Natural für Windows

Debugger

Version 6.3.8 für Windows

Februar 2010

Natural

Dieses Dokument gilt für Natural ab Version 6.3.8 für Windows.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Inhaltsverzeichnis

1 Debugger

Diese Dokumentation, die die Dokumentation *Natural Studio benutzen* ergänzt, erläutert, wie Fehler in einer ein Natural-Anwendungen mit dem Debugger bereinigt werden. Sie ist in die folgenden Abschnitte unterteilt:

Allgemeine Informationen	Über den Debugger, der in Natural Studio integriert ist. Informationen zu Remote-Debugging und wie Sie Ihre Umgebung für das Remote-Debugging einrichten.
Debugger starten und verlassen	Informationen zum Parameter `SYMGEN`. Wie der Debugger gestartet, neu gestartet und beendet wird.
Elemente des Debuggers	Informationen zu zusätzlichen Elementen, die nach dem Starten des Debuggers im Natural Studio-Fenster zur Verfügung stehen.
Durch den Code gehen	Wie Code schrittweise ausgeführt wird und wie man zu Breakpoints, Watchpoints, Events oder zur Cursor-Position geht.
Breakpoints und Watchpoints setzen	Wie Breakpoints und Watchpoints erstellt und im Break- und Watchpoints-Fenster verwaltet werden.
Variablen ändern und überwachen	Wie Variablen geändert, Watch-Variablen erstellt, und Variablen im Variablen-Fenster verwaltet werden.
Call-Stack benutzen	Wie Objekte im Call-Stack-Fenster verwaltet werden.

Zusätzlich zu den oben aufgeführten Themen, die den neuen Debugger beschreiben, der in Natural Studio integriert ist, steht auch die Dokumentation für den alten Debugger im Abschnitt ***Alten Debugger benutzen*** zur Verfügung. Der alte Debugger kann erscheinen, wenn eine alte Version von Natural auf dem Entwicklungs-Server installiert ist; weitere Informationen finden Sie im Abschnitt ***Allgemeine Informationen***.

2 Allgemeine Informationen

Über den Debugger

Ab Natural für Windows Version 6.2 ist der Debugger in Natural Studio integriert. Die vollständige Natural Studio-Funktionalität kann somit parallel zum Debugger benutzt werden. Wenn beispielsweise der Debugger aktiv ist, können Sie zu einem anderen Objekt im Library-Workspace navigieren, oder aber können Sie mit dem Befehl **Find Object** nach einem bestimmten Objekt suchen.

Der Debugger wird benutzt, um Natural-Anwendungen in den folgenden Umgebungen auszutesten:

- in der lokalen Umgebung und
- in einer Remote-Umgebung, d.h.: auf einem per Mapping zugeordneten Entwicklungs-Server (SPoD). Die Voraussetzung dazu ist, dass eine der folgenden Versionen auf dem Entwicklungs-Server installiert ist:
  - Natural for Mainframes Version 4.2 oder darüber.
  - Natural for UNIX Version 6.2 oder darüber.

Für den Debugger sind keine zusätzlichen Einstellungen erforderlich. Natural Studio wickelt alle Schritte intern ab (wie z.B. das Herstellen oder Beenden der Kommunikation mit dem entsprechenden Server).

Siehe auch *Umgebungen und Views im Library Workspace* in der Dokumentation *Natural Studio benutzen* und *Remote Entwicklungsumgebung aufrufen* in der Dokumentation *Remote-Entwicklung mit SPoD*.

Anmerkung: Es gibt mehrere Unterschiede, wenn Sie Anwendungen in einer Remote-Großrechnerumgebung austesten. Diese Unterschiede sind in der plattform-spezifischen Natural Development Server-Dokumentation (NDV) aufgeführt, die für diese Natural-Release gilt. Die NDV-Dokumentation steht separat zur Verfügung; sie ist nicht Bestandteil der Natural für Windows-Dokumentation.

Wann wird der alte Debugger noch mit SPoD benutzt?

Wenn Sie mit Natural Studio arbeiten und den Debugger für ein Objekt auf einem per Mapping zugeordneten Entwicklungs-Server aufrufen, ist es möglich, dass der alte Debugger anstatt des neuen integrierten Debuggers aufgerufen wird. Dies ist der Fall, wenn eine alte Version von Natural auf dem Entwicklungs-Server installiert ist. Es gibt die folgenden alten Versionen:

- Natural for UNIX Version 6.1.1 oder darunter.

Siehe ***Alten Debugger benutzen***.

Remote-Debugging

Mit einer der nächsten Versionen wird Remote-Debugging nicht mehr unterstützt. Statt dessen müssen Sie dann den Debugger benutzen, der in Natural Studio integriert ist.

Remote-Debugging erfolgt, wenn Sie eine ursprüngliche Natural für UNIX-Anwendung von einem Windows-Computer aus austesten, oder wenn Sie eine Natural-Dialoganwendung von einem anderen PC aus im Remote-Betrieb austesten. Dies erfolgt außerhalb des Rahmens von SPoD.

