ly T. Hoang 1983

The author hereby grants to M. I. T. permission to reproduce and to distribute copies of this thesis document in whole or in part.

Signature of Author: [signature]

Sloan School of Management, May 1983

Certified by: [signature]

Prof. Stuart E. Madnick, Thesis Supervisor

Accepted by: [signature]

Dr. Jeffrey A. Barks, Director of Master's Programs
SCRIPT TO RUNOFF CONVERSION PROGRAM

CONTENTS

1. INTRODUCTION  page 4
2. DESIGN  page 5
3. FLOWCHART  page 6
4. PROGRAM EXECUTION  page 7
5. CONCLUSION  page 8
6. PROGRAM LISTING  page 9
7. SAMPLE SESSION  page 74
1. INTRODUCTION

SCRIPT and RUNOFF are text formatters available on the IBM and PRIME computers, respectively. Sloan students and staff have access to both computer systems. However, the PRIME computer belongs exclusively to the Sloan School, whereas the IBM computer does not, therefore, it is more economical for Sloan computer users to do as much computer processing as possible on the PRIME.

The purpose of this program is to convert existing SCRIPT files into RUNOFF files so that they can be processed on the PRIME computer.

This program also enables PRIME users who know SCRIPT but not RUNOFF to do word processing on the PRIME (currently, RUNOFF is the only text formatter available on this system).
2. DESIGN

The conversion process is straightforward: the SCRIPT input file is read line by line. If the input line is a SCRIPT command, it is converted to one or more RUNOFF commands to achieve equivalent result. Any SCRIPT option is also taken care of during this conversion process and following SCRIPT input lines are modified if necessary.
3. FLOWCHART

START

READ

WRITE OUTPUT LINE(S)

PROCESS COMMAND

END OF FILE?

INPUT A SCRIPT COMMAND?

END
4. PROGRAM EXECUTION

To convert a SCRIPT file into a RUNOFF file, the user just type STR (for Script To Runoff) and the name of the SCRIPT file (filename).

Example: C>STR TEST

A copy of this file (TEST) is made under the name T$.SCRIPT. This allows hard coding of the input file name in the program. A temporary output file (T$.OUTSCRIPT) is created. After processing, the temporary input file is deleted and the temporary output file is renamed to filename.RUNOFF. The copying and renaming are done through the STR PRIMOS CPL file (a copy of this STR.CPL file is printed out during the sample session enclosed). The user can then execute RUNOFF to process this newly created text file.

Example: C>MRUNOFF TEST.RUNOFF
5. CONCLUSION

Since SCRIPT is more sophisticated and supports many more features than RUNOFF, many SCRIPT commands cannot be converted fully or at all to RUNOFF. However, the current conversion program is adequate to convert intermediate-level SCRIPT file.

Future work on this program can be implemented if other SCRIPT features are available in RUNOFF. Otherwise, it is probably not beneficial to put more effort in this program just to accommodate some additional SCRIPT commands.
6. PROGRAM LISTING

A listing of the program begins on next page.
SCRIPT TO RUNOFF CONVERSION PROGRAM

STR:

DCL IN
DCL OUT
DCL EDF
DCL NO_READ
DCL NO_WRITE
DCL 1 INREC,
   2 INLEN
   2 INCARD
DCL 1 OUTREC,
   2 OUTLEN
   2 OUTCARD
DCL COMMAND
DCL PARM
DCL PART1
DCL PART2
DCL PART3
DCL TAB_CHAR
DCL INCOUNT
DCL OUTCOUNT
DCL I
DCL J
DCL K
DCL TEMP_NUM
DCL TEMP_CARD
DCL LENGTH
DCL SUBSTR
DCL TRANSLATE
DCL VERIFY

PROC OPTIONS(MAIN):

FILE INPUT;
FILE OUTPUT:

BIT STATIC INIT('O'B);
BIT STATIC INIT('O'B);
BIT STATIC INIT('O'B);

FIXED BIN.
CHAR(78):

CHAR(2):
CHAR(78) VAR:
CHAR(78) VAR:
CHAR(78) VAR:
CHAR(78) VAR:
CHAR STATIC INIT('');

FIXED BIN STATIC INIT(O):
FIXED BIN STATIC INIT(O):

FIXED BIN;
FIXED BIN;
FIXED BIN;

PIC'Z9';;
CHAR(78) VAR:

BUILTIN.
BUILTIN.
BUILTIN.
BUILTIN;
SCRIPT TO RUNOFF CONVERSION PROGRAM

OPEN FILE(IN) TITLE('T$.SCRIPT') STREAM INPUT;
OPEN FILE(OUT) TITLE('T$.OUTSCRIPT') STREAM OUTPUT;

ON ENDFILE(IN) BEGIN;
  EOF = '1'B;
  PUT SKIP LIST('INCOUNT IS ',INCOUNT);
  PUT SKIP LIST('OUTCOUNT IS ',OUTCOUNT);
  PUT SKIP LIST('END OF JOB');
  GOTO EOJ;
END;

PUT SKIP LIST('JOB STARTED');

DO WHILE(^EOF);
  IF NO_READ THEN NO_READ = '0'B;
  ELSE CALL READ;
  IF SUBSTR(INCARD,1,1)='.' THEN CALL PROCESS;
  ELSE OUTCARD = INCARD;
  IF NO_WRITE THEN NO_WRITE = '0'B;
  ELSE CALL WRITE;
END;

EOJ:
CLOSE FILE(IN);
CLOSE FILE(OUT);
RETURN;
SCRIPT TO RUNOFF CONVERSION PROGRAM

PROCESS:

PROC;

DCL BM PIC'ZZ9' STATIC INIT(5);
DCL IN PIC'ZZ9' STATIC INIT(0);
DCL IL PIC'ZZ9' STATIC INIT(0);
DCL IR PIC'ZZ9' STATIC INIT(0);
DCL LL PIC'ZZ9' STATIC INIT(85);
DCL PA PIC'ZZ9' STATIC INIT(0);
DCL PL PIC'ZZ9' STATIC INIT(66);
DCL UN PIC'ZZ9' STATIC INIT(0);
DCL TM PIC'ZZ9' STATIC INIT(5);
DCL MOD PIC'ZZ9';
DCL TEMP PIC'SZ99';
DCL SIGN_POS FIXED BIN;
DCL SIGN CHAR;

COMMAND=SUBSTR(INCARD,2,2);
COMMAND=TRANSLATE(COMMAND,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'abcdefghijklmnopqrstuvwxyz');
The .BM (Bottom Margin) control word specifies the amount of space to be reserved at the bottom of the output pages, overriding the initial value established for the device. 

$v$ specifies the amount of space to be reserved at the bottom of output pages. If $+v$ or $-v$ is specified, the current value of the bottom margin is incremented or decremented. If no value is specified for $v$, the initial setting is restored.
/* .BM processing */

IF COMMAND='BM' THEN DO;
    IF INLEN=3 THEN DO;
        OUTCARD=INCARD;
        BM = 5;
        END;
    ELSE DO;
        SIGN_POS=VERIFY(SUBSTR(INCARD,4,INLEN-3),'0123456789');
        IF SIGN_POS = 0 THEN DO;
            OUTCARD=INCARD;
            IF SUBSTR(INCARD,4,INLEN-3) &= ' ' THEN DO;
                BM = SUBSTR(INCARD,4,INLEN-3);
                /* Runoff specification */
                IF BM = 0 THEN BM = 5;
                END;
            ELSE BM = 5;
            END;
        ELSE DO;
            MOD = SUBSTR(INCARD,SIGN_POS+4,INLEN-SIGN_POS-3);
            IF SUBSTR(INCARD,SIGN_POS+3,1)= '+' THEN BM = BM + MOD;
            ELSE IF SUBSTR(INCARD,SIGN_POS+3,1)= '-' THEN DO;
                TEMP = BM - MOD;
                IF TEMP > 0 THEN BM = 1;
                ELSE BM = TEMP;
                END;
            OUTCARD = '.BM' || BM;
            END;
        END;
    END;
RETURN;
END;
The .BR (Break) control word prevents the concatenation of the following text line with preceding text.
.CD \n
The .CD (Column Definition) control word defines how many columns of output are to be formatted on each page. \n\n\n
n is the number of columns of output to be formatted onto each subsequent output page. It may be any number from 1 to 9.

ELSE IF COMMAND='CD' THEN DO;
   DO J=4 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' ');
   END;
   DO K=J TO INLEN WHILE(SUBSTR(INCARD,K,1)^=' ');
   END;
   OUTCARD = '.C' || SUBSTR(INCARD,J,K-J);
   RETURN;
END;
The .CE (Center) control word centers text lines between the current left and right margins.

n specifies the number of input lines to be centered. If omitted, 1 is assumed. If .CE n is specified when .CE ON is in effect, centering is turned off when n lines have been centered, or when .CE OFF is encountered.

ON specifies that subsequent text lines are to be centered.

OFF terminates centering mode if it was ON, or if n has been specified and has not been exhausted.

line is a line of text to be centered. The line is considered to start with the first nonblank character after the .CE control word.

