Breakpoint Maintenance

A breakpoint causes the execution of a Natural object to be interrupted at a specific statement line. This section describes how and when to set breakpoints. Note that the maintenance functions described here may also be invoked from an object source by using the **List object source** function.

To invoke Breakpoint Maintenance

- In the **Debug Main Menu**, enter function code B.

  Or:
  Enter the following direct command:

  ```
  BM
  ```

  The **Breakpoint Maintenance** menu appears.

This section describes conditions for using breakpoint maintenance, the functions provided in the **Breakpoint Maintenance** menu and the fields and columns contained in a breakpoint screen.

- Conditions of Use
- Set Test Mode ON/OFF
- Activate Breakpoint
- Deactivate Breakpoint
- Delete Breakpoint
- Display Breakpoint
- Modify Breakpoint
- Set Breakpoint
- Fields and Columns on Breakpoint Screens

Conditions of Use

A breakpoint is set by specifying the name of the Natural object to be processed and the line number in the object’s source code where the breakpoint is to be executed.

Once a breakpoint has been specified, it remains set for the entire Natural session, unless you delete it.

A breakpoint refers to a specific line number in source code. A subsequent change of the source code itself may therefore lead to the breakpoint no longer applying to the desired statement, and thus the Natural object not being interrupted at the desired position. To circumvent this problem with program loops, labels can be set within these loops. Breakpoints set for these labels are adjusted to the correct line number if statement lines are inserted or deleted.

The unique identifier for a breakpoint is the spy number as assigned by the debugger.

Breakpoints cannot be set on comment lines, on any statement line other than the first one (if a single statement occupies more than one program line), and on lines that contain one of the following statements only:

- `AT BREAK OF`
- `AT END OF DATA`
- `AT END OF PAGE`
- `AT START OF DATA`
- `AT TOP OF PAGE`
- `BEFORE BREAK`
- `DECIDE`
  See also the usage restrictions described in *Operational Requirements*.
- `DEFINE SUBROUTINE`
- `DEFINE WINDOW`
- `FORMAT`
- `IF NO RECORDS FOUND`
- `ON ERROR`
- `OPTIONS`

Whether it is possible or not to set breakpoints for lines compiled with the Natural Optimizer Compiler depends on the `NODBG` option of the `OPTIONS` statement described in *Switching on the Optimizer Compiler* in the *Natural Optimizer Compiler* documentation.

Set Test Mode ON/OFF

See the section *Switch Test Mode On and Off*.

Activate Breakpoint

To set the current state of specified breakpoints to active

- In the **Breakpoint Maintenance** menu, enter function code `A`, an object name and/or a line number.

  Or:
  Use the direct command `ACTIVATE`, the syntax of which is described in the section *Command Summary and Syntax*.

If you do not specify an object name or a line number, *all* breakpoints are activated.

Deactivate Breakpoint

To set the current state of specified breakpoints to inactive

- In the **Breakpoint Maintenance** menu, enter function code `B`, an object name and/or a line number.

  Or:
  Use the direct command `DEACTIVATE`, the syntax of which is described in the section *Command Summary and Syntax*.

  If you do not specify an object name or a line number, *all* breakpoints are deactivated.

Delete Breakpoint

To delete specified breakpoints

- In the **Breakpoint Maintenance** menu, enter function code `C`, an object name and/or a line number.

  Or:
  Use the direct command `DELETE`, the syntax of which is described in the section *Command Summary and Syntax*.

  If you do not specify an object name or a line number, *all* breakpoints are deleted.