Um Remote-Debugging einzuschalten, müssen Sie wie folgt vorgehen:

- Installieren Sie das Debug-Frontend auf einem Windows-Computer. Dadurch wird auch der Remote-Debugging-Service `natdbgsv` installiert, der für Remote-Debugging aktiv sein muss. Siehe ***Remote-Debugger installieren***.
- Definieren Sie die Parameter `RDNODE`, `RDPORT` und `RDACTIVE` in der Umgebung, die die Anwendung enthält, die ausgetestet werden soll. Weitere Informationen finden Sie unter ***Einrichten Ihrer Umgebung für Remote-Debugging***.
- Rufen Sie den Debugger auf, indem Sie das Systemkommando `DEBUG` *`objektname`* in der Umgebung eingeben, die die Anwendung enthält, welche ausgetestet werden soll.

Die folgenden Themen werden nachfolgend behandelt:

⚠ **Wichtig:** Um den Remote-Debugger unter Microsoft Windows zu starten, muss die Personal-Firewall deaktiviert sein. Siehe *Configuring the Microsoft Windows Personal Firewall to Run Natural* in der *Operations*-Dokumentation für Natural für Windows.

Remote-Debugger installieren

Wenn bei Ihnen Natural für Windows installiert ist, müssen Sie den mit Natural für Windows ausgelieferten Remote-Debugger benutzen. Wenn der Remote-Debugger noch nicht installiert wurde, benutzen Sie die Option **Modify** (Ändern) des Natural-Installationspakets, um den Remote-Debugger zu Ihrer Natural für Windows-Installation hinzuzufügen. Siehe *Maintaining Your Natural or Natural Runtime Environment* in der *Installation*-Dokumentation für Natural für Windows.

Sie müssen den Remote-Debugger nur dann alleinstehend installieren, wenn Natural für Windows bei Ihnen nicht installiert haben. Wenn Sie eine Natural-Anwendung austesten möchten, die auf einer UNIX-Plattform gespeichert wird, kopieren Sie *$NATDIR/$NATVERS/dbrmt/I386/nrd.exe* vom UNIX-Installationsmedium auf Ihren Windows-Computer (zum Beispiel in ein temporäres

Verzeichnis) und entpacken Sie die Datei. Starten Sie *setup.exe*, um die Installation des Remote-Debugger zu starten.

Einrichten Ihrer Umgebung für Remote-Debugging

Die folgenden Themen werden nachfolgend behandelt:

- Windows-seitig ohne Terminal-Services
- Windows-seitig mit Terminal-Services
- Natural-seitig

Windows-seitig ohne Terminal-Services

Installieren Sie entweder den Remote-Debugger (die entsprechenden Dateien finden Sie auf dem UNIX-Installationsmedium), oder installieren Sie Natural für Windows (der Remote-Debugger kann optional mit dem Setup-Typ "Custom" von Natural für Windows installiert werden; siehe die *Installation*-Dokumentation für Natural für Windows). Dadurch wird auch der Natural-Remote-Debugging-Service `natdbgsv` installiert.

Um den Remote-Debugging-Service zu deinstallieren, geben Sie `natdbgsv -u` in der Kommandozeile ein. Um den Portnamen und die Version des aktuellen Service zu erfahren, geben Sie `natdbgsv -s` ein. Um den Service auf einem anderen Port neu zu installieren, deinstallieren Sie ihn zuerst, und geben Sie dann `natdbgsv -i portnummer` ein, wobei *portnummer* der Wert des Profilparameters `RDPORT` ist. Wenn die Portnummer bereits in Benutzung ist, erscheint ein Dialog, in dem Sie eine neue Portnummer eingeben können.

Anmerkung: Bevor Sie den Remote-Debugging-Service auf einem anderen Port als dem 2600er (Standardwert) installieren, müssen Sie den Wert des Profilparameters `RDPORT` ändern, so dass er der Portnummer des Client-Computers entspricht, auf dem die Natural-Anwendung ausgetestet wird.

Windows-seitig mit Terminal-Services

Installieren Sie den Remote-Debugger (die entsprechenden Dateien finden Sie auf dem UNIX-Installationsmedium), oder installieren Sie Natural für Windows (der Remote-Debugger kann optional mit dem Setup-Typ **Custom** von Natural für Windows installiert werden; siehe die *Installation*-Dokumentation für Natural für Windows). Dadurch wird auch die Debugger-Verknüpfung für den Listener-Prozess `natdbgsv` im **Start**-Menü erstellt (in demselben Programme-Verzeichnis, in dem Sie die Verknüpfungen für Natural finden können). Um Remote-Debugging zu verwenden, muss `natdbgsv` gestartet werden. Wenn der Listener-Prozess zum ersten Mal in einer bestimmten Benutzer-Session aufgerufen wird, wird eine freie Portnummer angezeigt, die in dem entsprechenden Feld des Profilparameters `RDPORT` eingegeben werden muss.

Jede nachfolgende Aktivierung von `natdbgsv` bewirkt, dass der Listener mit derselben Portnummer gestartet wird. Wenn diese Nummer bereits von einer anderen Anwendung benutzt wird, dann

muss der Benutzer im Port-Dialog von `natdbgsv` eine neue Portnummer angeben, und `RDPORT` muss entsprechend angepasst werden.

Natural-seitig

Starten Sie Natural mit den folgenden Profilparameter-Einstellungen:

- `RDACTIVE` ist auf "ON" gesetzt.
- `RDNODE` mit dem Knotennamen des Windows-Servers.
- `RDPORT` ist auf "2600" oder eine andere Portnummer gesetzt: entweder die Nummer des Ports, auf dem Sie den Remote-Debugging-Service installiert haben (siehe ***Windows-seitig ohne Terminal-Services***), oder des Ports auf dem der Listener-Prozess gestartet wurde (siehe ***Windows-seitig mit Terminal-Services***).

