Creating and Modifying Data

This section describes the functions that are available for all object types using main commands and/or line commands.

- Inserting and Deleting Lines
- Copying, Moving, Overlaying and Repeating Lines
- Copying or Moving a Window with Data
- Setting Horizontal and Vertical Boundaries
- Defining a Mask Line
- Ordering Data between Boundaries
- Centering Data
- Justifying Data
- Using the Physical or Logical Tabulator
- Sorting Lines in Alphabetical Order
- Finding a Character String
- Replacing a Character String

Inserting and Deleting Lines

This section describes the line commands and main commands available to insert and delete lines.

- Line Commands Available
- POWER Command
- DELETE Command

Line Commands Available

The following line commands are available to insert and delete lines on your editor screen.

Line Command	Explanation
D	Deletes this line.
D*n*	Deletes the next *n* lines.
DD	Marks the first line of a block to be deleted. A second DD line command is required to delineate the block. The deletion is performed after the second DD is entered.
DX	Deletes the line labeled .X.
DY	Deletes the line labeled .Y.
DX-Y	Deletes the block of lines from the line labeled .X to the line labeled .Y.
I	Inserts one line after this one. The editor switches to insert mode. This means if you type data or enter a blank in the new line and press ENTER, a new line is automatically inserted and the cursor placed in it. If you enter no new data in an inserted line and press ENTER, the editor leaves insert mode and the blank line is deleted. If you have defined a mask line and the MASK setting is ON, the mask line is inserted when you issue this command.
I*n*	Inserts *n* lines after this one. You can type data in the new lines. When you press ENTER, unused lines are deleted but one blank line remains with the cursor in it (editor stays in insert mode).
TE	Switches the editor to text enter mode. This means that beginning with this line the editor screen is blank (without line numbers) and you can enter data. When you press ENTER, any remaining blank lines are deleted, the line numbers are re-displayed and the text is reformatted within the set margins and with the specified justification. See also *POWER Command*.
W	Opens a window of one line. No new line is inserted if you enter data in the window line and press ENTER.
W*n*	Opens a window of *n* lines. When you press ENTER, all unused lines are deleted.

POWER Command

You can also use the POWER main command to switch to text enter mode.

When you issue the POWER main command, you are presented with a blank editor screen (without line numbers) and you can enter data. When you press ENTER, any remaining blank lines are deleted, the line numbers are redisplayed and the text is reformatted within the set margins and with the specified justification.

DELETE Command

Line deletion can also be performed with the DELETE main command.

For example, the command:

```
DEL C'Abc' .X .Y 10 30 ALL
```

deletes all lines containing the string Abc exactly as entered here between columns 10 to 30 within the block delineated by the labels .X and .Y.

You can specify all operands described for the FIND command above, except that the ALL direction operand specifies deletion of all lines with the given string. An unqualified DELETE command deletes the current line.

Copying, Moving, Overlaying and Repeating Lines

With the following line commands, you can copy, move or repeat lines or blocks of data.

Line Command	Explanation
A	Marks the target line for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. Data is inserted *after* this line.
B	Marks the target line for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. Data is inserted *before* this line.
C	Copies this line to the position marked by an A, B or O line command.
C*n*	Copies the next *n* lines to the position marked by an A, B or O line command.
CC	Marks the first line of the block to be copied. A second CC is required to the delineate block. Copying is performed after target has been marked by an A, B or O line command.
CX	Copies the line labeled .X. Inserts data after this line.
CY	Copies the line labeled .Y. Inserts data after this line.
CX-Y	Copies the block of lines from the line labeled .X to the line labeled .Y. Inserts data after this line.
M	Moves this line to the position marked by an A, B or O line command.
M*n*	Moves the next *n* lines to the position marked by an A or B line command.
MM	Marks the first line of the block to be moved. A second MM is required to delineate the block. The move is performed after the target has been marked by an A, B or O line command.
MX	Moves the line labeled .X. Inserts data after this line.
MY	Moves the line labeled .Y. Inserts data after this line.
MX-Y	Moves the block of lines from the line labeled .X to the line labeled .Y. Inserts data after this line.