*/

ELSE IF COMMAND='CE' THEN DO;
         CALL CE_START;
         RETURN;
         END:
The .CM (Comment) control word allows comments to be stored in the Script file for future reference.

```plaintext
/*-----------------------------------------------*/

.CM

ELSE IF COMMAND='CM' THEN DO;
  OUTCARD='.*' || SUBSTR(INCARD,4,INLEN-3);
  RETURN;
END;
```
The .CO (Concatenate Mode) cancels or restores concatenation of input lines and truncation at the current column length.
ON restores concatenation of input lines. ON is the initial setting, as well as the default value.
OFF cancels concatenation of input lines. If justification is still in effect, .CO OFF results in each line being padded with blanks to column length.
The .DS (Double Space mode) causes subsequent output lines to be doublespaced.
The .FO (Format mode) controls concatenation and justification of input lines.
ON restores default formatting, including both justification and concatenation of lines. If the .FO control word is used with no operands, ON is assumed.
OFF cancels concatenation of input lines and justification of output lines.
LEFT specifies that the input lines are to be concatenated but not justified.

ELSE IF COMMAND='FO' THEN DO;
    INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                       'abcdefghijklmnopqrstuvwxyz');
    IF INLEN = 3 THEN DO;
        OUTCARD = '.F';
        CALL WRITE;
        OUTCARD = '.AD';
        END;
/* Processing continued on next page */
SCRIPT TO RUNOFF CONVERSION PROGRAM

/* .FO processing continued */
ELSE DO:
   DO I=4 TO INLEN WHILE(SUBSTR(INCARD,I,1)=' ');
      END;
   DO J=I TO INLEN WHILE(SUBSTR(INCARD,J,1)^=' ');
      END;
   PARM = SUBSTR(INCARD,I,J-I);
   IF PARM='ON' THEN DO;
      OUTCARD = '.F';
      CALL WRITE;
      OUTCARD = '.AD';
      END;
   ELSE IF PARM='OFF' THEN DO;
      OUTCARD = '.NF';
      CALL WRITE;
      OUTCARD = '.NA';
      END;
   ELSE IF PARM='LEFT' THEN DO;
      OUTCARD = '.F';
      CALL WRITE;
      OUTCARD = '.NA';
      END;
   ELSE IF PARM='RIGHT' THEN DO;
      OUTCARD = '.F';
      CALL WRITE;
      CALL ERROR;
      END;
   ELSE IF PARM='CENTER' THEN DO;
      OUTCARD = '.F';
      CALL WRITE;
      CALL ERROR;
      END;
   END;
   RETURN;
END;
END;
The .IL (Indent Line) control word indents the next output line the specified amount of horizontal space. 

- **h** specifies the amount of horizontal space to shift the next output line from the current margin. +h specifies that text is shifted to the right, and -h shifts text to the left.

O is default and initial setting value.

```
ELSE IF COMMAND='IL' THEN DO;
   IF INLEN = 3 THEN OUTCARD = '.U';
   ELSE DO;
      SIGN_POS = VERIFY(SUBSTR(INCARD,4,INLEN-3), '0123456789');
      IF SIGN_POS = 0 THEN DO;
         MOD = SUBSTR(INCARD,4,INLEN-3);
         IF MOD = 0 THEN OUTCARD = '.U';
         ELSE DO;
            IL = IN + MOD;
            IF IL = 0 THEN OUTCARD = '.U';
            ELSE OUTCARD = '.P ' || MOD || ' O';
      END;
   END;
END;

/* Processing continued on next page */
```
/* .IL processing continued */

ELSE DO;
    MOD=SUBSTR(INCARD,SIGN_POS+4,INLEN-SIGN_POS-3);
    SIGN = SUBSTR(INCARD,SIGN_POS+3,1);
    IF SIGN='+' THEN DO:
        IL = IN + MOD;
        IF IL = 0 THEN OUTCARD = '.U';
        ELSE DO:
            DO I=1 TO 3 WHILE (SUBSTR(MOD,I,1)=' ');
            END;
            OUTCARD = '.P + ' || SUBSTR(MOD,I,4-I) || ' 0';
            END;
    END;
    ELSE IF SIGN='-' THEN DO:
        TEMP = IN - MOD;
        IF TEMP ^> 0 THEN OUTCARD = '.U';
        ELSE DO:
            DO I=1 TO 3 WHILE (SUBSTR(MOD,I,1)=' ');
            END;
            OUTCARD = '.P - ' || SUBSTR(MOD,I,4-I) || ' 0';
            END;
    END;
    END;
    CALL WRITE;
    CALL READ;
    DO J=1 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' ');
    END;
    OUTCARD = SUBSTR(INCARD,J,INLEN-J+1);
    RETURN;
END;
The .IN (Indent) changes the left margin displacement of output. 
h specifies the amount of space to be indented. If omitted, 0 is 
assumed, and indentation reverts to the left margin. If +h or -h is 
used, the current left margin is incremented or decremented 
accordingly.
/* .IN processing */

ELSE IF COMMAND='IN' THEN DO;
    IF INLEN=3 THEN DO:
        IN = 0;
        OUTCARD = '.U';
        END;
    ELSE DO:
        SIGN_POS=VERIFY(SUBSTR(INCARD,4,INLEN-3),'0123456789');
        IF SIGN_POS = 0 THEN DO:
            IN = SUBSTR(INCARD,4,INLEN-3);
            IF IN = 0 THEN OUTCARD = '.U';
            ELSE OUTCARD = '.I ' || IN;
            END;
        END;
    ELSE OUTCARD = '.I ' || IN;
    END;
END;

SCRIPT TO RUNOFF CONVERSION PROGRAM

RETURN;
The .IR (Indent Right) specifies the amount of space subsequent input lines are to be indented from the current right margin. h specifies the amount of space to be indented. If omitted, 0 is assumed, and indentation reverts to the right margin. If +h or -h is used, the current right margin is incremented or decremented accordingly.
/* .IR processing */

ELSE IF COMMAND='IR' THEN DO;
  IF INLEN=3 THEN DO;
    IR = 0;
    OUTCARD = '.RU';
  END;
  ELSE DO;
    SIGN_POS=VERIFY(SUBSTR(INCARD,4,INLEN-3),'0123456789');
    IF SIGN_POS = 0 THEN DO:
      IF SUBSTR(INCARD,4,INLEN-3) ^= ' ' THEN DO;
        IR = SUBSTR(INCARD,4,INLEN-3);
      END;
      IF IR = 0 THEN OUTCARD = '.RU';
    ELSE DO;
      IR = SUBSTR(INCARD,SIGNPOS+4,INLEN-SIGNPOS-3);
      SIGN = SUBSTR(INCARD,SIGN_POS+3,1);
      IF SIGN='+' THEN DO;
        IR = IR + MOD;
        OUTCARD = '.R' || IR;
      END;
    ELSE IF SIGN='-' THEN DO;
      TEMP = IR - MOD;
      IF TEMP > 0 THEN DO;
        IR = 0;
        OUTCARD = '.RU';
      END;
    ELSE DO;
      IR = TEMP;
      OUTCARD = '.R' || IR;
    END;
  END;
END;
RETURN;
END;
The .JU (Justify Mode) causes left and right justification of output lines as needed to make the end of each line even with the current right margin.
ON restores right justification of output lines. If neither ON nor Off is specified, ON is assumed.
OFF cancels justification of output lines. If concatenation is still in effect, .JU OFF results in ragged right output.

```plaintext
ELSE IF COMMAND='JU' THEN DO;
  INCARD=TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ','abcdefghijklmnopqrstuvwxyz');
  IF INLEN = 3 THEN OUTCARD = '.AD';
  ELSE DO;
    DO K=4 TO INLEN WHILE(SUBSTR(INCARD,K,i)=' '):
      END;
    IF K>INLEN THEN OUTCARD = '.AD';
    ELSE IF SUBSTR(INCARD,K,2)='ON' THEN OUTCARD = '.AD';
    ELSE IF SUBSTR(INCARD,K,3)='OFF' THEN OUTCARD = '.NA';
    END;
  RETURN;
END;
```
The .LI (Literal) control word allows all input lines, including those that begin with periods, to be processed as text. n specifies the number of input lines to be treated literally. If omitted, 1 is assumed. If .LI n is specified when .LI ON is in effect, literal mode is turned off after n lines or when .LI OFF is encountered. ON specifies that subsequent text lines are to be treated literally. OFF terminates literal mode if it was ON, or if n has been specified and has not been exhausted. line is a line of text to be treated literally. The line is considered to start with the first nonblank character after the .LI control word.

```
ELSE IF COMMAND='LI' THEN DO;
   CALL LI_START;
   RETURN;
END;
```
The .LL (Line Length) control word specifies the length of each subsequent output line. 
h specifies an output line length not greater than the output device capability. If no value is specified for h, the default value established for the device being used will be taken. If +h or -h is used, the current line length is incremented or decremented accordingly.
/ * LL processing */