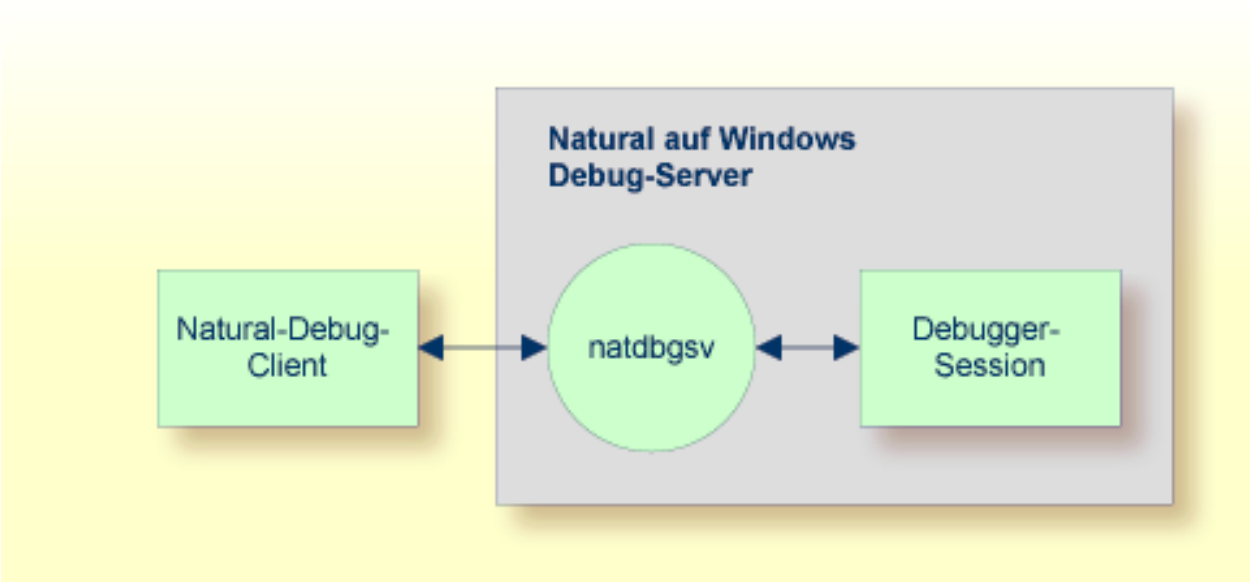
Szenarien für Remote-Debugging

Es gibt unterschiedliche Szenarien, wie Sie Remote-Debugging verwenden können: Ein einzelner Natural-Client läuft unter der Kontrolle einer Remote-Debugging-Session, oder eine verteilte Natural-Anwendung läuft unter der Kontrolle mehrerer Remote-Debugging-Sessions. Eine solche verteilte Anwendung kann sowohl Natural-RPC- und DCOM-Server enthalten als auch nicht in Natural geschriebene Komponenten, wie z.B. Visual Basic-Clients.

Die folgenden Themen werden nachfolgend behandelt:

Szenario 1: Einzelne Natural-Anwendung austesten

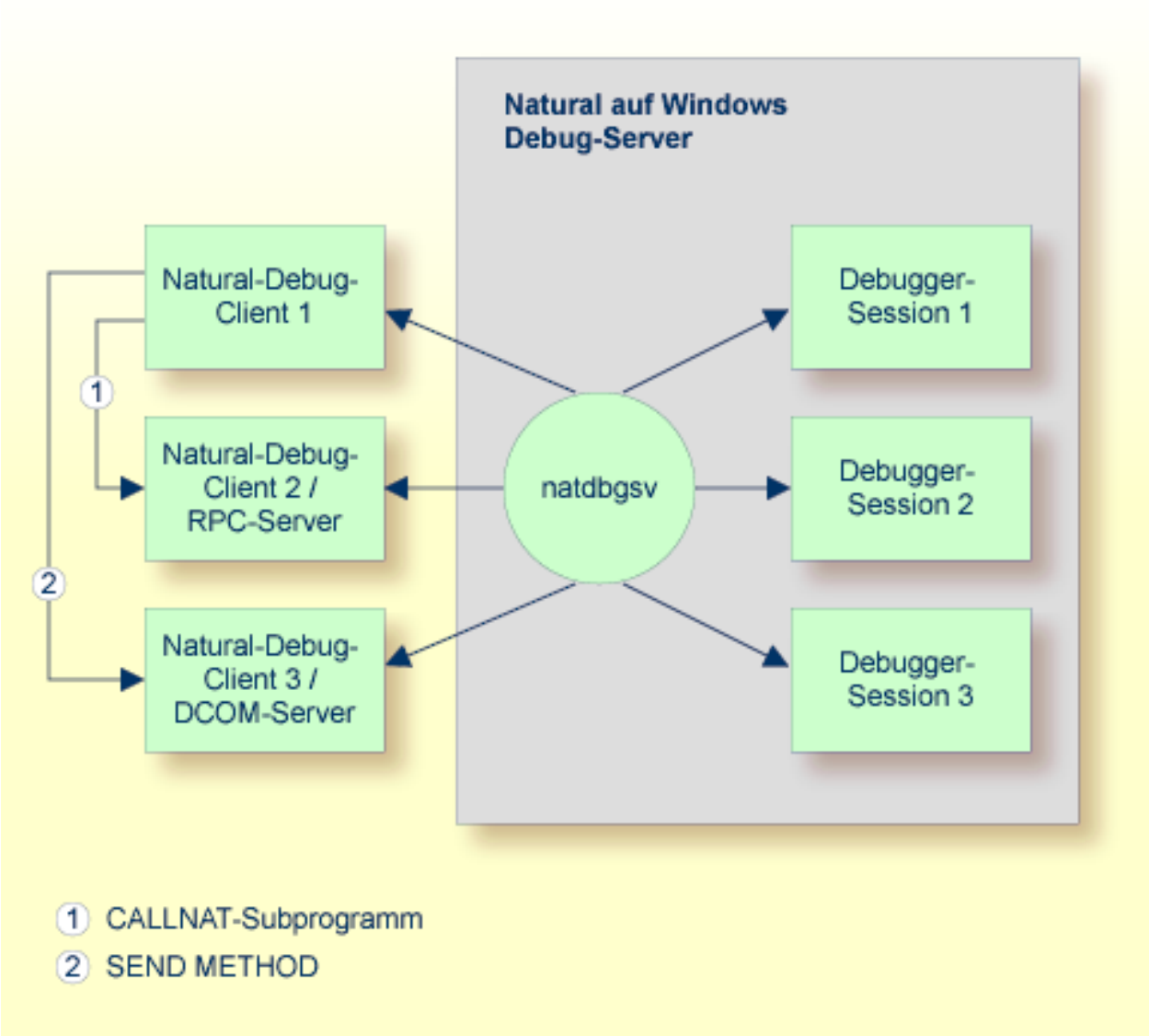
Die folgende Abbildung veranschaulicht das Austesten einer einzelnen Natural-Anwendung.