Line Command	Explanation
O	Marks the target line for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. Data is merged with this line. (Only blank characters *within* boundaries are changed). See also *Example of Overlaying Data*.
O*n*	Marks this line and the next *n* lines as the target lines for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. The moved or copied lines are merged with these lines, that is, blank characters in the lines are overlaid. See also *Example of Overlaying Data*.
OO	Marks the first line of a block of target lines for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. A second OO command is required to mark the last line of the block of target lines. The moved or copied line(s) are merged with these lines, that is, blank characters in the lines are overlaid. See also *Example of Overlaying Data*.
R	Repeat this line once.
R*n*	Repeat this line *n* times.
RR	Marks the first line of a block to be repeated. A second RR is required to delineate the block. The repeat is performed after second RR is entered.
RR*n*	Repeat block delineated by two RR*n* line commands *n* times. The) (parenthesis) line commands (see below), move text the full number of columns specified, but only *within* the set boundaries, therefore, part of the moved text could disappear.
)	Moves this line right by one column beginning with left boundary.
)*n*	Moves this line right by *n* columns.
))*n*	Marks first line of a block to be moved right by *n* columns. A second))*n* is required to delineate the block. The move is performed after the second))*n* command is entered. Two unqualified)) line commands move the block right by 1 column.
(	Moves this line left by one column.
(*n*	Moves this line left by *n* columns.
((*n*	Marks first line of a block to be moved left by *n* columns. A second ((*n* is required to delineate the block. The move is performed after the second ((*n* command is entered. Two unqualified ((line commands move the block left by 1 column. If you use the > (greater than) or < (less than) symbols (see below) to move data, the maximum move possible is up to the next non-blank character *within* the set boundaries.

Line Command	Explanation
>	Moves this line right by one column.
>*n*	Moves this line right by *n* columns.
>>*n*	Marks first line of a block to be moved right by *n* columns. A second >> is required to delineate the block. The move is performed after the second >> command is entered. Two unqualified >> line commands move the block right by 1 column.
<	Moves this line left by one column.
<*n*	Moves this line left by *n* columns.
<<*n*	Marks first line of a block to be moved left by *n* columns. A second << is required to delineate the block. The move is performed after the second << command is entered. Two unqualified << line commands move the block left by 1 column.

Example of Overlaying Data

An overlay line command (O, O*n* or OO) allows you to merge single-column lists into multi-column format (that is, tabular form). You can use an overlay line command in conjunction with a copy (C, C*n* or CC) or move (M, M*n* or MM) line command.

The following two figures illustrate this function:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------------ Columns 001 072
 COMMAND===>                                                    SCROLL===> CSR
000090 //JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 // 'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
OO0140 //STEPLIB DD
000150 //        DD
000160 //        DD
OO0170 //        DD
MM0180                    DISP=SHR,DSN=SPF.SYSF.ADALOAD
000190                    DISP=SHR,DSN=SPF.SYSF.LOAD
000200                    DISP=SHR,DSN=SPF.SYSF.PROD.INST   * OPS INSTALL
MM0210                    DISP=SHR,DSN=SPF.SYSF.SOURCE      * OPS DOCUMENTS
000220 //DDCARD  DD *
000230 ADARUN DA=9,DE=3380,SVC=249
000240 //CMPRINT DD SYSOUT=X
000250 //CMPRT01 DD SYSOUT=X
000260 //CMWKF01 DD DUMMY
000270 //CMSYNIN DD *
000280 LOGON SYSMAIN2
000290 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000300 FIN
000310 /*
****** *************************** bottom of data ***************************




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

In the figure above, lines 180 to 210 are marked by the MM (Move) line command. They are to be overlaid on lines 140 to 170, which are marked by the OO (Overlay) line command.

This figure shows the result of the line commands displayed in the previous figure. Lines 180 to 210 have been overlaid on lines 140 to 170:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-76K ------------ Checkpoint done
 COMMAND===>                                                    SCROLL===> CSR
000090 //JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 // 'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
000140 //STEPLIB DD        DISP=SHR,DSN=SPF.SYSF.ADALOAD
000150 //        DD        DISP=SHR,DSN=SPF.SYSF.LOAD
000160 //        DD        DISP=SHR,DSN=SPF.SYSF.PROD.INST   * OPS INSTALL
000170 //        DD        DISP=SHR,DSN=SPF.SYSF.SOURCE      * OPS DOCUMENTS
000180 //DDCARD  DD *
000190 ADARUN DA=9,DE=3380,SVC=249
000200 //CMPRINT DD SYSOUT=X
000210 //CMPRT01 DD SYSOUT=X
000220 //CMWKF01 DD DUMMY
000230 //CMSYNIN DD *
000240 LOGON SYSMAIN2
000250 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000260 FIN
000270 /*
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

Copying or Moving a Window with Data

You can specify a window with data for move or copy operations. This allows you to copy or move data that does not start or end at the beginning or end of a line. This function can be performed by using editor line commands and/or main commands.

When you define a window, all data on your screen between start and end of the window becomes part of the window.

This section covers the following topics:

- Line Commands Available
- Main Commands Available

- Example of Using a Data Window

Line Commands Available

Line Command	Explanation
`WS`	Marks start of data window. Cursor position marks the column from which data is read. If the cursor is not in the line for which the command is entered, the window starts in column 1.
`WS`*n*	Data window starts in column *n* of this line.
`WE`	Marks end of data window. Works in the same way as `WS`. If the window is to start and end in the same line, type the `WS` command over the `WE` command. The editor acknowledges the set window with messages `WS` (`WS`*n*) and `WE` (`WE`*n*) in the prefix area, or with `WW` if the start and end of the window are in the same line. Before defining a new window, reset the old window with the `RESET` command to avoid a command conflict.
`WE`*n*	Data window ends in column *n* of this line.
`WC`	Copies the data window. The cursor position marks the column at which this line is to be split to insert the copied data.
`WC`*n*	Splits this line in column *n*, and copies the data between the two parts of the line.
`WM`	Moves the data window. Works in the same way as `WC`, but the original data is deleted after the copy operation.
`WM`*n*	Splits this line in column *n*, and moves the data between the two parts of the line.

