

*TRIM - Remove Leading and/or Trailing Blanks

$*TRIM (operand \left[, \left\{ \begin{array}{l} LEADING \\ TRAILING \end{array} \right\} \right])$

Format/length: same as *operand* (A, U or B)/DYNAMIC.

This chapter covers the following topics:

- Function
 - Restrictions
 - Syntax Description
 - Examples
-

Function

The Natural system function *TRIM removes all leading and/or trailing blanks from an alphanumeric or a binary string. The content of the operand is not modified. When using a dynamic variable as operand, the length of this variable is adapted according to the result.

The *TRIM system function may be specified as an operand in any position of a statement wherever an operand of format A, U or B is allowed.

Restrictions

When using the system function *TRIM, the following restrictions apply:

- *TRIM must not be used where a target variable is expected.
- You may not nest *TRIM in a system function.
- If the operand is a static variable, it is not possible to remove trailing blanks using *TRIM, because for static variables the remaining trailing positions of the variable memory are filled with space characters.

Syntax Description

Operand Definition Table:

Operand	Possible Structure			Possible Formats													Referencing Permitted	Dynamic Definition			
<i>operand</i>	C	S	A			A	U	B												yes	no

Syntax Element Description:

*TRIM(<i>operand</i> , LEADING)	Remove Leading Blanks When the keyword LEADING is used as a second argument, all leading blanks are removed from the string contained in <i>operand</i> .
*TRIM(<i>operand</i> , TRAILING)	Remove Trailing Blanks When the keyword TRAILING is used as a second argument, all trailing blanks are removed from the string contained in <i>operand</i> .
*TRIM(<i>operand</i>)	Remove Both Leading and Trailing Blanks When no keyword is used as a second argument, both the leading and the trailing blanks are removed from the string contained in <i>operand</i> .

Examples

The following examples are provided below:

- Example 1 - Using an Alphanumeric Argument
- Example 2 - Using a Binary Argument

Example 1 - Using an Alphanumeric Argument

```

DEFINE DATA LOCAL
/*****
/* STATIC VARIABLE DEFINITIONS
/*****
1 #SRC (A15) INIT <' ab CD '>
1 #DEST (A15)

/* FOR PRINT OUT WITH DELIMITERS
1 #SRC-PRN (A20)
1 #DEST-PRN (A20)

/*****
/* DYNAMIC VARIABLE DEFINITIONS
/*****
1 #DYN-SRC (A)DYNAMIC INIT <' ab CD '>
1 #DYN-DEST (A)DYNAMIC

/* FOR PRINT OUT WITH DELIMITERS
1 #DYN-SRC-PRN (A)DYNAMIC
1 #DYN-DEST-PRN (A)DYNAMIC

END-DEFINE

PRINT 'static variable definition:'
PRINT '-----'

```

```

COMPRESS FULL ':' #SRC ':' TO #SRC-PRN LEAVING NO SPACE
PRINT ' '
PRINT ' 123456789012345          123456789012345'

MOVE *TRIM(#SRC, LEADING) TO #DEST
COMPRESS FULL ':' #DEST ':' TO #DEST-PRN LEAVING NO SPACE
DISPLAY #SRC-PRN #DEST-PRN '*TRIM(#SRC, LEADING)'
```

```

MOVE *TRIM(#SRC, TRAILING) TO #DEST
COMPRESS FULL ':' #DEST ':' TO #DEST-PRN LEAVING NO SPACE
DISPLAY #SRC-PRN #DEST-PRN '*TRIM(#SRC, TRAILING)'
```

```

MOVE *TRIM(#SRC) TO #DEST
COMPRESS FULL ':' #DEST ':' TO #DEST-PRN LEAVING NO SPACE
DISPLAY #SRC-PRN #DEST-PRN '*TRIM(#SRC)'
```

```

PRINT ' '
PRINT 'dynamic variable definition:'
PRINT '-----'
COMPRESS FULL ':' #DYN-SRC ':' TO #DYN-SRC-PRN LEAVING NO SPACE
PRINT ' '
PRINT ' 1234567890          12345678'
```

```

MOVE *TRIM(#DYN-SRC, LEADING) TO #DYN-DEST
COMPRESS FULL ':' #DYN-DEST ':' TO #DYN-DEST-PRN LEAVING NO SPACE
DISPLAY (AL=20) #DYN-SRC-PRN #DYN-DEST-PRN '*TRIM(#SRC, LEADING)'
```

```

MOVE *TRIM(#DYN-SRC, TRAILING) TO #DYN-DEST
COMPRESS FULL ':' #DYN-DEST ':' TO #DYN-DEST-PRN LEAVING NO SPACE
DISPLAY (AL=20) #DYN-SRC-PRN #DYN-DEST-PRN '*TRIM(#SRC, TRAILING)'
```

```

MOVE *TRIM(#DYN-SRC) TO #DYN-DEST
COMPRESS FULL ':' #DYN-DEST ':' TO #DYN-DEST-PRN LEAVING NO SPACE
DISPLAY (AL=20) #DYN-SRC-PRN #DYN-DEST-PRN '*TRIM(#SRC)'
```

```

PRINT ' '
PRINT '":' := delimiter character to show the start and ending of a string!'
END
```