Szenario 2: Verteilte Natural-Anwendung austesten

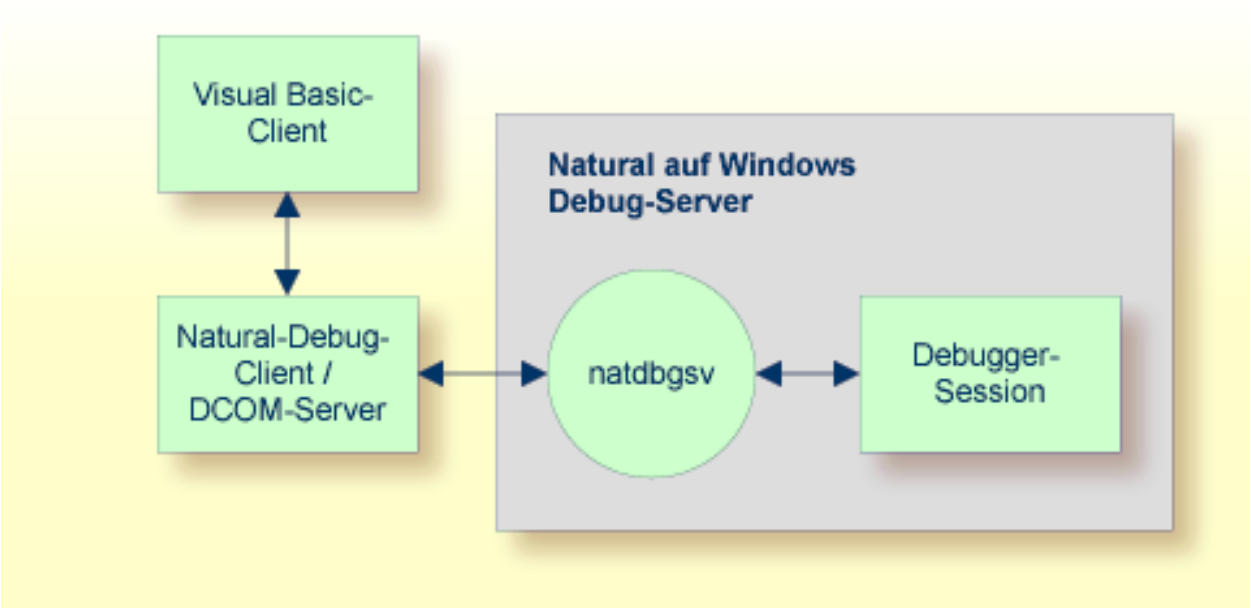
Um jede Komponente der folgenden verteilten Natural-Anwendung auszutesten, geben Sie in der Kommandozeile des Natural-Debug-Clients 1 das folgende Kommando ein: `DEBUG objektname`. Wenn der Natural-Debug-Client zum ersten Mal ein Subprogramm auf einem Natural-RPC-Server aufruft, wird eine neue Debugging-Session für den RPC-Server geöffnet. Dann wird die Verarbeitung des RPC-Servers ausgetestet. Die Debugging-Session wird geschlossen, sobald der RPC-Server beendet wird.

Dasselbe gilt für einen Natural-DCOM-Server.