Main Commands Available

Main Command	Explanation

`WINDOW`	Defines a window. The starting line and column, and the end line column are specified in the command parameters. At least one parameter is required. Examples:
	
	`WINDOW 5 10 24 13`
	
	Defines a window starting in line 5 / column 24, and ending in line 10, column 13.
	
	`WINDOW 5 10 24`
	
	Defines a window starting in line 5 / column 24, and ending in line 10 / last column.
	
	`WINDOW 5 10`
	
	Defines a window starting in line 5 / first column, and ending in line 10 / last column.
	
	`WINDOW 5 5`
	
	Defines a window starting in line 5 / first column, and ending in line 5 / last column.

`CWINDOW`	Copies a window defined with the `WINDOW` command. Optional parameters specify the line at which the window is to be inserted. Examples:
	
	`CWINDOW 5`
	
	Copies the window after line 5.
	
	`CWINDOW 5 24`
	
	Splits line 5 at column 24 and copies the window in between the two parts.

`DWINDOW`	Deletes a window of data defined by the `WINDOW` command.
`MWINDOW`	Moves window defined by the `WINDOW` command. Works like the `CWINDOW` command, but data in the original window is deleted after the copy operation.

Example of Using a Data Window

This section provides an example of defining and moving text with a data window by using either line commands or corresponding main commands.

The example refers to the text shown in Step 1 below and assumes that you want to move the whole sentence starting `Note that when...` (line 8) to follow the first sentence of the displayed text ending `...copy operations` (line 3).

To define and move a window using line commands

1. Type in text as shown below:

```
EDIT-NAT:NATLIB1(WINEX)-Text->Struct-Free-78K ----------------- Columns 001 072
 COMMAND===>                                                   SCROLL===> CSR
****** ***************************** top of data ****************************
000001 Copy a Window with Data
000002
000003 You can specify a window with data for move or copy operations. This
000004 allows you to copy or move data that does not start or end at the
000005 beginning or end of a line. This function can be performed using
000006 editor line commands and/or main commands.
000007
000008 Below are some examples of copying windows with data. Note that when
000009 you define a window, all data on your screen between start and end of
000010 the window become part of the window. Available line commands are:
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

2. Type the line command `WS` in line 8, the first line of data to be moved, place the cursor in the required column (`N` of the word `Note`) and press ENTER.

   The message `WS55` appears in the prefix area of line 8, indicating the column number selected:

```
EDIT-NAT:NATLIB1(WINEX)-Text->Struct-Free-78K ---------------- Block is pending
 COMMAND===>                                                    SCROLL===> CSR
****** ***************************** top of data ****************************
000001 Copy a Window with Data
000002
000003 You can specify a window with data for move or copy operations. This
000004 allows you to copy or move data that does not start or end at the
000005 beginning or end of a line. This function can be performed using
000006 editor line commands and/or main commands.
000007
WS55   Below are some examples of copying windows with data. Note that when
000009 you define a window, all data on your screen between start and end of
000010 the window become part of the window. Available line commands are:
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

3. Type the line command `WE` in line 10, the last line of data to be moved, move the cursor to the last column to be moved (full stop (`.`) after `window`) and press ENTER.

   The message `WE37` appears in the prefix area of line 10:

```
EDIT-NAT:NATLIB1(WINEX)-Text->Struct-Free-78K ---------------- Block is pending
 COMMAND===>                                                    SCROLL===> CSR
****** ***************************** top of data ****************************
000001 Copy a Window with Data
000002
000003 You can specify a window with data for move or copy operations. This
000004 allows you to copy or move data that does not start or end at the
000005 beginning or end of a line. This function can be performed using
000006 editor line commands and/or main commands.
000007
WS55   Below are some examples of copying windows with data. Note that when
000009 you define a window, all data on your screen between start and end of
WE37   the window become part of the window. Available line commands are:
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

4. Type the line command `WM` in line 3 (the data will be moved to the following line, line 4), and move the cursor to the column at which line 3 is to be split (the blank before the word `This`). Press ENTER.

   The specified text section is moved:

```
EDIT-NAT:NATLIB1(WINEX)-Text->Struct-Free-78K ---------------- Checkpoint done
 COMMAND===>                                                   SCROLL===> CSR
****** ***************************** top of data ****************************
000001 Copy a Window with Data
000002
000003 You can specify a window with data for move or copy operations.
000004 Note that when
000005 you define a window, all data on your screen between start and end of
000006 the window become part of the window.
000007 This
000008 allows you to copy or move data that does not start or end at the
000009 beginning or end of a line. This function can be performed using
000010 editor line commands and/or main commands.
000011
000012 Below are some examples of copying windows with data.
000013                                        Available line commands are:
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

You can achieve the same result described in the instructions above when using the command sequence indicated below.