ELSE IF COMMAND = 'LL' THEN DO:
    IF INLEN = 3 THEN DO:
        LL = 85;
        OUTCARD = '.W 85';
    END;
ELSE DO:
    SIGN_POS = VERIFY (SUBSTR (INCARD, 4, INLEN - 3), '0123456789');
    IF SIGN_POS = 0 THEN DO:
        IF SUBSTR (INCARD, 4, INLEN - 3) ^= '' THEN DO:
            LL = SUBSTR (INCARD, 4, INLEN - 3);
            IF LL > 170 THEN OUTCARD = '.W 170';
            ELSE OUTCARD = '.W ' || LL;
        END;
    END;
ELSE DO:
    MOD = SUBSTR (INCARD, SIGN_POS + 4, INLEN - SIGN_POS - 3);
    $SIGN = SUBSTR (INCARD, SIGN_POS + 3, 1);
    IF SIGN = '+' THEN DO:
        LL = LL + MOD;
        IF LL > 170 THEN OUTCARD = '.W 170';
        ELSE OUTCARD = '.W ' || LL;
    END;
ELSE IF SIGN = '-' THEN DO:
    TEMP = LL - MOD;
    IF TEMP ^= 0 THEN DO:
        LL = 0;
        OUTCARD = '.W 0';
    END;
ELSE DO:
    LL = TEMP;
    OUTCARD = '.W ' || LL;
END;
END;
RETURN;
END;
The .PA (Page Eject) control word causes a page eject, and can set the page number of the new page. 
n specifies the page number of the next page. If n is not specified, sequential page numbering is assumed, and the next page number is one greater than the current page number. n must be an arabic number with no decimal point.

```
/*------------------------------------------------*/

.PA [ n ]

/*------------------------------------------------*/

ELSE IF COMMAND='PA' THEN DO;
    IF INLEN = 3 THEN OUTCARD = '.E';
    ELSE DO;
        DO I=4 TO INLEN WHILE(SUBSTR(INCARD,I,1)=' ');
        END;
        IF VERIFY(SUBSTR(INCARD,I,INLEN-I+1),'0123456789')=0 THEN DO;
            PA = SUBSTR(INCARD,I,INLEN-I+1);
            OUTCARD = '.E';
            CALL WRITE;
            OUTCARD = '.PAG ' || PA;
            END;
        ELSE CALL ERROR;
        END;
    RETURN;
END;
```
The .PL (Page Length) control word specifies the amount of space, including top and bottom margins, for each output page. 

v specifies the vertical length, or depth, of output pages. If no value is specified for v, the default value for the device will be used. If +v or -v is specified, the current page length is incremented or decremented accordingly.
/* .PL processing */

ELSE IF COMMAND='PL' THEN DO;
   IF INLEN = 3 THEN DO:
      PL = 66;
      OUTCARD = '.L';
   END;
ELSE DO:
   SIGN_POS=VERIFY(SUBSTR(INCARD,4,INLEN-3),'0123456789');
   IF SIGN_POS = 0 THEN DO:
      PL = SUBSTR(INCARD,4,INLEN-3);
      IF PL > 132 THEN PL = 132;
      ELSE IF PL < 1 THEN PL = 1;
      OUTCARD = '.L ' || PL;
   END:
ELSE DO:
   SIGN = SUBSTR(INCARD,SIGN_POS+3,1);
   MOD = SUBSTR(INCARD,SIGN_POS+4,INLEN-SIGN_POS-3);
   IF SIGN = '+' THEN DO:
      PL = PL + MOD;
      IF PL > 132 THEN PL = 132;
      OUTCARD = '.L ' || PL;
   END:
ELSE IF SIGN = '-' THEN DO:
   PL = PL - MOD;
   IF PL < 1 THEN PL = 1;
   OUTCARD = '.L ' || PL;
   END:
ELSE CALL ERROR;
END;
RETURN;
END;
.PP [ line ]

The .PP (Paragraph Start) control word begins formatting the output line as the start of a paragraph after a skip. line is the text that begins a new paragraph. If line is omitted, the text from the next input line after the .PP control word begins the new paragraph.

ELSE IF COMMAND='PP' THEN DO;
  OUTCARD='.P 3 1';
  IF INLEN = 3 THEN RETURN;
  ELSE DO;
    CALL WRITE;
    DO J=4 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' ');
    END;
    OUTCARD = SUBSTR(INCARD,J,INLEN-J+1);
    END;
  RETURN;
END;
The .QQ (Quick Quit) control word causes Script processing to terminate immediately.
The `.SK` (Skip Lines) control word specifies the amount of space to insert before the next text output line, except at the top of a column or page. 

V is the amount of space to be inserted in the output. If no number is given, 1 line is assumed.
The .SP (Space Lines) control word specifies the amount of space to insert before the next text output line. The specified number of lines are inserted even when the .SP occurs at the top of a page or column.

* Processing on next page */
/* .SP processing */
ELSE IF COMMAND='SP' THEN DO;
   IF INLEN = 3 THEN DO;
      OUTCARD = '.BL ~';
      CALL WRITE;
      OUTCARD = '.B';
      CALL WRITE;
      OUTCARD = '~';
      CALL WRITE;
      OUTCARD = '.B';
      END;
   ELSE DO;
      IF VERIFY(SUBSTR(INCARD,4,INLEN-3), '0123456789')=0 THEN DO;
         TEMP_NUM = SUBSTR(INCARD,4,INLEN-3);
         OUTCARD = '.BL ~';
         CALL WRITE;
         DO I=1 TO TEMP_NUM;
            OUTCARD = '.B';
            CALL WRITE;
            OUTCARD = '~';
            CALL WRITE;
            END;
         OUTCARD = '.B';
         END;
      ELSE DO;
         CALL ERROR;
         RETURN;
      END;
   END;
RETURN;
END:
The .SS (Single-Spaced Mode) control word causes subsequent output lines to be single-spaced.

ELSE IF COMMAND='SS' THEN DO;
  OUTCARD='SP 1';
  RETURN;
END;
The .SX control word is used to split a string of text between the left and right column margins, with a filler in between the two.

C specifies that the middle part of the string is to be centered. / is any delimiter character. The first non blank character will be taken as the delimiter character.

lpart is the string to be placed against the current left margin.
fill is the string to be centered.
rpart is the string to be placed against the current right margin.

```bash
ELSE IF COMMAND='SX' THEN DO;
    DO I=4 TO INLEN WHILE(SUBSTR(INCARD,I,1)==' ');
    END;
    IF SUBSTR(INCARD,I,1)=='C' & SUBSTR(INCARD,I,1)!='c' THEN DO;
        CALL ERROR;
        RETURN;
    END;
    ELSE DO;
        DO J=I+1 TO INLEN WHILE(SUBSTR(INCARD,J,1)==' ');
        END;
        DO I=J+1 TO INLEN WHILE(SUBSTR(INCARD,I,1)!='' & SUBSTR(INCARD,I,1)!='c' & SUBSTR(INCARD,I,1)!='C');
        PART1 = SUBSTR(INCARD,I+1,J-I-J);
        DO J=I+1 TO INLEN WHILE(SUBSTR(INCARD,J,1)!='' & SUBSTR(INCARD,J,1)!='c' & SUBSTR(INCARD,J,1)!='C');
        PART2 = SUBSTR(INCARD,J+1,I-J-1);
        DO I=J+1 TO INLEN WHILE(SUBSTR(INCARD,I,1)!='' & SUBSTR(INCARD,I,1)!='c' & SUBSTR(INCARD,I,1)!='C');
        PART3 = SUBSTR(INCARD,J+1,I-J-1);
        OUTCARD = '/| PART1 || /| PART2 || /| PART3 || /|';
        PART1 = '/';
        PART2 = '/';
        PART3 = '/';
        RETURN;
    END;
END;
```
The .TB (Tab Setting) control word defines how tab characters (hexadecimal 05) are to be resolved. SET specifies that all the old tab stops are to be cleared and a new set of tab stops is to be defined. h specifies the horizontal displacements of the tab stops.

```plaintext
ELSE IF COMMAND='TB' THEN DO;
  INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                     'abcdefghijklmnopqrstuvwxyz');
  IF VERIFY(SUBSTR(INCARD,4,INLEN-3),0123456789)=0 THEN DO;
    OUTCARD = '.TA ' || TAB_CHAR ' ' || INCARD;
    RETURN;
  END;
  ELSE DO;
    DO I=4 TO INLEN WHILE(SUBSTR(INCARD,I,1)=' ');
    CALL ERROR;
    RETURN;
  END;
ELSE DO;
  OUTCARD='.'TAB_CHAR ' ' SUBSTR(INCARD,I+3,INLEN-I-2);
  RETURN;
END;
END;
```
The .TI control word translates the input text from one input representation to another.
s is the source character to be translated.
t is the desired output representation of the source character.

```
ELSE IF COMMAND='TI' THEN DO;
  DO I=4 TO INLEN WHILE (SUBSTR(INCARD,I,1)=' ');
  END;
  IF I=INLEN THEN DO;
    CALL ERROR;
    RETURN;
    END;
  ELSE DO;
    TAB_CHAR = SUBSTR(INCARD,I,1);
    DO I=I+1 TO INLEN WHILE(SUBSTR(INCARD,I,1)=' ');
    END;
    IF I=INLEN | SUBSTR(INCARD,I,2)='05' THEN DO;
      TAB_CHAR = ' ';
      CALL ERROR;
      END;
      NO_WRITE = '1'B;
      RETURN;
      END;
      END;
```
The .TM (Top Margin) control word specifies the amount of space in the top margin area.

v specifies the amount of vertical space to be reserved at the top of the output pages. If no value is specified for v, the default value for the logical device will be used.