Szenario 3: Natural-Teil einer heterogenen Anwendung austesten

Wie auch im vorherigen Szenario: Wenn eine Methode auf dem DCOM-Server zum ersten Mal aufgerufen wird, dann wird eine neue Debugging-Session für den DCOM-Server geöffnet, die Verarbeitung des DCOM-Servers wird ausgetestet, und die Debugger-Session wird geschlossen, sobald der DCOM-Server beendet wird:


Visual Basic-
Client
Natural auf Windows
Debug-Server
Natural-Debug-
Client /
DCOM-Server
natdbgsv
Debugger-
Session

3 Debugger starten und verlassen

Benutzung des Debuggers vorbereiten

Damit Sie die vollständige Funktionalität des Debuggers ausnutzen können, müssen Sie den Parameter SYMGEN auf "ON" setzen. Sie können diesen Parameter auf eine der folgenden Arten setzen:

- dynamisch beim Start von Natural,
- nur für die aktuelle Session durch Ändern der Session-Parameter, oder
- in Ihrer Parameterdatei unter Benutzung der Configuration Utility.

Wenn Sie ein Objekt katalogisieren oder mit dem Befehl **Stow** speichern, und SYMGEN auf "ON" gesetzt ist, wird eine Symboltabelle als Teil des generierten Programms generiert. Da diese Tabelle Informationen über die aktiven Variablen für dieses Objekt enthält, können Variablen ohne Angabe von SYMGEN nicht aufgerufen werden; es ist jedoch noch möglich, das Objekt auszutesten.

Anmerkung: Es ist nicht erforderlich, den Parameter SYMGEN zu setzen, wenn Sie die Debug-Funktion in einer SPoD-Umgebung auf einem Großrechner durchführen.

Debugger starten

Der Debugger kann zusammen mit Natural-Programmen und -Dialogen benutzt werden, die in Source- und Objekt-Form gespeichert (Stow) oder katalogisiert wurden. Er kann in der lokalen Umgebung und in der Remote-Umgebung benutzt werden.

Siehe auch die Beschreibung des Systemkommandos DEBUG.

Debugger starten

1 Öffnen Sie den Editor für das auszutestende Objekt.

Oder:

Markieren Sie das Objekt im Library-Workspace.

2 Wählen Sie aus dem Menü **Debug** den Befehl **Start**.

Oder:

Drücken Sie STRG+F7.

Oder:

Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Oder:

Wenn Sie ein Objekt im Library-Workspace markiert haben, rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Debug**.

Wenn der Editor für das ausgewählte Objekt noch nicht geöffnet wurde, so wird er jetzt geöffnet.

Für einen Dialog erscheint die Dialog-Source jetzt in einem separaten Fenster.

Nachdem der Debugger gestartet wurde, stehen zusätzliche Elemente im Natural Studio-Fenster zur Verfügung. Weitere Informationen finden Sie im Abschnitt *Elemente des Debuggers*.

Debugger neu starten (Restart)

Wenn Sie Ihre Debugging-Session neu starten, stellt sich der Debugger wieder an den Anfang der Anwendung, wobei alle Ihre aktuellen Einstellungen für Breakpoints, Watchpoints und Watch-Variablen beibehalten werden. Der erneute Start einer Debugging-Session ist also nützlich, wenn Sie Ihre Anwendung noch einmal laufen lassen möchten, ohne die für das Austesten relevanten Einstellungen noch einmal angeben zu müssen.

Debugger neu starten

- Wählen Sie aus dem Menü **Debug** den Befehl **Restart**.

  Oder:

  Drücken Sie STRG+UMSCHALT+F7.

  Oder:

Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Debugger verlassen

Der Debugger wird automatisch beendet, wenn die Anwendung ohne einen Fehler abgeschlossen wird. Sie können den Debugger auch stoppen, bevor er automatisch beendet wird; siehe die Beschreibung unten.

Anmerkung: Durch Schließen des Editor-Fensters wird der Debugger nicht gestoppt.

Wenn der Debugger beendet oder gestoppt wird, werden Ihre Einstellungen für Breakpoints, Watchpoints und Watch-Variablen automatisch gespeichert. Alle diese Einstellungen werden das nächste Mal wiederhergestellt, wenn Sie den Debugger starten.

Im Falle eines Fehlers wird die entsprechende Source angezeigt, und die Trace-Position (Position zur Ablaufverfolgung) verweist auf die Zeile, die den Fehler verursacht hat. Es erscheint ein Meldungsfenster mit der betreffenden Fehlermeldung und die Auswahlmöglichkeit, mit der Verarbeitung fortzufahren oder die Debugging-Session zu beenden. Eine Fortsetzung der Debugging-Session kann beispielsweise nützlich sein, wenn Ihre Anwendung eine Fehlerverarbeitung (einschließlich Fehler-Transaktionen) enthält, oder wenn Sie Variablen anzeigen möchten, bevor Sie Ihre Debugging-Session beenden.

Debugger stoppen

- Wählen Sie aus dem Menü **Debug** den Befehl **Stop**.

  Oder:

  Drücken Sie UMSCHALT+F7.

  Oder:

Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Die Debugging-Session wird beendet, und die Kontrolle wird an Natural zurückgegeben.

4 Elemente des Debuggers

Wenn der Debugger gestartet wurde, stehen zusätzliche Elemente im Natural Studio-Fenster zur Verfügung.

Debugger-Informationen in der Titelleiste

Die Titelleiste des Natural Studio-Fensters zeigt eine der folgenden Informationen an:

- **[break]**
  Wenn "[break]" in der Titelleiste angezeigt wird, hat der Debugger die Kontrolle.
- **[running]**
  When "[running]" in der Titelleiste angezeigt wird, hat die gerade ausgetestete Natural-Anwendung die Kontrolle.