To define and move a window using main commands

1. Type in the text shown in Step 1 above.
2. In the command line, enter the following:

```
WINDOW 8 10 55 37;MWINDOW 3 64
```

The specified text section is moved as illustrated in Step 4 above.

Setting Horizontal and Vertical Boundaries

The editor provides commands which allow you to set horizontal and vertical boundaries within which certain functions can be performed such as the main commands `FIND`, `CHANGE`, `CENTER`, `ORDER`, `JLEFT` and `JRIGHT`, as well as their corresponding line commands (for example, `TC`, `TO`, `LJ` or `RJ`).

This section covers the following topics:

- Setting Boundaries
- Labeling Single or Multiple Lines

Setting Boundaries

With the BNDS main command, you can define horizontal boundaries as described in the following example instructions.

To set and display boundaries

1. Issue the following main command:

   ```
   BNDS 20 50
   ```

   Horizontal limits are set at columns 20 and 50.

2. To display the current boundary settings, issue the following line command:

   ```
   BNDS
   ```

   The following figure shows the result of the BNDS 20 50 main command followed by a BNDS line command in line 2:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------------ Columns 001 072
 COMMAND===>                                                   SCROLL===> CSR
****** ***************************** top of data ****************************
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
=bnds>                    <                              >
000010   RESET #JOBNAME(A8)
000020   RESET #FD(N3) #FL(A8) #FF(N3)
000030   RESET #TD(N3) #TL(A8) #TF(N3)
000040   COMPRESS *INIT-USER 'SM' INTO #JOBNAME LEAVING NO SPACE
000050   SET CONTROL 'WL60C6B005/010F'
000060   INPUT  'ENTER PARAMETERS FOR LIBRARY COPY:'
000070    /     'FROM:   DBID:' #FD 'FNR:' #FF 'LIB:' #FL
000080    /     'TO  :   DBID:' #TD 'FNR:' #TF 'LIB:' #TL
000090 // #JOBNAME JOB JWO,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT21,REGION=2000K,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
000140 //STEPLIB  DD  DISP=SHR,DSN=OPS.SYSF.V5.ADALOAD
000150 //         DD  DISP=SHR,DSN=OPS.SYSF.PROD.LOAD
000160 //DDCARD   DD  *
000170 ADARUN DA=9,DE=3380,SVC=249
000180 //CMPRINT  DD  SYSOUT=X
000190 //CMPRT01 DD SYSOUT=X
000200 //CMWKF01 DD DUMMY
000210 //CMSYNIN DD *
000220 LOGON SYSMAIN2
000230 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000240 FIN
000250 /*
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

Labeling Single or Multiple Lines

You can label the current line (line currently at the top of editing area) or a block of lines by using either the LABEL main commands or corresponding line commands.

To label current lines using LABEL

1. Issue the following main command:

   ```
   LABEL  .X
   ```

   The current line is labeled .X.

2. To delineate a block of lines, you can now scroll to the next required delimiting line and issue the following main command:

   ```
   LABEL .Y
   ```

The new current line is labeled `.Y` denoting the last line of a block of lines.

To label specific lines using line commands

1. Next to the first line to be labeled, issue the following line command:

   ```
   .X
   ```

   The specified line is labeled `.X`.

2. Next to the last line to be labeled, issue the following line command:

   ```
   .Y
   ```

   The specified line is labeled `.Y` delimiting a block of lines that starts with the line labeled `.X`.

Tip:
You can use any string to label lines, for example `.START` and `.END`.

For examples of using labeled lines, see the sections *Finding a Character String* and *Replacing a Character String*.

Defining a Mask Line

You can define data that is automatically placed in a line added through a line insertion operation (for example, by using the line command `I` or `W`). Such a line is referred to as a mask line. A mask line is useful when you must write several lines of code which are identical or very similar.

Note:
You can only have one mask line during an editing session. If you define a new mask line, any existing mask line definition is automatically updated with the new value.

To define and use a mask line

1. Issue the following line command:

   ```
   MASK
   ```

   A blank line indicated by `=mask>` appears above the line in which you entered the command.

2. In the blank line, enter the data you want to define as a mask line and press ENTER.

   The mask line is now available for the current source until you update the mask with a new mask line or until you deactivate the mask function.

3. Issue the following main command:

```
MASK ON
```

   The mask function is activated. The defined mask line now appears in all lines added through a line insert operation.