+v or -v increases or decreases the existing top margin by the amount given.
*/ .TM processing */

ELSE IF COMMAND='TM' THEN DO;
   IF INLEN=3 THEN DO;
      OUTCARD=INCARD;
      TM = 5;
   END;
   ELSE DO;
      SIGN_POS=VERIFY(SUBSTR(INCARD,4,INLEN-3),' 0123456789');
      IF SIGN_POS = 0 THEN DO;
         OUTCARD=INCARD;
         IF SUBSTR(INCARD,4,INLEN-3) ^= '' THEN DO;
            TM = SUBSTR(INCARD,4,INLEN-3);
            /* Runoff specification */
            IF TM = 0 THEN TM = 5;
         END;
      END;
      ELSE TM = 5;
   END;
   ELSE DO;
      MOD = SUBSTR(INCARD,SIGN_POS+4,INLEN-SIGN_POS-3);
      SIGN=SUBSTR(INCARD,SIGN_POS+3,1);
      IF SIGN='+' THEN TM = TM + MOD;
      ELSE IF SIGN='-' THEN DO;
         TEMP = TM - MOD;
         IF TEMP ^= 0 THEN TM = 1;
         ELSE TM = TEMP;
      END;
      OUTCARD = '.TM ' || TM;
   END;
END;
RETURN;
END;
The \texttt{.UC} (Underscore and Capitalize) control word underscores and capitalizes one or more input lines. However, because of Runoff restrictions, this program treats the \texttt{.UC} command as the \texttt{.UP} (Uppercase) command. \texttt{n} specifies the number of input lines to be converted to uppercase. If omitted, 1 is assumed. If \texttt{.UC} \texttt{n} is specified when \texttt{.UC ON} is in effect, capitalization is turned off when \texttt{n} lines have been capitalized, or when \texttt{.UC OFF} is encountered. \texttt{ON} specifies that subsequent text lines are to be capitalized. \texttt{OFF} terminates capitalization mode if it was \texttt{ON}, or if \texttt{n} has been specified and has not been exhausted. \texttt{line} is a line of text to be capitalized. The line is considered to start with the first nonblank character after the \texttt{.UC} control word.

```bash
/------------------------------------------------------------------------

\texttt{.UC [ 1 \mid n \mid ON \mid OFF \mid line ]}

ELSE IF COMMAND = 'UC' THEN DO;
   CALL UC\_START;
   RETURN;
END;
```

**Pseudocode:**
```bash
/------------------------------------------------------------------------

\texttt{.UC [ 1 \mid n \mid ON \mid OFF \mid line ]}

The \texttt{.UC} (Underscore and Capitalize) control word underscores and capitalizes one or more input lines. However, because of Runoff restrictions, this program treats the \texttt{.UC} command as the \texttt{.UP} (Uppercase) command. \texttt{n} specifies the number of input lines to be converted to uppercase. If omitted, 1 is assumed. If \texttt{.UC} \texttt{n} is specified when \texttt{.UC ON} is in effect, capitalization is turned off when \texttt{n} lines have been capitalized, or when \texttt{.UC OFF} is encountered. \texttt{ON} specifies that subsequent text lines are to be capitalized. \texttt{OFF} terminates capitalization mode if it was \texttt{ON}, or if \texttt{n} has been specified and has not been exhausted. \texttt{line} is a line of text to be capitalized. The line is considered to start with the first nonblank character after the \texttt{.UC} control word.

```
SCRIPT TO RUNOFF CONVERSION PROGRAM

UN [ 0 | h | +h | -h ]

The .UN (Undent) control word causes the next output line's indentation to change: it is moved to the left of the current margin. h specifies the amount of horizontal space by which the indentation is to be altered for the next output line only. If -h is specified, the .UN control word is effectively the same as the .IL (Indent Line) control word. If omitted, 0 is assumed, and the indentation is not changed.

ELSE IF COMMAND='UN' THEN DO;
   IF INLEN = 3 THEN OUTCARD = '.U';
   ELSE DO;
      SIGN_POS = VERIFY(SUBSTR(INCARD,4,INLEN-3), '0123456789');
      IF SIGN_POS = 0 THEN DO;
         MOD = SUBSTR(INCARD,4,INLEN-3);
         IF MOD = 0 THEN OUTCARD = '.U';
      ELSE DO;
         TEMP = IN - MOD;
         IF TEMP ^> 0 THEN OUTCARD = '.U';
      ELSE DO;
         UN = TEMP;
         OUTCARD = '.I ' || UN;
      END;
   END;
END;

/* Processing continued on next page */
/* .UN processing continued */

ELSE DO:
    MOD = SUBSTR(INCARD, SIGN_POS + 4, INLEN - SIGN_POS - 3);
    SIGN = SUBSTR(INCARD, SIGN_POS + 3, 1);
    IF SIGN = '-' THEN DO:
        UN = IN + MOD;
        OUTCARD = '.I' || UN;
        END;
    ELSE IF SIGN = '+' THEN DO:
        TEMP = IN - MOD;
        IF TEMP ^> 0 THEN OUTCARD = '.U';
        ELSE DO:
            UN = TEMP;
            OUTCARD = '.I' || UN;
            END;
        END;
    END;
    CALL WRITE;
    CALL READ;
    DO J=1 TO INLEN WHILE (SUBSTR(INCARD, J, 1)=' ');
    END;
    OUTCARD = SUBSTR(INCARD, J, INLEN-J+1);
    CALL WRITE;
    IF IN = 0 THEN OUTCARD = '.U';
    ELSE OUTCARD = '.I' || IN;
    RETURN;
END;
The .UP (Uppercase) control word prints one or more subsequent input
lines in UPPERCASE characters.
n specifies the number of input lines to be converted to uppercase.
If omitted, 1 is assumed. If .UP n is specified when .UP ON is in
effect, capitalization is turned off when n lines have been
capitalized, or when .UP OFF is encountered.
ON specifies that subsequent text lines are to be capitalized.
OFF terminates capitalization mode if it was ON, or if n has been
specified and has not been exhausted.
line is a line of text to be capitalized. The line is considered to
start with the first nonblank character after the .UP control word.

ELSE IF COMMAND = 'UP' THEN DO;
   CALL UP_START;
   RETURN;
END;
.US [ 1 | n | ON | OFF | line ]

The .US (Underscore) control word underscores one or more input lines. However, because of Runoff restrictions, this program treats the .US command as the .UP (Uppercase) command.
n specifies the number of input lines to be converted to uppercase. If omitted, 1 is assumed. If .US n is specified when .US ON is in effect, capitalization is turned off when n lines have been capitalized, or when .US OFF is encountered.
ON specifies that subsequent text lines are to be capitalized.
OFF terminates capitalization mode if it was ON, or if n has been specified and has not been exhausted.
line is a line of text to be capitalized. The line is considered to start with the first nonblank character after the .US control word.

ELSE IF COMMAND = 'US' THEN DO;
   CALL US_START;
   RETURN;
END;
The .* control word allows comments to be stored in the Script file for future reference.

/* [comments]*/

ELSE IF SUBSTR(COMMAND, 1, 1) = '*' THEN DO:
    OUTCARD = INCARD;
    RETURN;
    END;

ELSE CALL ERROR;

END PROCESS;
SCRIPT TO RUNOFF CONVERSION PROGRAM

/* Start of .CE (center) control word processing */

CE_START: PROC RECURSIVE;

IF INLEN = 3 THEN DO;
    CALL READ;
    CALL CE;
    RETURN;
END;
ELSE DO:
    DO J=4 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' ');
    END;
    IF VERIFY(SUBSTR(INCARD,J,INLEN-J+1),'0123456789')=0 THEN DO;
        CALL CE_N;
        RETURN;
    END;
ELSE DO;
    PARM = SUBSTR(INCARD,J,INLEN-J+1);
    TEMPCARD = PARM;
    PARM = TRANSLATE(PARM,'FNO','fno');
    IF PARM = 'ON' THEN DO:
        CALL CE_ON;
        RETURN;
    END;
    ELSE IF PARM = 'OFF' THEN DO:
        NO_WRITE = '1'B;
        RETURN;
    END;
    ELSE DO:
        OUTCARD = '.' > TEMPCARD;
        RETURN;
    END;
    RETURN;
END;
END CE_START;
// Centering processing */

CE: PROC RECURSIVE:

  IF INCARD = ' ' THEN DO:
    NO_WRITE = '1'B;
    RETURN;
  END;

  ELSE DO:
    IF SUBSTR(INCARD,1,1)='.' THEN DO:
      DO J=1 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' '):
      END;
      OUTCARD = '.>' || SUBSTR(INCARD,J,INLEN-J+1);
      RETURN;
  END;

END CE;
/* Processing of .CE control word having option n */

CE_N:

PROC RECURSIVE:

K = SUBSTR(INCARD,J,INLEN-J+1);
IF K>0 THEN DO I=1 TO K WHILE("EDF");
  CALL READ;
  IF SUBSTR(INCARD,1,1)^='.' THEN DO;
    CALL CE;
    CALL WRITE;
  END;
ELSE DO:
  NO_READ = '1'B;
  RETURN;
  END;
END;
NO_WRITE = '1'B;
END CE_N;
/* Processing of .CE control word having option ON */

CE_ON: PROC RECURSIVE;

DO WHILE(^EOF);
   CALL READ;
   COMMAND = TRANSLATE(SUBSTR(INCARD,2,2), 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz');
   IF COMMAND ^= 'CE' THEN DO;
      CALL CE;
      CALL WRITE;
      END;
   ELSE DO;
      CALL CE_START;
      RETURN;
      END;
   END;
END CE_ON;
SCRIPT TO RUNOFF CONVERSION PROGRAM

/* Start of .LI (Literal) control word processing */

LI_START: PROC RECURSIVE;