Display Breakpoint

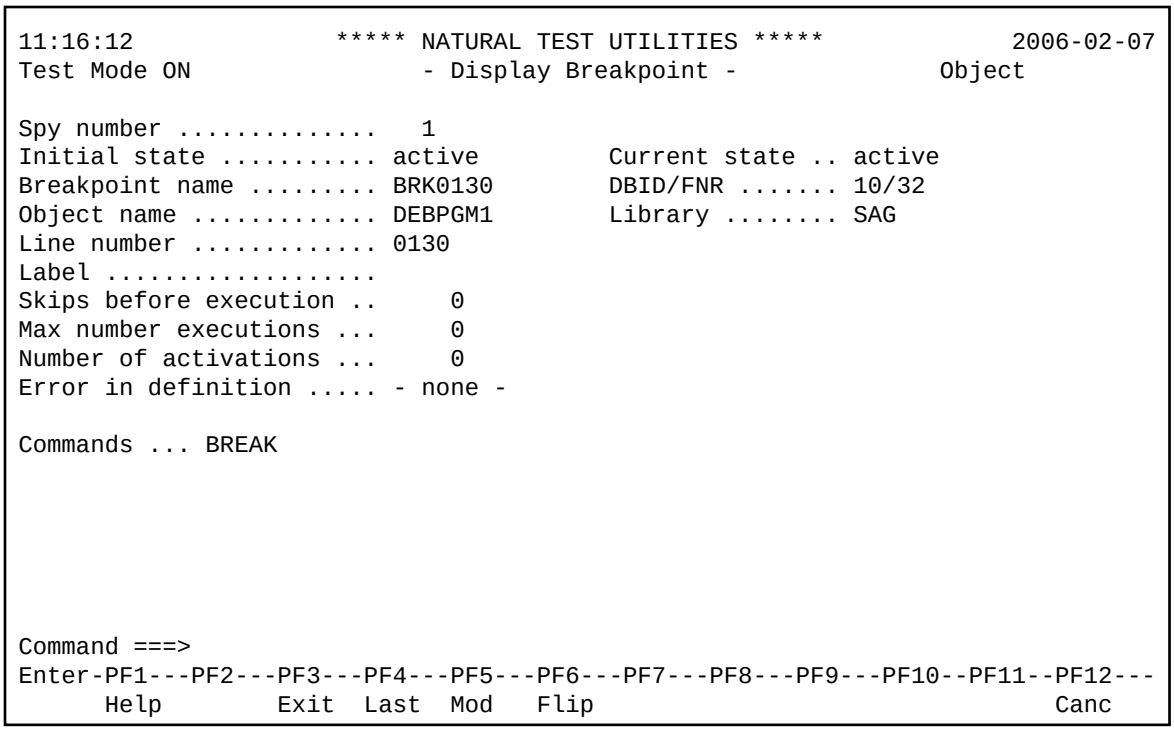
To display a breakpoint

- In the **Breakpoint Maintenance** menu, enter function code `D`, an object name and a line number.

  If you do not enter an object name, the default object (if specified) is used.

  Or:
  Use the direct command `DISPLAY`, the syntax of which is described in the section *Command Summary and Syntax*.

  If a breakpoint has been set for the specified object and line number, a **Display Breakpoint** screen with all breakpoint definitions appears similar to the example below:



If no unique breakpoint is found, the **List Breakpoints** screen described below appears.

The fields on the **Display Breakpoint** screen are described in *Fields and Columns on Breakpoint Screens*.

To list breakpoints

- In the **Breakpoint Maintenance** menu, enter function code `D`, an object name or a line number. You can use asterisk (*) notation to specify a range of object names, for example, `ABC*`. If you enter an asterisk (*) only, all object names are selected. If you do not enter an object name, the default object (if specified) is used.

  Or:
  Use the direct command `DISPLAY`, the syntax of which is described in the section *Command Summary and Syntax*.

  A **List Breakpoints** screen similar to the example below appears which lists all breakpoints set for the specified object(s) or line number:

```
11:41:56              ***** NATURAL TEST UTILITIES *****            2006-01-30
Test Mode ON                   - List Breakpoints -                Object
                                                                          All
Co No. BP Name      Library  Object   Line  DBID   FNR Stat Skips Execs Count E
       *____________ *_______ *_______ 0000            I  C
__   1 BRK0130      SAG      DEBPGM1  0130    10    32 A  A     0     0     0
__   2 BRKPGM3-END  SAG      DEBPGM3  END     10    32 A  A     0     0     0
__   3 BRKPGM3-300  SAG      DEBPGM3  0300    10    32 A  A     0     0     0
__   4 BRKPGM2-400  SAG      DEBPGM2  0400    10    32 A  A     0     0     0
__   5 BRKPGM2-430  SAG      DEBPGM2  0430    10    32 A  A     0     0     0
__   6 BRKPGM1-END  SAG      DEBPGM1  END     10    32 A  A     0     0     0
__   7 BRKPGM1-ALL  SAG      DEBPGM1  ALL     10    32 A  A     0     0     0









Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Last        Flip  -     +                       Canc
```

The list is sorted in ascending order by the spy numbers contained in the **No.** column.

For details on the columns contained in the **List Breakpoints** screen and the line commands that can be executed on any list item, refer to *Fields and Columns on Breakpoint Screens*.

Modify Breakpoint

To modify a breakpoint

1. In the **Breakpoint Maintenance** menu, enter function code `M`, an object name and a line number. If you do not enter an object name, the default object (if specified) is used.