4. Issue an insert line command, for example:

```
I2
```

   Two new lines are inserted into the source with the text of the mask line. The text of a mask line appears in all lines added with an insert command.

5. Modify the text in the new lines. If you do not modify the text, any inserted line is deleted the next time you press ENTER.

6. If required, deactivate the mask function with the following main command:

```
MASK OFF
```

   The MASK OFF command deactivates the mask function but does not delete the contents of the mask line.

See also MASK in *Summary of Main Commands*.

Ordering Data between Boundaries

You can change the indentation of specified lines by using the ORDER main command together with a boundary setting. For example, the command sequence:

```
BNDS 3;ORDER 5 20
```

moves lines 5 to 20 right to start in column 3.

Note:
If the end of any ordered line traverses the right boundary, it is automatically split.

To re-justify the shifted data to the left, use a JLEFT command.

You can also change the indentation of lines or of a block of data by using line commands. Here, too, if the end of any line traverses the right boundary, it is automatically split.

Line Command	Explanation
TF	Orders data from the line on which it was entered to the end of the paragraph or to the next blank line with the right boundary. This line command can be entered with a numerical value specifying the right boundary, for example, the line command TF50 orders data with column 50.
TO	Marks one line to be ordered.
TOO	Marks the first line of a block of data to be ordered. Requires a second TOO line command to delineate the block. Ordering is performed after the second TOO is issued.

Data can be ordered within set boundaries and justified to the left boundary, right boundary or both by using the JUSTIFY command. For example, the command:

```
BNDS 6 60;JUSTIFY BOTH
```

activates justification to columns 5 and 60. To perform the ordering, mark a block of data with two TOO line commands.

The editor also provides a line command with which you can split a single line into two. Type the line command S in the prefix area of the line you wish to split, move the cursor to the position where the split is to occur and press ENTER.

Centering Data

The editor also provides commands with which you can center specified data within set boundaries. For example, the sequence:

```
BNDS 5 60;CENTER 5 15
```

centers data in lines 5 to 15 between columns 5 and 60.

Note:
Only text already within the boundaries is centered. Text to the left and right of the boundaries is not affected.

Alternatively, you can use line commands to perform the centering function:

Line Command	Explanation
TC	Centers this line within the set boundaries.
TCC	Marks the first line in a block of data to be centered. Requires a second TCC line command to delineate the block. The centering is performed after the second TCC command is issued.

Justifying Data

A number of main commands and line commands are available to rearrange lines or blocks of data on your screen, depending on the setting of your horizontal boundaries (BNDS main command); see the section *Setting Horizontal and Vertical Boundaries*.

The JLEFT and JRIGHT main commands justifies the specified data with the left and right boundaries respectively. For example, the sequence:

```
BNDS 16 80;JLEFT 140 170
```

justifies the data between columns 16 to 80 in lines 140 to 170 with column 16.

The figure below illustrates this example:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------------ Columns 001 072
 COMMAND===> BNDS 16 80;JLEFT 140 170                           SCROLL===> CSR
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000090 // #JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
000140 //STEPLIB  DD           DISP=SHR,DSN=SPF.SYSF.ADALOAD
000150 //         DD           DISP=SHR,DSN=SPF.SYSF.LOAD
000160 //         DD           DISP=SHR,DSN=SPF.SYSF.PROD.INST  * OPS INSTALL
000170 //         DD           DISP=SHR,DSN=SPF.SYSF.SOURCE     * OPS DOCUMENTS
000180 //DDCARD   DD  *
000190 ADARUN DA=9,DE=3380,SVC=249
000200 //CMPRINT  DD  SYSOUT=X
000210 //CMPRT01 DD SYSOUT=X
000220 //CMWKF01 DD DUMMY
000230 //CMSYNIN DD *
000240 LOGON SYSMAIN2
000250 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000260 FIN
000270 /*
****** *************************** bottom of data ***************************




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

The following screen shows the result of the command displayed in the command line of the previous screen:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------ File has been ordered
 COMMAND===>                                                  SCROLL===> CSR
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000090 // #JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
000140 //STEPLIB  DD  DISP=SHR,DSN=SPF.SYSF.ADALOAD
000150 //         DD  DISP=SHR,DSN=SPF.SYSF.LOAD
000160 //         DD  DISP=SHR,DSN=SPF.SYSF.PROD.INST  * OPS INSTALL
000170 //         DD  DISP=SHR,DSN=SPF.SYSF.SOURCE     * OPS DOCUMENTS
000180 //DDCARD   DD  *
000190 ADARUN DA=9,DE=3380,SVC=249
000200 //CMPRINT  DD  SYSOUT=X
000210 //CMPRT01 DD SYSOUT=X
000220 //CMWKF01 DD DUMMY
000230 //CMSYNIN DD *
000240 LOGON SYSMAIN2
000250 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000260 FIN
000270 /*
****** *************************** bottom of data ***************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

The sequence:

```
BNDS 10;JRIGHT 15
```

justifies the data on the right of column 10 in line 15 to the end of the screen with the last column of your editor screen (column 88 of your terminal screen).