Wenn Sie ein Objekt in einer Remote-Umgebung mit SPoD austesten, wird in der Titelleiste auch die Portnummer des Hosts angezeigt.

Menübefehle

Die Befehle im Menü **Debug** gelten für den Debugger.

Solange der Debugger noch nicht gestartet wurde, ist nur der Befehl **Start** im Menü **Debug** aktiviert. Wenn der Debugger gestartet wurde, werden die übrigen Befehle im Menü **Debug** aktiviert, und der Befehl **Go** wird anstatt des Befehls **Start** angezeigt.

Wenn ein Editor-Fenster aktiv ist, und der Debugger für das Objekt in diesem Fenster gestartet wurde, zeigt das Kontextmenü Befehle, die für den Debugger gelten. Solange der Debugger nicht gestartet wurde, steht nur der Debug-Befehl **Toggle Breakpoint** (Breakpoint ein-/ausschalten) im Kontextmenü zur Verfügung.

Detaillierte Beschreibungen dieser Befehle finden Sie weiter unten in dieser Dokumentation.

Symbolleiste

Der Debugger hat eine spezielle Symbolleiste, die einen schnellen Zugriff auf die Befehle im Menü **Debug** bietet. Solange der Debugger noch nicht gestartet wurde, sind in der Debug-Symbolleiste nur die Schaltflächen für die Befehle **Start** und **Toggle Breakpoint** (Breakpoint ein-/ausschalten) aktiviert. Wenn der Debugger gestartet wurde, sind alle anderen Schaltflächen in der Symbolleiste aktiviert.

Die Schaltflächen in der Debug-Symbolleiste stehen für die folgenden Menübefehle:

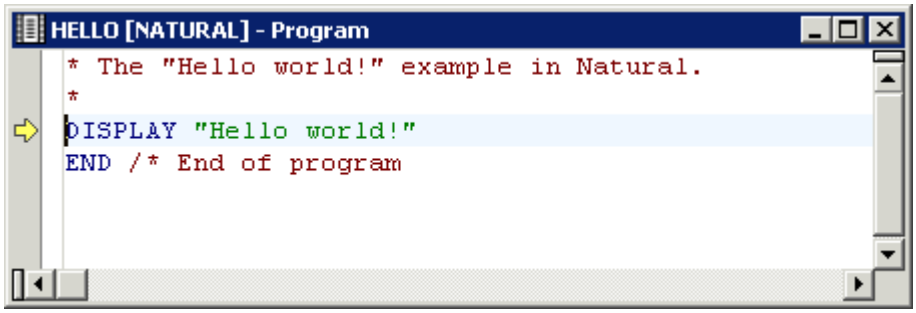
- **Start** (erscheint nur, wenn der Debugger noch nicht gestartet wurde)
- **Go** (erscheint nur, nachdem der Debugger gestartet wurde)
- **Restart** (Neu starten)
- **Stop**
- **Step Over** (Objekt überspringen)
- **Step Into** (Objekt schrittweise ausführen)
- **Step Out** (Objekt verlassen)
- **Show Trace Position** (Position zur Ablaufverfolgung zeigen)
- **Toggle Breakpoint** (Breakpoint ein-/ausschalten)
- **Modify Variable** (Variable ändern)

Die Anzeige der Debug-Symbolleiste kann ein- und ausgeschaltet werden. Weitere Informationen finden Sie unter *Natural Studio anpassen* in der Dokumentation *Natural Studio benutzen*.

Trace-Position in Editor-Fenstern

Die aktuelle Trace-Position (Position zur Ablaufverfolgung) wird durch einen blauen Pfeil am linken Rand des Editor-Fensters angezeigt.

Wenn der Debugger gestartet wird, wird die Trace-Position in der ersten ausführbaren Sourcecode-Zeile angezeigt. Beispiel:



Wenn Sie das Editor-Fenster durchgeblättert haben, so dass die Trace-Position nicht mehr sichtbar ist, können Sie wie unten beschrieben zur Trace-Position zurückkehren.

Anmerkung: Siehe auch *Zum Objekt an der aktuellen Trace-Position zurückkehren*.

Zur Trace-Position zurückkehren

- Wählen Sie aus dem Menü **Debug** den Befehl **Show Trace Position**.

  Oder:

  Drücken Sie ALT+NUM*.

  Anmerkung: NUM* ist die Taste auf dem numerischen Tastenfeld, die für die Multiplikation benutzt wird.

  Oder:

  Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Debugger-Fenster

Wenn der Debugger gestartet worden ist, erscheinen verschiedene Debugger-Fenster mit den folgenden Inhalten:

- Variablen
- Break- und Watchpoints
- Call-Stack

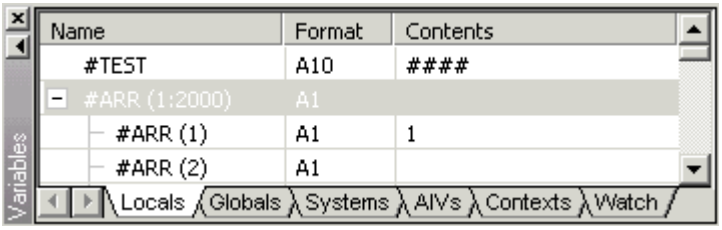
Jedes Register eines Debugger-Fensters bietet ein Kontextmenü, das entweder die Befehle enthält, die in Verbindung mit dem gesamten Register benutzt werden können (wenn ein Eintrag nicht markiert ist), oder die Befehle, die in Zusammenhang mit dem markierten Eintrag benutzt werden können. Diese Befehle sind weiter unten in dieser Dokumentation beschrieben.