   Or:
   Use the direct command `MODIFY`, the syntax of which is described in the section *Command Summary and Syntax*.

   If a unique breakpoint has been specified, the **Modify Breakpoint** screen appears where you can change the field entries. The fields on the **Modify Breakpoint** screen are described in *Fields and Columns on Breakpoint Screens*.

   If no unique breakpoint is found, the **List Breakpoints** screen (see *Display Breakpoint*) appears.

2. When you have finished editing the breakpoint definitions, choose PF3 (Exit) or PF5 (Save) to save any modification. See also *Maintenance and Validation* for information on validity checks of debug entries. If you choose PF12 (Canc), the breakpoint remains unchanged.

Set Breakpoint

To add a breakpoint for a session

- In the **Breakpoint Maintenance** menu, enter function code S, an object name and/or a line number.

  Or:
  Use the direct command SET, the syntax of which is described in the section *Command Summary and Syntax*.

  If you specify not an object name but a valid line number, the name of the default object (see the section *Start the Debugger*) is assumed. If no default object is specified, a selection window appears that displays all objects available in the current library.

  If object name and line number are specified correctly, the breakpoint is usually set and confirmed immediately.

  However, a breakpoint set for copycode can only be validated when a program that contains the copycode is executed. See also *Maintenance and Validation* for information on validity checks of debug entries.

  The breakpoint receives the default command (BREAK), its initial and current state are set to active and no execution restrictions are specified. Note that if you delete the command BREAK when setting a breakpoint and you do not enter any command that issues a dialog, there is no way for the debugger to receive control during program interruption.

Fields and Columns on Breakpoint Screens

The fields contained in a **Display Breakpoint** or a **Modify Breakpoint** screen and the columns of a **List Breakpoints** screen are described in the following table:

Field	Column	Explanation
Test Mode		Indicates whether test mode is set to ON or OFF.
Object		Displays the name of the default object (see *Start the Debugger*) if specified.
	Co	Input field for any of the following line commands: AC Activate breakpoint DA Deactivate breakpoint DI Display breakpoint MO Modify breakpoint DE Delete breakpoint ? List valid line commands . Exit breakpoint screen

Field	Column	Explanation
Spy number	**No.**	A unique number assigned by the debugger when setting the breakpoint.
Initial state	**Stat I**	Specifies the initial state and the current state of the breakpoint: active (`A`) or inactive (`I`).
Current state	**Stat C**	
Breakpoint name	**BP Name**	The name of the breakpoint. Valid values: 1 to 12 characters. The default name for a breakpoint consists of the object name and the line number.
DBID/FNR	**DBID**	The database ID (DBID) and file number (FNR) of the system file where the Natural object is stored.
	FNR	
Library	**Library**	The name of the library that contains the object.
Object name	**Object**	The name of the object available in the current library or one of its steplibs.
Line number	**Line**	The line number of a statement in the object source code. See also *Conditions of Use* above. You can also specify `BEG`, `END` or `ALL` as line numbers: `BEG` Specifies the breakpoint that is to interrupt program execution at the first statement executed in an object. `BEG` breakpoints cannot be specified for copycode. `END` Specifies the breakpoint that is to interrupt program execution at the last statement executed in an object, for example, an `END` or a `FETCH` statement. `END` breakpoints cannot be specified for copycode. `ALL` Specifies that a breakpoint is to interrupt program execution at each program line that contains an executable statement.
Label		Refers to a label set earlier in the source code of an object for statements that define processing loops: see also *Conditions of Use* above. Valid values: 1 to 32 characters.
Skips before execution	**Skips**	Determines that the breakpoint is not to be executed until the corresponding statement line has been executed a certain number of times. Valid values: 0 (default) to 32767.

Field	Column	Explanation
Max number executions	**Execs**	Any value greater than zero (0) determines the maximum number of breakpoint executions. Valid values: 0 (default) to 32767.
Number of activations	**Count**	Indicates how many times a breakpoint was activated for the relevant statement line. The counter is reset when a program is started at Level 1.
Error in definition	**E**	Indicates that the statement line in the breakpoint definition cannot be found in the cataloged object during program execution. This error can be caused if the source of an object is changed and recataloged during debugging.
Commands		Up to six debug commands. Enter one command per line. For a summary of all available commands, see *Command Summary and Syntax*. **Caution:** If you delete the command BREAK when modifying a breakpoint and you do not enter any command that issues a dialog, there is no way for the debugger to receive control during program interruption.