IF INLEN = 3 THEN DO;
    CALL READ;
    CALL LI;
    RETURN;
END;
ELSE DO;
    DO J=4 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' ');
    END;
    IF VERIFY(SUBSTR(INCARD,J,INLEN-J+1),'0123456789')=O THEN DO;
        CALL LI_N;
        RETURN;
    END;
    ELSE DO;
        PARM = SUBSTR(INCARD,J,INLEN-J+1);
        TEMPCARD = PARM;
        PARM = TRANSLATE(PARM,'FNO','fno');
        IF PARM = 'ON' THEN DO;
            CALL LI_ON;
            RETURN;
        END;
        ELSE IF PARM = 'OFF' THEN DO;
            NO_WRITE = '1'B;
            RETURN;
        END;
        ELSE DO;
            OUTCARD = '.+ ' TEMPCARD;
            RETURN;
        END;
    END;
END LI_START;
/* Literal processing */

LI: PROC RECURSIVE;

IF INCARD = ' ' THEN DO:
    NO_WRITE = '1'B;
    RETURN;
END;
ELSE DO:
    DO j=1 TO INLEN WHILE(SUBSTR(INCARD,j,1)=' '):
    END;
    OUTCARD = '.' " SUBSTR(INCARD,j,INLEN-J+1);
    RETURN;
END;
END LI;
/* Processing of .LI control word having option n */

LI_N:

PROC RECURSIVE;

K = SUBSTR(INCARD,J,INLEN-J+1);
IF K>0 THEN DO I=1 TO K WHILE (^EOF);
   CALL READ;
   CALL LI;
   CALL WRITE;
   END;
NO_WRITE = '1'B;
END LI_N;
END;
/* Processing of .LI control word having option ON */

LI_On:
PROC RECURSIVE;
DO WHILE(^EOF);
    CALL READ;
    COMMAND = TRANSLATE(SUBSTR(INCARD,2,2), 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz');
    IF COMMAND ^= 'LI' THEN DO;
        CALL LI;
        CALL WRITE;
        END;
    ELSE DO;
        CALL LI_START;
        RETURN;
        END;
END;
END LI_On;
SCRIPT TO RUNOFF CONVERSION PROGRAM

/* Start of .UC control word processing */

UC_START: PROC RECURSIVE;

INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                  'abcdefghijklmnopqrstuvwxyz');

IF INLEN = 3 THEN DO;
   CALL READ;
   CALL UC;
   RETURN;
END;
ELSE DO;
   DO J=4 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' ');
END;

IF VERIFY(SUBSTR(INCARD,J,INLEN-J+1),'0123456789')=0 THEN DO;
   CALL UC_N;
   RETURN;
END;
ELSE DO;
   PARM = SUBSTR(INCARD,J,INLEN-J+1);
   IF PARM = 'ON' THEN DO;
      CALL UC_ON;
      RETURN;
   END;
   ELSE IF PARM = 'OFF' THEN DO;
      NO_WRITE = '1'B;
      RETURN;
   END;
   ELSE DO;
      OUTCARD = PARM;
      RETURN;
   END;
END;
END UC_START;
/* Processing of .UC control word under Runoff restrictions */

UC:
PROC RECURSIVE;

INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                   'abcdefghijklmnopqrstuvwxyz');

IF INCARD = '' THEN DO:
   NO_WRITE = '1'B;
   RETURN;
   END;
ELSE DO:
   DO J=1 TO INLEN WHILE (SUBSTR(INCARD,J,1)=' ');
   END;
   OUTCARD = SUBSTR(INCARD,J,INLEN-J+1);
   RETURN;
END;
END UC;
SCRIPT TO RUNOFF CONVERSION PROGRAM

/* Processing of .UC control word having option n */

UC_N: PROC RECURSIVE;

    K = SUBSTR(INCARD,J,INLEN-J+1);
    IF K>0 THEN DO I=1 TO K WHILE(^EOF);
        CALL READ;
        CALL UC;
        CALL WRITE;
    END;

    NO_WRITE = '1'B;
END UC_N;

SCRIPT TO RUNOFF CONVERSION PROGRAM

/* Processing of .UC control word having option ON */

UC_ON:

PROC RECURSIVE;

DO WHILE(^EOF);
    CALL READ;
    INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz');
    COMMAND = SUBSTR(INCARD,2,2);
    IF COMMAND ^=IUCI THEN DO;
        CALL UC;
        CALL WRITE;
        END;
    ELSE DO;
        CALL UC_START;
        RETURN;
        END;
    END;
END UC_ON;
/* Start of .UP (Uppercase) control word processing */

UP_START: PROC RECURSIVE;

INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                 'abcdefghijklmnopqrstuvwxyz');

IF INLEN = 3 THEN DO;
    CALL READ;
    CALL UP;
    RETURN;
END;
ELSE DO;
    DO J=4 TO INLEN WHILE(SUBSTR(INCARD,J,I)=' ');
END;

IF VERIFY(SUBSTR(INCARD,J,INLEN-J+1),'0123456789')=O THEN DO;
    CALL UP_N;
    RETURN;
END;
ELSE DO:
    PARM = SUBSTR(INCARD,J,INLEN-J+1);
    IF PARM = 'ON' THEN DO:
        CALL UP_ON;
        RETURN;
    END;
    ELSE IF PARM = 'OFF' THEN DO:
        NO_WRITE = '1' 'B;
        RETURN;
    END;
    ELSE DO:
        OUTCARD = PARM;
        RETURN;
    END;
END;
END UP_START;
/* Capitalization processing */
UP: PROC RECURSIVE;
    INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                      'abcdefghijklmnopqrstuvwxyz');
    IF INCARD = ' ' THEN DO;
        NO_WRITE = '1';
        RETURN;
    END;
    ELSE DO;
        DO J=1 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' ');
        END;
        OUTCARD = SUBSTR(INCARD,J,INLEN-J+1);
        RETURN;
    END;
END UP;
/* Processing of .UP control word having option n */

UP_N:

PROC RECURSIVE;

  K = SUBSTR(INCARD,J,INLEN-J+1);
  IF K>0 THEN DO I=1 TO K WHILE(^EOF):
     CALL READ;
     CALL UP;
     CALL WRITE;
   END;
  NO_WRITE = '1' B;
END UP_N;

/* Processing of .UP control word having option ON */
UP_ON:
PROC RECURSIVE;
DO WHILE(^EOF);
    CALL READ;
    INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
        'abcdefghijklmnopqrstuvwxyz');
    COMMAND = SUBSTR(INCARD,2,2);
    IF COMMAND ^= 'UP' THEN DO;
        CALL UP;
        CALL WRITE;
    END;
    ELSE DO;
        CALL UP_START;
        RETURN;
    END;
END UP_ON;
/* Start of .US (Underscore) control word processing */

US_START:

PROC RECURSIVE:

    INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
        'abcdefghijklmnopqrstuvwxyz');

    IF INLEN = 3 THEN DO;
        CALL READ;
        CALL US;
        RETURN;
    END;

    ELSE DO;
        DO J=4 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' ');
    END;

    IF VERIFY(SUBSTR(INCARD,J,INLEN-J+1),'0123456789')=O THEN DO;
        CALL US_N;
        RETURN;
    END;

    ELSE DO;
        PARM = SUBSTR(INCARD,J,INLEN-J+1);
        IF PARM = 'ON' THEN DO;
            CALL US_ON;
            RETURN;
        END;

        ELSE IF PARM = 'OFF' THEN DO;
            NO_WRITE = '1'B;
            RETURN;
        END;

        ELSE DO;
            OUTCARD = PARM;
            RETURN;
        END;

    END;

END US_START;
/* Processing of .US control word under Runoff restrictions */
US:
    PROC RECURSIVE;
    INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                        'abcdefghijklmnopqrstuvwxyz');
    IF INCARD = ' ' THEN DO;
        NO_WRITE = '1'B;
        RETURN;
    END;
    ELSE DO;
        DO J=1 TO INLEN WHILE(SUBSTR(INCARD,J,1)=' ');
        END;
        OUTCARD = SUBSTR(INCARD,J,INLEN-J+1);
        RETURN;
    END;
END US;
/* Processing of .US control word having option n */

US_N:
PROC RECURSIVE;

   K = SUBSTR(INCARD,J.INLEN-J+1);
   IF K>0 THEN DO I=1 TO K WHILE(^EOF); 
      CALL READ;
      CALL US;
      CALL WRITE;
   END;
   NO_WRITE = '1'B;
END US_N;
/* Processing of .US control word having option ON */

US_ON: PROC RECURSIVE:

DO WHILE(^EOF):
    CALL READ;
    INCARD = TRANSLATE(INCARD,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                      'abcdefghijklmnopqrstuvwxyz');
    COMMAND = SUBSTR(INCARD,2,2);
    IF COMMAND ^= 'US' THEN DO;
        CALL US;
        CALL WRITE;
    END;
    ELSE DO;
        CALL US_START;
        RETURN;
    END;
END US_ON;
SCRIPT TO RUNOFF CONVERSION PROGRAM

/* Error processing */

ERROR: PROC;

  OUTCARD = '* THE FOLLOWING SCRIPT COMMAND CANNOT BE PROCESSED:';
  CALL WRITE;
  OUTCARD = '* ' || INCARD;
  END ERROR;

READ: PROC;

  INCARD = ' ';
  READ FILE(IN) INTO(INREC);
  INCOUNT = INCOUNT + 1;
  END READ;

WRITE: PROC;

  OUTLEN = LENGTH(OUTCARD);
  WRITE FILE(OUT) FROM(OUTREC);
  OUTCOUNT = OUTCOUNT + 1;
  OUTCARD = '';
  END WRITE;

END STR;
7. SAMPLE RUN

Following is a sample run of a STR session. The input file name is TEST.
The following commands were executed:

C>CO TE
(compile and link STR.PL1G)

C>TYPEF TEST
(print the input file (TEST) on the terminal)

C>TYPEF STR.CPL
(print the cpl file which executes the program)

C>STR TEST
(convert Script file TEST into Runoff file)

C>TYPEF TEST.RUNOFF
(print the output file (TEST.RUNOFF) on the terminal)
SCRIPT TO RUNOFF CONVERSION PROGRAM

C>CD TE
C>filmem
C>date
Saturday, May 14, 1983 12:46 AM
C>pl1g str.pl1g
0000 ERRORS (PL1G-REV 18.3 )
C>seg
[SEG rev 18.3]
# load str
$ load str
$ li pl1glb
$ li
LOAD COMPLETE
$ save
$ quit
C>date
Saturday, May 14, 1983 12:47 AM
C>cominput -end
C>TYPEF TEST
This file is used to test the STR (Script To Runoff) program.
Its main use is to see if the Script commands are converted properly to Runoff commands.