Die Debugger-Fenster sind verschiebbar und andockbar. Siehe *Andockbare Fenster* in der Dokumentation *Natural Studio benutzen*.

Anmerkung: Wenn die Anzeige des Debugger-Fensters vorher mittels des entsprechenden Befehls im Menü **View** eingeschaltet worden ist, dann wird dieses Debugger-Fenster (im dem die Breakpoints und Watchpoints angezeigt werden, die in der aktiven Umgebung definiert wurden) durch die unten beschriebenen Fenster ersetzt. Siehe auch *Debugger-Fenster* in der Dokumentation *Natural Studio benutzen*.

Variablen

Dieses Fenster zeigt alle Variablen, die zum aktuellen Stand der Programmausführung zur Verfügung stehen.



Wenn links neben dem Variablennamen ein Umschaltsymbol zum auf- und zusammenklappen zu sehen ist, dann ist dies eine Gruppe, ein Array oder ein redefiniertes Feld. Das Symbol für ein redefiniertes Feld trägt ein zusätzliches "R" (zum Beispiel: +R).

Die Variablen sind in unterschiedliche Kategorien unterteilt. Es gibt ein Register für jede Kategorie:

- **Locals**
  Zeigt die lokalen Variablen, die im aktiven generierten Programm benutzt werden.
- **Globals**
  Zeigt die globalen Variablen der referenzierten Global-Data-Area.
- **Systems**
  Zeigt alle Systemvariablen auf der aktuellen Plattform. Wenn Sie beispielsweise gerade eine Anwendung in einer per Mapping zugeordneten UNIX-Umgebung austesten, erscheinen alle Systemvariablen, die für UNIX gültig sind.
- **AIVs**
  Zeigt die aktuell verfügbaren anwendungsunabhängigen Variablen (application-independent variables - AIVs) in der Anwendung.
- **Contexts**
  Zeigt die aktuell verfügbaren Kontextvariablen in der Anwendung.
- **Watch**
  Zeigt die Variablen, die Sie selbst hinzugefügt haben, um sie zu überwachen. Siehe *Watch-Variable hinzufügen*.

Sie können zwischen der Anzeige der unterschiedlichen Variablentypen umschalten, indem Sie das entsprechende Register am unteren Rand des Variablen-Fensters wählen.

Weitere Informationen finden Sie unter *Variablen ändern und überwachen*.

Anzeige des Variablen-Fensters ein- und ausschalten

- Wählen Sie aus dem Menü **Debug** den Befehl **Windows > Variables**.

Oder:

Drücken Sie STRG+ALT+1.

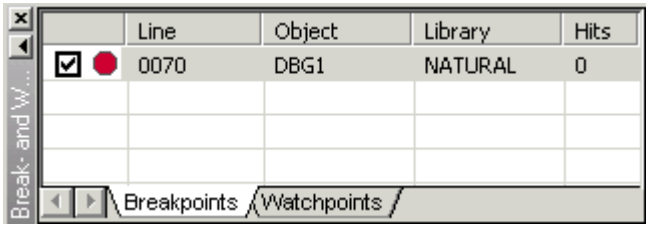
Wenn das Variablen-Fenster im Natural Studio-Fenster angezeigt wird, wird ein Häkchen neben diesem Menübefehl angezeigt.

Variablen-Fenster mittels einer Tastenkombination aktivieren

- Wenn das Variablen-Fenster im Natural Studio-Fenster angezeigt wird, drücken Sie STRG+UMSCHALT+V, um es zu aktivieren.

Break- und Watchpoints

Dieses Fenster zeigt alle aktuell definierten Breakpoints und Watchpoints.



Sie können zwischen der Anzeige der Watchpoints und Breakpoints umschalten, indem Sie das entsprechende Register am unteren Rand des Break- und Watchpoints-Fensters wählen.

Weitere Informationen finden Sie unter *Breakpoints and Watchpoints setzen*.

Anzeige des Break- und Watchpoints-Fensters ein- und ausschalten

- Wählen Sie aus dem Menü **Debug** den Befehl **Windows > Break- and Watchpoints**.

  Oder:

  Drücken Sie STRG+ALT+2.

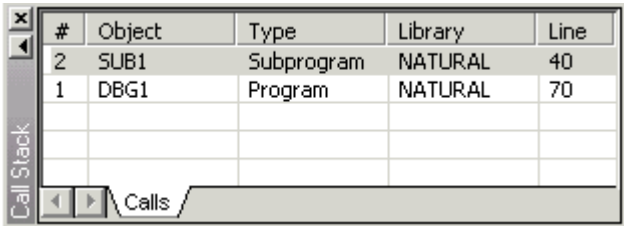
  Wenn das Break- und Watchpoints-Fenster im Natural Studio-Fenster angezeigt wird, wird ein Häkchen neben diesem Menübefehl angezeigt.

Break- und Watchpoints-Fenster mittels einer Tastenkombination aktivieren

- Wenn das Break- und Watchpoints-Fenster im Natural Studio-Fenster angezeigt wird, drücken Sie STRG+UMSCHALT+B, um es zu aktivieren.

