# FOR

```
FOR  operand1  ⎧ [:]=  ⎫  ⎧ operand2                ⎫
               ⎨ EQ    ⎬  ⎨ (arithmetic-expression) ⎬
               ⎩ FROM  ⎭  ⎩                         ⎭

               ⎧ TO    ⎫  ⎧ operand3                ⎫
               ⎨ THRU  ⎬  ⎨ (arithmetic-expression) ⎬
               ⎩       ⎭  ⎩                         ⎭

               ⎡ STEP  ⎧ operand4                ⎫ ⎤
               ⎢       ⎨ (arithmetic-expression) ⎬ ⎥
               ⎣       ⎩                         ⎭ ⎦

               statement ...

END-FOR        (structured mode only)

[LOOP]         (reporting mode only)
```

**Note:**
For compatibility reasons, the keywords `:=`, `EQ`, `FROM`, `TO`, `THRU` and `STEP` are optional if the corresponding subsequent operand (*operand2*, *operand3* or *operand4*) is used instead of an arithmetic expression.

This chapter covers the following topics:

- Function

- Syntax Description

- Example

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Related Statements: REPEAT | ESCAPE

Belongs to Function Group: *Loop Execution*

---

# Function

The `FOR` statement is used to initiate a processing loop and to control the number of times the loop is processed.

## Consistency Check

Before the `FOR` loop is entered, the values of the operands are checked to ensure that they are consistent (that is, the value of *operand3* can be reached or exceeded by repeatedly adding *operand4* to *operand2*). If the values are not consistent, the `FOR` loop is not entered (however, no error message is output, except when the `STEP` value is zero).

# Syntax Description

Operand Definition Table:

| Operand | | Possible Structure | | | | | | Possible Formats | | | | | | | | | | Referencing Permitted | Dynamic Definition |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *operand1* | | S | | | | | | N | P | I | F | | | | | | | yes | yes |
| *operand2* | C | S | | | N | E | | N | P | I | F | | | | | | | yes | no |
| *operand3* | C | S | | | N | E | | N | P | I | F | | | | | | | yes | no |
| *operand4* | C | S | | | N | E | | N | P | I | F | | | | | | | yes | no |

Syntax Element Description:

| Syntax Element | Description |
|---|---|
| *operand1* | **Loop Control Variable (*operand1*) and Initial Setting (*operand2*):** |
| | *operand1* is used to control the number of times the processing loop is to be executed. It may be a database field or a user-defined variable. The value specified after the keyword FROM (*operand2*) is assigned to the loop control variable field before the processing loop is entered for the first time. This value is |
| *operand2* | incremented (or decremented if the STEP value is negative) using the value specified after the STEP keyword (*operand4*) each additional time the loop is processed. |
| | The loop control variable value may be referenced during the execution of the processing loop and will contain the current value of the loop control variable. |
| *operand3* | **TO Value:** |
| | The processing loop is terminated when *operand1* is greater than (or less than if the initial value of the STEP value was negative) the value specified for *operand3*. |
| *operand4* | **STEP Value:** |
| | The STEP value may be positive or negative. If a STEP value is not specified, an increment of +1 is used. |
| | The compare operation will be adjusted to "less than" or "greater than" depending on the sign of the STEP value when the loop is entered for the first time. |
| | *operand4* must not be zero. |
| (*arithmetic-expression*) | **Arithmetic Expression:** |
| | In place of *operand2*, *operand3* or *operand4*, any arithmetic expression may be specified. For further information on arithmetic expressions, see *arithmetic-expression* in the COMPUTE statement description. |
| | **Note:** The arithmetic expression must be enclosed in parentheses. |
| END-FOR | **End of FOR Statement:** |
| | The Natural reserved word END-FOR must be used to end the FOR statement. |

# Example

```
** Example 'FOREX1S': FOR (structured mode)
************************************************************************
DEFINE DATA LOCAL
1 #INDEX (I1)
1 #ROOT  (N2.7)
END-DEFINE
*
FOR #INDEX 1 TO 5
  COMPUTE #ROOT = SQRT (#INDEX)
  WRITE NOTITLE '=' #INDEX 3X '=' #ROOT
END-FOR
*
SKIP 1
FOR #INDEX 1 TO 5 STEP 2
  COMPUTE #ROOT = SQRT (#INDEX)
  WRITE '=' #INDEX 3X '=' #ROOT
END-FOR
*
END
```

### Output of Program FOREX1S:

```
#INDEX:    1   #ROOT:   1.0000000
#INDEX:    2   #ROOT:   1.4142135
#INDEX:    3   #ROOT:   1.7320508
#INDEX:    4   #ROOT:   2.0000000
#INDEX:    5   #ROOT:   2.2360679

#INDEX:    1   #ROOT:   1.0000000
#INDEX:    3   #ROOT:   1.7320508
#INDEX:    5   #ROOT:   2.2360679
```

Equivalent reporting-mode example: FOREX1R.