*/-----------------------------------------------*

.BM [ v | +v | -v ]

The .BM (Bottom Margin) control word specifies the amount of space to be reserved at the bottom of the output pages, overriding the initial value established for the device.

v specifies the amount of space to be reserved at the bottom of output pages. If +v or -v is specified, the current value of the bottom margin is incremented or decremented. If no value is specified for v, the initial setting is restored.

.bm 5
Next amount should be 3
.bm -2
Next amount should be 4
.bm +1
.bm

---------------------------------------------------------------------

.BR

The .BR (Break) control word prevents the concatenation of the following text line with preceding text.

.br

---------------------------------------------------------------------

.CD n

The .CD (Column Definition) control word defines how many columns of output are to be formatted on each page.

n is the number of columns of output to be formatted onto each subsequent output page. It may be any number from 1 to 9.

.cd 3
.cd 2
The `.CE (Center)` control word centers text lines between the current left and right margins.

- `n` specifies the number of input lines to be centered. If omitted, 1 is assumed. If `.CE n` is specified when `.CE ON` is in effect, centering is turned off when `n` lines have been centered, or when `.CE OFF` is encountered.
- `.CE ON` specifies that subsequent text lines are to be centered.
- `.CE OFF` terminates centering mode if it was `ON`, or if `n` has been specified and has not been exhausted.
- `line` is a line of text to be centered. The line is considered to start with the first nonblank character after the `.CE control word.

```
This line is to be centered.
This line is not to be centered, but the next 3 lines are.
   .ce 3
   One
   Two
   Three
   Next line is to be centered.
   .ce This line is centered.
   The following several lines are centered until .ce off is encountered.
   .ce on
center
center
center
center
   last in group to be centered
   .ce off
Next command is .ce 555 Main Street, program should center only this one line instead of the next 555 lines according to Script specification.
   .ce 555 Main Street
This line is not to be centered.
```
The .CM (Comment) control word allows comments to be stored in the Script file for future reference.

This is a comment line.

The .CO (Concatenate Mode) cancels or restores concatenation of input lines and truncation at the current column length. ON restores concatenation of input lines. ON is the initial setting, as well as the default value. OFF cancels concatenation of input lines. If justification is still in effect, .CO OFF results in each line being padded with blanks to column length.

Next Script command is .co on
Next Script command is .co off
The .DS (Double Space mode) causes subsequent output lines to be doublespaced.

Next Script command is .ds

The .FO (Format mode) controls concatenation and justification of input lines. ON restores default formatting, including both justification and concatenation of lines. If the .FO control word is used with no operands, ON is assumed. OFF cancels concatenation of input lines and justification of output lines. LEFT specifies that the input lines are to be concatenated but not justified.

Next Script command is .fo on

---
The .IL (Indent Line) control word indents the next output line the specified amount of horizontal space. 
  h specifies the amount of horizontal space to shift the next output line from the current margin. +h specifies that text is shifted to the right, and -h shifts text to the left. 
  0 is default and initial setting value.

Next Script command is .il
 .il
Next Script command is .il 3
 .il 3
Next Script command is .il -1
 .il -1
Next Script command is .il +2
 .il +2

The .IN (Indent) changes the left margin displacement of output.
  h specifies the amount of space to be indented. If omitted, 0 is assumed, and indentation reverts to the left margin. If +h or -h is used, the current left margin is incremented or decremented accordingly.

Next Script command is .in
 .in
Next Script command is .in 9
 .in 9
Next Script command is .in +2
 .in +2
Next Script command is .in -8
 .in -8
The .IR (Indent Right) specifies the amount of space subsequent input lines are to be indented from the current right margin.

h specifies the amount of space to be indented. If omitted, 0 is assumed, and indentation reverts to the right margin. If +h or -h is used, the current right margin is incremented or decremented accordingly.

Next Script command is .ir
  .ir
Next Script command is .ir 7
  .ir 7
Next Script command is .ir -4
  .ir -4
Next Script command is .ir + 3
  .ir + 3

The .JU (Justify Mode) causes left and right justification of output lines as needed to make the end of each line even with the current right margin.

ON restores right justification of output lines. If neither ON nor OFF is specified, ON is assumed.

OFF cancels justification of output lines. If concatenation is still in effect,.JU OFF results in ragged right output.

Next Script command is .ju
  .ju
Next Script command is .ju off
  .ju off
Next Script command is .ju on
  .ju on
The .LI (Literal) control word allows all input lines, including those that begin with periods, to be processed as text. 

n specifies the number of input lines to be treated literally. If omitted, 1 is assumed. If .LI n is specified when .LI ON is in effect, literal mode is turned off after n lines or when .LI OFF is encountered.

ON specifies that subsequent text lines are to be treated literally. OFF terminates literal mode if it was ON, or if n has been specified and has not been exhausted.

line is a line of text to be treated literally. The line is considered to start with the first nonblank character after the .LI control word.

.This line is to be treated literally despite the period at the beginning.

The next 3 lines are treated literally.

.One
.Two
.Three

Next line is to be treated literally.

.This line is treated literally.

The following several lines are treated literally until .LI off is encountered.

.on
.l literal
.l literal
.l literal
.l literal
..last in group to be treated literally
..off

Next command is .LI 555 Main Street, program should treat only this one line literally instead of the next 555 lines according to Script specification.

.555 Main Street

.ce This line is to be centered and not treated literally.
The .LL (Line Length) control word specifies the length of each subsequent output line. 
h specifies an output line length not greater than the output device capability. If no value is specified for h, the default value established for the device being used will be taken. If +h or -h is used, the current line length is incremented or decremented accordingly.

Next Script command is .11 55
 .11 55
Next Script command is .11 -10
 .11 -10
Next Script command is .11 +3
 .11 +3

The .PA (Page Eject) control word causes a page eject, and can set the page number of the new page. n specifies the page number of the next page. If n is not specified, sequential page numbering is assumed, and the next page number is one greater than the current page number. n must be an arabic number with no decimal point.

Next Script command is .pa
 .pa
Next Script command is .pa 5
 .pa 5
The .PL (Page Length) control word specifies the amount of space, including top and bottom margins, for each output page. \( v \) specifies the vertical length, or depth, of output pages. If no value is specified for \( v \), the default value for the device will be used. If \(+v\) or \(-v\) is specified, the current page length is incremented or decremented accordingly.

Next Script command is .pl 55

Next Script command is .pl -8

Next Script command is .pl +2

The .PP (Paragraph Start) control word begins formatting the output line as the start of a paragraph after a skip. \( \text{line} \) is the text that begins a new paragraph. If \( \text{line} \) is omitted, the text from the next input line after the .PP control word begins the new paragraph.

Next Script command is .pp This is the beginning of a paragraph.

Next Script command is .pp followed by a separate input line.

This is the separate input line.

The .QQ (Quick Quit) control word causes Script processing to terminate immediately.

Next Script command is .qq
The .SK (Skip Lines) control word specifies the amount of space to insert before the next text output line, except at the top of a column or page. 

Next Script command is .sk

The .SP (Space Lines) control word specifies the amount of space to insert before the next text output line. The specified number of lines are inserted even when the .SP occurs at the top of a page or column.

Next Script command is .sp

The .SS (Single-Spaced Mode) control word causes subsequent output lines to be single-spaced.
The .SX (Split Text) control word is used to split a string of text between the left and right column margins, with a filler in between the two. 
C specifies that the middle part of the string is to be centered. 
/ is any delimiter character. The first non blank character will be taken as the delimiter character. 
lpart is the string to be placed against the current left margin. 
fill is the string to be centered. 
rpart is the string to be placed against the current right margin.

Next Script command is .sx c /left/middle/right/ 
   .sx c /left/middle/right/
Next Script command is .sc c *leftleft*midmid*rightright* 
   .sx c *leftleft*midmid*rightright* 
   --------------------------------------------------------------*/
   /*-----------------------------------------------*/

The .TB (Tab Setting) control word defines how tab characters (hexadecimal 09) are to be resolved. 
SET specifies that all the old tab stops are to be cleared and a new set of tab stops is to be defined. 
h specifies the horizontal displacements of the tab stops.

Next Script command is .tb set 5 20 30 
   .tb set 5 20 30 --------------------------------------------------------------*/
The .TI control word translates the input text from one input representation to another. s is the source character to be translated. t is the desired output representation of the source character. Currently, this command is processible only when defining tab character.