Alternatively, you can justify lines or blocks of data by using any of the line commands listed in the following table:

Line Command	Explanation
LJ	Justifies the data within the set boundaries in this line with the left boundary.
LJJ	Marks the first line of a block of data within the boundaries (set with the BNDS main command) to be justified to the left. A second LJJ line command is required to delineate the block. Justification is performed after the second LJJ is issued.
RJ	Justifies the data within the set boundaries in this line with the right boundary.
RJJ	Marks the first line of a block of data within the boundaries (set with the BNDS main command) to be justified to the right. A second RJJ line command is required to delineate the block. Justification is performed after the second RJJ is issued.

You can also justify data to the left boundary, or order data between left and right boundary in conjunction with the JUSTIFY command.

For example, the command sequence:

```
BNDS 10 60;JUSTIFY LEFT
```

enables justification to the left boundary. Mark a block of data with two TOO line commands (explained below) to reformat the data between columns 10 and 60, justified to column 10.

Using the Physical or Logical Tabulator

This section provides instructions for using the TABS main command and/or the TABS line command to control tabulator settings.

In the examples of tabulation provided in this section, the ampersand (&) is assumed to be the tabulation character; the COLS line command has been issued to display column positions.

This section covers the following topics:

- Using the TABS Main Command
- Using the TABS Line Command
- Example 1 - Tab Positions
- Example 2 - TABS RIGHT
- Example 3 - TABS DECIMAL
- Example 4 - Mixed Justification
- Example 5 - Using a Blank as Tabulation Symbol

Using the TABS Main Command

When you issue the `TABS ON` main command, the standard tab positions set in your editor profile are turned on and `tabs on std` appears in your profile. Issue the `TABS OFF` main command to turn tabulation off again.

For detailed information on the command syntax that applies to `TABS`, see the relevant section in *Summary of Main Commands*.

This section covers the following topics:

- Setting Standard Tab Positions
- Setting the Logical Tab Character
- Setting Justification Parameters

Setting Standard Tab Positions

To turn tabulation on and set the standard tab positions for your profile to columns 10, 20, 30, 40 and 50, for example, issue the main command `TABS 10 20 30 40 50`.

Setting the Logical Tab Character

To turn tabulation on and set the logical tab character to & (ampersand), for example, issue the main command `TABS &`.

You can enter data and automatically move it to a specific tab position by preceding it with a logical tab character. One tab character moves the data to the next tab position, two tab characters moves the data to the second tab position, and so on.

Setting Justification Parameters

Apart from tab positions, you can specify the following parameters with the `TABS` main command:

Parameter	Explanation
`DECIMAL`	Orders data with the decimal point in the data at the tab position. See also *Example 3 - TABS DECIMAL*.
`LEFT`	Orders data to the left of the tab position. See also *Example 1 - Tab Positions*.
`RIGHT`	Orders data to the right of the tab position. See also *Example 2 - TABS RIGHT*.

To display the current logical tab character and shift parameter (excluding tab positions), issue the `PROFILE` main command.

Using the TABS Line Command

When you issue the TABS line command in any line, the current tab positions set in your editor profile are displayed in that line and marked with asterisks (*) if no logical tab character has been set. This command does not turn tabulation on.

For example, issue the TABS line command to display the positions set with the main command TABS `10 20 30 40 50`.

This displays the current tab positions as follows:

```
=tabs>            *       *       *       *       *       *
```

Setting Multiple Logical Tab Characters and Mixed Justification Parameters

To tabulate data in a specific column and with a specific shift, multiple logical tab characters and mixed justification parameters are possible.

To set the multiple logical tab characters, issue the TABS line command and type a special character over each asterisk (*). Any data typed in preceded by any of these logical tab characters are tabulated in the corresponding column.

To set the mixed justification parameters, type L (Left), R (Right) or D (Decimal) to the right of each logical tab character for left, right or decimal ordering.

See also *Example 4 - Mixed Justification*.

Example 1 - Tab Positions

The command

```
TABS 10 20 40 LEFT
```

activates logical tabs with tabulation columns 10, 20, and 40 with left justification. After you press ENTER, the input text line

```
&abc &def &ghi
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
               abc       def               ghi
```

Example 2 - TABS RIGHT

The command

```
TABS RIGHT
```

activates logical tabs with right justification. After you press ENTER, the input text line

```
&abc &def &ghi
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
             abc       def                ghi
```

Example 3 - TABS DECIMAL

The command

```
TABS DECIMAL
```

activates logical tabs with justification of the decimal point in the tab position. After you press ENTER, the input text line

```
&15.27$ &16.3 EUR &13 IS
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
             15.27$    16.3 EUR             13 IS
```