Output of Example 1:

```

#SRC-PRN          #DEST-PRN
-----

static variable definition:
-----

123456789012345          123456789012345
: ab CD :                :ab CD :                *TRIM(#SRC, LEADING)
: ab CD :                : ab CD :                *TRIM(#SRC, TRAILING)
: ab CD :                :ab CD :                *TRIM(#SRC)
```

```

dynamic variable definition:
-----

1234567890          12345678
: ab CD :                :ab CD :                *TRIM(#SRC, LEADING)
: ab CD :                : ab CD:                *TRIM(#SRC, TRAILING)
: ab CD :                :ab CD:                *TRIM(#SRC)
```

```

':' := delimiter character to show the start and ending of a string!
```

Example 2 - Using a Binary Argument

```

DEFINE DATA LOCAL
/*****
/* STATIC VARIABLE DEFINITIONS
/*****
1 #SRC (B10) INIT <H'2020FFFF2020FFFF2020'>
1 #DEST (B10)

/*****
/* DYNAMIC VARIABLE DEFINITIONS
/*****
1 #DYN-SRC (B)DYNAMIC INIT <H'2020FFFF2020FFFF2020'>
1 #DYN-DEST (B)DYNAMIC
END-DEFINE

FORMAT LS=100

PRINT 'static variable definition:
PRINT '-----'
MOVE *TRIM(#SRC, LEADING) TO #DEST
PRINT #SRC #DEST '*TRIM(#SRC, LEADING)'

MOVE *TRIM(#SRC, TRAILING) TO #DEST
PRINT #SRC #DEST '*TRIM(#SRC, TRAILING)'

MOVE *TRIM(#SRC) TO #DEST
PRINT #SRC #DEST '*TRIM(#SRC)''

PRINT ' '
PRINT 'dynamic variable definition:'
PRINT '-----'

MOVE *TRIM(#DYN-SRC, LEADING) TO #DYN-DEST
PRINT #DYN-SRC #DYN-DEST ' *TRIM(#SRC, LEADING)''

MOVE *TRIM(#DYN-SRC, TRAILING) TO #DYN-DEST
PRINT #DYN-SRC #DYN-DEST ' *TRIM(#SRC, TRAILING)''

MOVE *TRIM(#DYN-SRC) TO #DYN-DEST
PRINT #DYN-SRC #DYN-DEST ' *TRIM(#SRC)''

PRINT ' '

PRINT 'hex."20" := space character'
END

```

Output of Example 2:

```

static variable definition:
-----

2020FFFF2020FFFF2020 0000FFFF2020FFFF2020 *TRIM(#src, leading)
2020FFFF2020FFFF2020 00002020FFFF2020FFFF *TRIM(#src, trailing)
2020FFFF2020FFFF2020 00000000FFFF2020FFFF *TRIM(#src)

dynamic variable definition:
-----

2020FFFF2020FFFF2020 FFFF2020FFFF2020 *TRIM(#src, leading)

```

```
2020FFFF2020FFFF2020 2020FFFF2020FFFF  
2020FFFF2020FFFF2020 FFFF2020FFFF
```

```
*TRIM(#src, trailing)  
*TRIM(#src)
```

```
hex.'20' := space character
```