Next Script command is .ti @ 05 (to reset tab character to @)

Next Script command is .tb 10 20 30

The .TM (Top Margin) control word specifies the amount of space in the top margin area. v specifies the amount of vertical space to be reserved at the top of the output pages. If no value is specified for v, the default value for the logical device will be used. +v or -v increases or decreases the existing top margin by the amount given.

Next Script command is .tm 5

Next Script command is .tm -3

Next Script command is .tm +2
The .UC (Underscore and Capitalize) control word underscores and capitalizes one or more input lines.

**HOWEVER, BECAUSE OF RUNOFF RESTRICTIONS, THIS PROGRAM TREATS THE .UC COMMAND AS THE .UP (UPPERCASE) COMMAND.**

- **n** specifies the number of input lines to be converted to uppercase.
- If omitted, 1 is assumed. If .UC n is specified when .UC ON is in effect, capitalization is turned off when n lines have been capitalized, or when .UC OFF is encountered.
- ON specifies that subsequent text lines are to be capitalized.
- OFF terminates capitalization mode if it was ON, or if n has been specified and has not been exhausted.
- **line** is a line of text to be capitalized. The line is considered to start with the first nonblank character after the .UC control word.

Next Script command is .uc this is the input line.
\[ .uc this is the input line. \]
Next Script command is .uc 2
\[ .uc 2 \]
\[ one \]
\[ two \]
Next Script command is .uc on
\[ .uc on \]
\[ on \]
\[ on \]
\[ on \]
next line is .uc off
\[ .uc off \]
next line is .uc on
\[ .uc on \]
\[ on \]
\[ on \]
\[ on \]
next line is .uc 2 so only 2 more lines will be capitalized
\[ .uc 2 \]
\[ one \]
\[ two \]
\[ three \]
The .UN (Untdent) control word causes the next output line’s indentation to change: it is moved to the left of the current margin. h specifies the amount of horizontal space by which the indentation is to be altered for the next output line only. If -h is specified, the UN control word is effectively the same as the .IL (Indent Line) control word. If omitted, 0 is assumed, and the indentation is not changed.

Next Script command is .in 9
.in 9
Next Script command is .un
.un
This line is to be undented.
Next Script command is .un -7
.un 7
Line to be undented.
Next Script command is .un +3
.un +3
Line to be undented.
Next Script command is .un -4
.un -4
Line to be undented.
Next Script command is .un
.un
Line to be undented.

/*-----------------------------------------------*/
The .UP (Uppercase) control word prints one or more subsequent input lines in UPPERCASE characters.

n specifies the number of input lines to be converted to uppercase. If omitted, 1 is assumed. If .UP n is specified when .UP ON is in effect, capitalization is turned off when n lines have been capitalized, or when .UP OFF is encountered. ON specifies that subsequent text lines are to be capitalized. OFF terminates capitalization mode if it was ON, or if n has been specified and has not been exhausted. Line is a line of text to be capitalized. The line is considered to start with the first nonblank character after the .UP control word.

Next Script command is .up this is the input line.
.up this is the input line.
Next Script command is .up 2
.up 2
one
two
Next Script command is .up on
.up on
.on
.on
.on
next line is .up off
.up off
next line is .up on
.up on
.on 1
.on 2
next line is .up 2 so only 2 more lines will be capitalized
.up 2
one
two
three

/* -----------------------------------------------
   .UP [ 1 | n | ON | OFF | line ]
*/
The .US (Underscore) control word underscores one or more input lines. However, because of RUNOFF restrictions, this program treats the .US command as the .UP (UPPERCASE) command.

n specifies the number of input lines to be converted to uppercase. If omitted, 1 is assumed. If .US n is specified when .US ON is in effect, capitalization is turned off when n lines have been capitalized, or when .US OFF is encountered.

ON specifies that subsequent text lines are to be capitalized. OFF terminates capitalization mode if it was ON, or if n has been specified and has not been exhausted.

line is a line of text to be capitalized. The line is considered to start with the first nonblank character after the .US control word.

Next Script command is .US this is the input line.
Next Script command is .US this is the input line.
Next Script command is .US 2
one
two
Next Script command is .US on
.on
.on
.next line is .US off
..off
.next line is .US on
.on
.on 1
.on 2
.next line is .US 2 so only 2 more lines will be capitalized
..2
.on
two
three

**************************************************************************
//------------------------------
// [comments]

The .* control word allows comments to be stored in the Script file for future reference.

Next Script command is .* A comment line.
.* A comment line.
------------------------------*

Error Processing

The following invalid command is to illustrate the error handling in STR program.

Next command is .wrong command
.wrong command
------------------------------*/

C>TYPEF STR.CPL
&args filename
copyf %filename% t$.script
date
seg str
date
delete t$.script
cn t$.outscript %filename%.runoff

C>STR TEST
Saturday, May 14, 1983 12:49 AM
JOB STARTED
INCOUNT IS 644
OUTCOUNT IS 652
END OF JOB
Saturday, May 14, 1983 12:49 AM
C>TYPEF TEST.RUNOFF
This file is used to test the STR (Script To Runoff) program.
Its main use is to see if the Script commands are converted properly
to Runoff commands.
/**--------------------------------------------------------*/
.BM [ v | +v | -v ]

The .BM (Bottom Margin) control word specifies the amount of space to
be reserved at the bottom of the output pages, overriding the initial
value established for the device.

v specifies the amount of space to be reserved at the bottom of
output pages. If +v or -v is specified, the current value of the
bottom margin is incremented or decremented. If no value is specified
for v, the initial setting is restored.

.bm 5
Next amount should be 3
.BM  3
Next amount should be 4
.BM  4
.bm
/**--------------------------------------------------------*/
.BR

The .BR (Break) control word prevents the concatenation of the
following text line with preceding text.

.B
/**--------------------------------------------------------*/
.CD n

The .CD (Column Definition) control word defines how many columns of
output are to be formatted on each page.

n is the number of columns of output to be formatted onto each
subsequent output page. It may be any number from 1 to 9.

.C 3
.C 2
/**--------------------------------------------------------*/
The .CE (Center) control word centers text lines between the current left and right margins.

- **n** specifies the number of input lines to be centered. If omitted, 1 is assumed. If .CE n is specified when .CE ON is in effect, centering is turned off when n lines have been centered, or when .CE OFF is encountered.
- **ON** specifies that subsequent text lines are to be centered.
- **OFF** terminates centering mode if it was ON, or if n has been specified and has not been exhausted.
- **line** is a line of text to be centered. The line is considered to start with the first nonblank character after the .CE control word.

```plaintext
> This line is to be centered.
This line is not to be centered, but the next 3 lines are.
> One
> Two
> Three
Next line is to be centered.
> This line is centered.
The following several lines are centered until .CE OFF is encountered.
> center
> center
> center
> last in group to be centered
```

Next command is .CE 555 Main Street, program should center only this one line instead of the next 555 lines according to Script specification.

```plaintext
> 555 Main Street
This line is not to be centered.
```
The .CM (Comment) control word allows comments to be stored in the Script file for future reference.

Next Script command is .co on

Next Script command is .co off

The .DS (Double Space mode) causes subsequent output lines to be doublespaced.

Next Script command is .ds
The .FO (Format mode) controls concatenation and justification of input lines. 
ON restores default formatting, including both justification and concatenation of lines. If the .FO control word is used with no operands, ON is assumed. 
OFF cancels concatenation of input lines and justification of output lines. 
LEFT specifies that the input lines are to be concatenated but not justified. 

Next Script command is .fo on
.F
.AD
Next Script command is .fo off
.NF
.NA
Next Script command is .fo left
.F
.NA

The .IL (Indent Line) control word indents the next output line the specified amount of horizontal space. 
h specifies the amount of horizontal space to shift the next output line from the current margin. +h specifies that text is shifted to the right, and -h shifts text to the left. 
0 is default and initial setting value. 

Next Script command is .il
.U
Next Script command is .il 3
.P 3 0
Next Script command is .il -1
.U
Next Script command is .il +2
.P +2 0
SCRIPT TO RUNOFF CONVERSION PROGRAM

/*
  .IN [ 0 | h | +h | -h ]

The .IN (Indent) changes the left margin displacement of output.
  h specifies the amount of space to be indented. If omitted, 0 is
  assumed, and indentation reverts to the left margin. If +h or -h is
  used, the current left margin is incremented or decremented
  accordingly.

Next Script command is .in
  .U
Next Script command is .in 9
  .I 9
Next Script command is .in +2
  .I 11
Next Script command is .in -8
  .I 3

/*
  .IR [ 0 | h | +h | -h ]

The .IR (Indent Right) specifies the amount of space subsequent input
  lines are to be indented from the current right margin.
  h specifies the amount of space to be indented. If omitted, 0 is
  assumed, and indentation reverts to the right margin. If +h or -h is
  used, the current right margin is incremented or decremented
  accordingly.

Next Script command is .ir
  .RU
Next Script command is .ir 7
  .R 7
Next Script command is .ir -4
  .R 3
Next Script command is .ir +3
  .R 6

*/
/*-----------------------------------------------*/

.JU [ ON | OFF ]

The .JU (Justify Mode) causes left and right justification of output lines as needed to make the end of each line even with the current right margin.
ON restores right justification of output lines. If neither ON nor OFF is specified, ON is assumed.
OFF cancels justification of output lines. If concatenation is still in effect, .JU OFF results in ragged right output.

Next Script command is .ju
.AD
Next Script command is .ju off
.NA
Next Script command is .ju on
.AD

-----------------------------------------------*/
The .LI (Literal) control word allows all input lines, including those that begin with periods, to be processed as text.