Example 4 - Mixed Justification

Issue the command:

```
TABS 10 20 30 40 50
```

Then issue the TABS line command. This displays the current tab positions as follows:

```
=tabs>          *         *         *         *         *
```

Type an L, R or D next to each tab position as required (unmarked tab positions assume the value of the last TAB command):

```
=tabs>          *R        *D        *D        *D        *L
```

After you press ENTER, the input text line

```
&start &0.01 &0.02 &0.03 &end
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
            start        0.01      0.02      0.03       end
```

Example 5 - Using a Blank as Tabulation Symbol

Issue the command

```
TABS ' '
```

which activates tabulation with one blank as tabulation character. This means that words separated by one blank are tabulated. After you press ENTER, the input text line

```
this is a blank tabulation
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
               this      is        a         blank     tabulation
```

Sorting Lines in Alphabetical Order

You can sort data lines in ascending or descending alphabetical order according to sorting criteria. For example, the command:

```
SORT 10 15
```

sorts all lines in the source in ascending order according to the characters beginning in column 10 and ending in column 15.

To sort only a block of lines, for example, label the lines where the block is to start and end with `.X` and `.Y` respectively. The command:

```
SORT .X .Y D
```

sorts all lines in the block marked with `.X` and `.Y` in descending order.

To sort a block of lines according to the characters beginning in column 5 and ending in column 20, for example, label the lines where the block is to start and end with `.X` and `.Y` respectively. The command:

```
SORT 5 20 .X .Y
```

sorts all lines in the block marked by `.X` and `.Y` in ascending order according to the characters beginning in column 5 and ending in column 20.

Finding a Character String

To locate a specific character string, you can use the `FIND` main command with operands defining the string, the area to be searched and the direction of search. The cursor is placed on the first character of the string. If the line containing the string was excluded from display, it is now included in the display.

The following sections describe the possible command operands.

- String Definition Operand
- String-Matching Operand
- Direction Operand
- Line-Type Operand
- Block Operand
- Columns Operand
- Examples of the FIND Command

String Definition Operand

The string operand defines the character string to be located.

You can specify any of the following:

Operand	Explanation
*	Finds the string specified in previous FIND command.
'abc'	Finds the string abc regardless of whether the string is upper case or lower case.
C'Abc'	Finds the string exactly as entered here.
P'a*(char)*c'	Finds the string whose first character is a and third character is c. *(char)* stands for a special character acting as a wildcard character with the following meaning: = any character § alphabetic character # numeric character $ special character ^ non-blank character - non-numeric character < lower-case character > upper-case character
T'abc'	Finds the string abc regardless of whether the string is upper case or lower case.
X'D4A8'	Finds the string that corresponds to hexadecimal D4A8.

String-Matching Operand

The string-matching operand specifies whether any special occurrence of the string is to be located. The following options are possible:

Operand	Explanation
CHARS	No restrictions (any occurrence of the string).
PREFIX	Only those occurrences which are the prefix of a word.
SUFFIX	Only those occurrences which are the suffix of a word.
WORD	Only those occurrences which form a word.

Default is CHARS.

Direction Operand

The direction operand specifies the direction of the search operation.

The following options are possible:

Operand	Explanation
ALL	Any occurrence of the string (search all directions).
FIRST	First occurrence of the string.
LAST	Last occurrence of the string.
NEXT	Next occurrence of the string starting from the cursor position.
PREV	Previous occurrence of the string.

Default is NEXT.

Line-Type Operand

This line-type operand specifies whether excluded or included lines only are to be searched. The following options are possible:

Operand	Explanation
X	Search excluded lines only.
NX	Search non-excluded lines only.

If this operand is omitted, the editor searches all data for the given string, included and excluded lines. If the string is found in an excluded line, it is returned to display.

Block Operand

If you have labeled lines or a block of lines, you can use the block operand to restrict the search area for the FIND command.

Two examples of the block operand follow:

Operand	Explanation
.X	Search from line labeled .X to end of data.
.X .Y	Search from line labeled .X to line labeled .Y.

where X and Y can be any alphabetic character or four-character string.

Columns Operand

The column operand allows you to restrict the search for the given string between certain columns. Below are two examples of the columns operand.

Operand	Explanation
20	Locate given string starting in column 20 (the first character of the string must be in column 20).
20 40	Locate given string anywhere between columns 20 to 40.

Examples of the FIND Command

```
F C'HILITE' X PREV
```

Find the previous occurrence of the string HILITE exactly as entered here; search excluded lines only.

```
F P'RCV#' .X .Z 20 30
```

Find the string starting RCV with a numeric fourth character within the block .X .Z and between columns 20 to 30.

```
F X'6C' SUFFIX NX
```

Find the character corresponding to the hexadecimal 6C in non-excluded lines only. The character must end a word.

Command	Explanation
F C'HILITE' X PREV	Find the previous occurrence of the string HILITE exactly as entered here; search excluded lines only.
F P'RCV#' .X .Z 20 30	Find the string starting RCV with a numeric fourth character within the block .X .Z and between columns 20 to 30.
F X'6C' SUFFIX NX	Find the character corresponding to the hexadecimal 6C in non-excluded lines only. The character must end a word.

If single quotation marks are part of the string to be found, you must use a different separator in the FIND command, for example double quotation marks:

```
FIND C"'string'"
```

You can repeat a previous FIND command with the RFIND main command.

Replacing a Character String

You can find and replace a given character string by another character string by using the CHANGE main command.

If apostrophes are part of the string to be replaced, you must use a different separator in the CHANGE command, for example quotation marks:

```
CHANGE "'string1'" "'string2'"
```

The same operands as for the FIND command can be used with the CHANGE command. For the CHANGE command, however, the ALL directions operand means change all occurrences of the specified string.

After the replace operation is performed, the message ==chg> appears in the prefix area of the changed line.

This section covers the following topics:

- Examples of the CHANGE Command
- RFIND and RCHANGE Commands

Examples of the CHANGE Command

Command	Explanation
CHG 'LOW' 'HIGH'	Replaces the first occurrence of LOW by HIGH (upper and lower case ignored).
CHG C'OPS' 'SPF' .X .Y 28 32 all	Replaces OPS (exactly as entered here) by SPF; replace all occurrences in the block labeled .X and .Y and between the columns 28 and 32. See also the example screens below.
Repeated CHANGE commands:	
CHG * 'NEW'	Replaces the next occurrence of the string specified in the last CHANGE command by the new string NEW.
CHG 'OLD' *	Replaces OLD by the same new string as specified in the last CHANGE command.

The screen below illustrates the second example *before* the command is executed with ENTER:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------------ Columns 001 072
 COMMAND===> CHG C'OPS' 'SPF' .X .Y 28 32 ALL                   SCROLL===> CSR
****** ***************************** top of data ****************************
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000010   RESET #JOBNAME(A8)
000020   RESET #FD(N3) #FL(A8) #FF(N3)
000030   RESET #TD(N3) #TL(A8) #TF(N3)
000040   COMPRESS *INIT-USER 'SM' INTO #JOBNAME LEAVING NO SPACE
000050   SET CONTROL 'WL60C6B005/010F'
000060   INPUT  'ENTER PARAMETERS FOR LIBRARY COPY:'
000070    /     'FROM:   DBID:' #FD 'FNR:' #FF 'LIB:' #FL
000080    /     'TO  :   DBID:' #TD 'FNR:' #TF 'LIB:' #TL
000090 // #JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
.X     //STEPLIB  DD  DISP=SHR,DSN=OPS.SYSF.ADALOAD
000150 //         DD  DISP=SHR,DSN=OPS.SYSF.LOAD
000160 //         DD  DISP=SHR,DSN=OPS.SYSF.PROD.INST  * OPS INSTALL
.Y     //         DD  DISP=SHR,DSN=OPS.SYSF.SOURCE     * OPS DOCUMENTS
000180 //DDCARD   DD  *
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

The screen below illustrates the second example *after* the command is executed with ENTER:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------ 4 char  'OPS' changed
 COMMAND===>                                                   SCROLL===> CSR
****** ***************************** top of data ****************************
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000010   RESET #JOBNAME(A8)
000020   RESET #FD(N3) #FL(A8) #FF(N3)
000030   RESET #TD(N3) #TL(A8) #TF(N3)
000040   COMPRESS *INIT-USER 'SM' INTO #JOBNAME LEAVING NO SPACE
000050   SET CONTROL 'WL60C6B005/010F'
000060   INPUT  'ENTER PARAMETERS FOR LIBRARY COPY:'
000070    /     'FROM:   DBID:' #FD 'FNR:' #FF 'LIB:' #FL
000080    /     'TO  :   DBID:' #TD 'FNR:' #TF 'LIB:' #TL
000090 // #JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
.X     //STEPLIB  DD  DISP=SHR,DSN=SPF.SYSF.ADALOAD
==chg> //         DD  DISP=SHR,DSN=SPF.SYSF.LOAD
==chg> //         DD  DISP=SHR,DSN=SPF.SYSF.PROD.INST  * OPS INSTALL
.Y     //         DD  DISP=SHR,DSN=SPF.SYSF.SOURCE     * OPS DOCUMENTS
000180 //DDCARD   DD  *
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

All occurrences of the string OPS have been replaced by the string SPF between lines 140 and 170 and within the columns 28 to 32.

RFIND and RCHANGE Commands

When changing character strings, good use can be made of the RFIND (repeated FIND) and the RCHANGE (repeated CHANGE) commands, for example, the sequence:

```
FIND 'abc'
  CHANGE 'abc' 'def'
  RFIND
  RCHANGE
```

allows you to find occurrences of a certain string and optionally change them with relatively small effort.

These commands can also be assigned to PF keys.