.n specifies the number of input lines to be treated literally. If omitted, 1 is assumed. If .LI n is specified when .LI ON is in effect, literal mode is turned off after n lines or when .LI OFF is encountered.

.OFF specifies that subsequent text lines are to be treated literally. .OFF terminates literal mode if it was ON, or if n has been specified and has not been exhausted.

.lli is a line of text to be treated literally. The line is considered to start with the first nonblank character after the .LI control word.

+.This line is to be treated literally despite the period at the beginning.
The next 3 lines are treated literally.
+.One
+.Two
+.Three
Next line is to be treated literally.
+.This line is treated literally.
The following several lines are treated literally until .li off is encountered.
+.literal
+.literal
+.literal
+.literal
+.last in group to be treated literally
Next command is .li 555 Main Street, program should treat only this one line literally instead of the next 555 lines according to Script specification.
+.555 Main Street
>.This line is to be centered and not treated literally.
The .LL (Line Length) control word specifies the length of each subsequent output line.
h specifies an output line length not greater than the output device capability. If no value is specified for h, the default value established for the device being used will be taken. If +h or -h is used, the current line length is incremented or decremented accordingly.

Next Script command is .11 55.
.W 55
Next Script command is .11 -10.
.W 45
Next Script command is .11 +3.
.W 48

The .PA (Page Eject) control word causes a page eject, and can set the page number of the new page.
n specifies the page number of the next page. If n is not specified, sequential page numbering is assumed, and the next page number is one greater than the current page number. n must be an arabic number with no decimal point.

Next Script command is .pa.
.E
Next Script command is .pa 5.
.E
.PAG 5

------------------------------*/
The .PL (Page Length) control word specifies the amount of space, including top and bottom margins, for each output page. \( v \) specifies the vertical length, or depth, of output pages. If no value is specified for \( v \), the default value for the device will be used. If \(+v\) or \(-v\) is specified, the current page length is incremented or decremented accordingly.

Next Script command is .pl 55
.L 55
Next Script command is .pl -8
.L 47
Next Script command is .pl +2
.L 49

The .PP (Paragraph Start) control word begins formatting the output line as the start of a paragraph after a skip. \( line \) is the text that begins a new paragraph. If \( line \) is omitted, the text from the next input line after the .PP control word begins the new paragraph.

Next Script command is .pp This is the beginning of a paragraph.
.P 31
This is the beginning of a paragraph.
Next Script command is .pp followed by a separate input line.
.P 31
This is the separate input line.
The .QQ (Quick Quit) control word causes Script processing to terminate immediately.

Next Script command is .qq

The .SK (Skip Lines) control word specifies the amount of space to insert before the next text output line, except at the top of a column or page. v is the amount of space to be inserted in the output. If no number is given, 1 line is assumed.

Next Script command is .sk

Next Script command is .sk 4

--- --------------------------------------------------------------- */
The `.SP (Space Lines)` control word specifies the amount of space to insert before the next text output line. The specified number of lines are inserted even when the `.SP` occurs at the top of a page or column.

`v` is the amount of space to be inserted in the output. If no number is given, 1 line is assumed.

Next Script command is `.sp` 6
```
.BL ~
.B ~
.B ~
.B ~
.B ~
.B ~
.B ~
.B ~
```

Next Script command is `.sp` 2
```
.BL ~
.B ~
.B ~
.B ~
.B ~
.B ~
```

`---`
The .SS (Single-Spaced Mode) control word causes subsequent output lines to be single-spaced.

Next Script command is .ss
.SP 1

The .SX (Split Text) control word is used to split a string of text between the left and right column margins, with a filler in between the two.

C specifies that the middle part of the string is to be centered.
/ is any delimiter character. The first non blank character will be taken as the delimiter character.
lpart is the string to be placed against the current left margin.
fill is the string to be centered.
rpart is the string to be placed against the current right margin.

Next Script command is .sx c /left/middle/right/
./left/middle/right/
Next Script command is .sc c *leftleft*midmid*rightright*
./leftleft/midmid/rightright/

The .TB (Tab Setting) control word defines how tab characters (hexadecimal 05) are to be resolved.

SET specifies that all the old tab stops are to be cleared and a new set of tab stops is to be defined.
h specifies the horizontal displacements of the tab stops.

Next Script command is .tb set 5 20 30
.TA ' 5 20 30
The .TI control word translates the input text from one input representation to another.

s is the source character to be translated.

t is the desired output representation of the source character.

Currently, this command is processible only when defining tab character.

Next Script command is .ti @ 05 (to reset tab character to @)
Next Script command is .tb 10 20 30

The .TM (Top Margin) control word specifies the amount of space in the top margin area.

v specifies the amount of vertical space to be reserved at the top of the output pages. If no value is specified for v, the default value for the logical device will be used.

+v or -v increases or decreases the existing top margin by the amount given.

Next Script command is .tm 5

Next Script command is .tm -3

Next Script command is .tm 2

Next Script command is .tm +2

Next Script command is .tm 4
The `.UC` (Underscore and Capitalize) control word underscores and capitalizes one or more input lines. HOWEVER, BECAUSE OF RUNOFF RESTRICTIONS, THIS PROGRAM TREATS THE `.UC` COMMAND AS THE `.UP` (UPPERCASE) COMMAND.

`n` specifies the number of input lines to be converted to uppercase. If omitted, 1 is assumed. If `.UC n` is specified when `.UC ON` is in effect, capitalization is turned off when `n` lines have been capitalized, or when `.UC OFF` is encountered.

`.ON` specifies that subsequent text lines are to be capitalized.

`.OFF` terminates capitalization mode if it was `.ON`, or if `n` has been specified and has not been exhausted.

`.line` is a line of text to be capitalized. The line is considered to start with the first nonblank character after the `.UC` control word.

Next Script command is `.uc` this is the input line.
Next Script command is `.uc` 2
ONE
TWO
Next Script command is `.uc` on
ON
ON
NEXT LINE IS `.UC` OFF
next line is `.uc` on
ON 1
ON 2
NEXT LINE IS `.UC` 2 SO ONLY 2 MORE LINES WILL BE CAPITALIZED
ONE
TWO
three
The .UN (Undent) control word causes the next output line's indentation to change: it is moved to the left of the current margin. 
h specifies the amount of horizontal space by which the indentation is to be altered for the next output line only. If -h is specified, the UN control word is effectively the same as the .IL (Indent Line) control word. If omitted, 0 is assumed, and the indentation is not changed.

Next Script command is .in 9 .I 9
Next Script command is .un .U
This line is to be undented.
.I 9
Next Script command is .un -7 .I 2 Line to be undented.
.I 9
Next Script command is .un +3 .I 6 Line to be undented.
.I 9
Next Script command is .un -4 .I 13 Line to be undented.
.I 9
Next Script command is .un .U Line to be undented.
.I 9
The .UP (Uppercase) control word prints one or more subsequent input lines in UPPERCASE characters. 
n specifies the number of input lines to be converted to uppercase. 
If omitted, 1 is assumed. If .UP n is specified when .UP ON is in effect, capitalization is turned off when n lines have been capitalized, or when .UP OFF is encountered. 
ON specifies that subsequent text lines are to be capitalized. 
OFF terminates capitalization mode if it was ON, or if n has been specified and has not been exhausted. 
line is a line of text to be capitalized. The line is considered to start with the first nonblank character after the .UP control word.

Next Script command is .up this is the input line. 
THIS IS THE INPUT LINE. 
Next Script command is .up 2 
ONE 
TWO 
Next Script command is .up on 
ON 
ON 
ON 
NEXT LINE IS .UP OFF 
next line is .up on 
ON 1 
ON 2 
NEXT LINE IS .UP 2 SO ONLY 2 MORE LINES WILL BE CAPITALIZED 
ONE 
TWO 
three
SCRIPT TO RUNOFF CONVERSION PROGRAM

/*---------------------------------------------
 .US [ i | n | ON | OFF | line ]

The .US (Underscore) control word underscores one or more input lines.
HOWEVER, BECAUSE OF RUNOFF RESTRICTIONS, THIS PROGRAM TREATS THE .US
COMMAND AS THE .UP (UPPERCASE) COMMAND.

n specifies the number of input lines to be converted to uppercase.
If omitted, 1 is assumed. If .US n is specified when .US ON is in
effect, capitalization is turned off when n lines have been
capitalized, or when .US OFF is encountered.

ON specifies that subsequent text lines are to be capitalized.
OFF terminates capitalization mode if it was ON, or if n has been
specified and has not been exhausted.

line is a line of text to be capitalized. The line is considered to
start with the first nonblank character after the .US control word.

Next Script command is .us this is the input line.
THIS IS THE INPUT LINE.
Next Script command is .us 2
ONE
TWO
Next Script command is .us on
ON
ON
ON
NEXT LINE IS .US OFF
next line is .us on
ON 1
ON 2
NEXT LINE IS .US 2 SO ONLY 2 MORE LINES WILL BE CAPITALIZED
ONE
TWO
three
---------------------------------------------*/
/* [comments]  
The .* control word allows comments to be stored in the Script file for future reference.

Next Script command is .* A comment line.
.* A comment line.
---------------------------------------------------------------------*/

Error Processing

The following invalid command is to illustrate the error handling in STR program.

Next command is .wrong command  
.* THE FOLLOWING SCRIPT COMMAND CANNOT BE PROCESSED:
.* .wrong command
---------------------------------------------------------------------*/

C>como -end