Call-Stack

Dieses Fenster zeigt die Objekte, die während der aktuellen Debugging-Session aufgerufen worden sind in hierarchischer Reihenfolge.



Weitere Informationen finden Sie unter *Call-Stack benutzen*.

Anzeige des Call-Stack-Fensters ein- und ausschalten

- Wählen Sie aus dem Menü **Debug** den Befehl **Windows > Call Stack**.

  Oder:

  Drücken Sie STRG+ALT+3.

  Wenn das Call-Stack-Fenster im Natural Studio-Fenster angezeigt wird, wird ein Häkchen neben diesem Menübefehl angezeigt.

Call-Stack-Fenster mittels einer Tastenkombination aktivieren

- Wenn das Call-Stack-Fenster im Natural Studio-Fenster angezeigt wird, drücken Sie STRG+UMSCHALT+C, um es zu aktivieren.

5 Durch den Code gehen

Code schrittweise ausführen

Sie können den Debugger anweisen, den nächsten Programm-Schritt auszuführen. Unterschiedliche Befehle stehen zu diesem Zweck zur Verfügung:

- Ein anderes Objekt mit Step Over überspringen
- Ein anderes Objekt mit Step Into schrittweise ausführen
- Anderes Objekt mit Step Out verlassen

Ein anderes Objekt mit Step Over überspringen

Wenn Sie den Debugger anweisen, ein anderes Objekt mit **Step Over** zu überspringen, wird der nächste Programm-Schritt ausgeführt, und die Trace-Position erscheint in der entsprechenden Sourcecode-Zeile. Wenn diese Sourcecode-Zeile ein weiteres Natural-Objekt aufruft oder aufnimmt, überspringt der Debugger dieses Objekt, d.h. der gesamte Sourcecode dieses Objekts wird sofort ausgeführt. Der Debugger hält aber an, wenn dieses Objekt Watchpoints oder Breakpoints enthält.

Anderes Objekt mit Step Over überspringen

- Wählen Sie aus dem Menü **Debug** den Befehl **Step Over**.

  Oder:

  Drücken Sie F10.

  Oder:

  Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Ein anderes Objekt mit Step Into schrittweise ausführen

Wenn Sie den Debugger anweisen, ein anderes Objekt mit **Step Into** schrittweise auszuführen, wird der nächste Programm-Schritt ausgeführt, und die Trace-Position erscheint in der entsprechenden Sourcecode-Zeile. Wenn diese Sourcecode-Zeile ein weiteres Natural-Objekt aufruft oder aufnimmt, greift der Debugger auf dieses Objekt zu, und die Trace-Position erscheint in der ersten ausführbaren Zeile.

Anderes Objekt mit Step Into schrittweise ausführen

- Wählen Sie aus dem Menü **Debug** den Befehl **Step Into**.

  Oder:

  Drücken Sie F11.

  Oder:

  Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Anderes Objekt mit Step Out verlassen

Wenn Sie den Debugger anweisen, ein anderes Objekt mit **Step Out** zu verlassen, kehrt der Debugger zur vorigen Programmebene zurück. Der Debugger hält allerdings an, wenn ein Watchpoint oder Breakpoint gefunden wird, bevor diese vorige Ebene erreicht ist.

Dieser Befehl ist nützlich, wenn Sie ein Subprogramm austesten und mit der Ausführung des Rests des Subprogramms fortfahren möchten. Die Ausführung wird ohne Unterbrechung fortgesetzt und nach der Position in dem aufrufenden Programm angehalten, aus dem das Subprogramm aufgerufen wurde.

Anderes Objekt mit Step Out verlassen

- Wählen Sie aus dem Menü **Debug** den Befehl **Step Out**.

  Oder:

  Drücken Sie STRG+F11.

  Oder:

Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Zum nächsten Breakpoint oder Watchpoint gehen

Sie können den Debugger anweisen, das Objekt auszuführen, bis der nächste aktive Breakpoint gefunden ist, oder bis eine Watchpoint-Bedingung wahr wird. In diesem Fall hält der Debugger am Watchpoint oder Breakpoint an, und die Trace-Position erscheint in der entsprechenden Sourcecode-Zeile.

Zum nächsten Watchpoint oder Breakpoint gehen

- Wählen Sie aus dem Menü **Debug** den Befehl **Go**.

  Oder:

  Drücken Sie F7.

  Oder:

  Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Zum nächsten Event gehen

In einer ereignisgesteuerten Anwendung können Sie den Debugger anweisen, das Objekt auszuführen, bis das nächste Event (Ereignis) an die Anwendung gesandt wird. Der Debugger wird allerdings angehalten, wenn ein aktiver Watchpoint oder Breakpoint auftritt, bevor das nächste Event gesandt wird.

Zum nächsten Event gehen

- Wählen Sie aus dem Menü **Debug** den Befehl **Go Until Next Event** (Weiter bis zum nächsten Ereignis).

Anmerkung: In einer nicht-ereignisgesteuerten Anwendung hat dieser Befehl denselben Effekt wie der Befehl **Go**.

Oder:

Drücken Sie ALT+F7.

Zur Cursor-Position gehen

Sie können den Debugger anweisen, das Objekt auszuführen, bis die Sourcecode-Zeile an der aktuellen Cursor-Position erreicht ist.

Zur Cursor-Position gehen

1 Stellen Sie den Cursor in die Sourcecode-Zeile, in der die Ausführung angehalten werden soll.

2 Rufen Sie das Kontextmenü im Editor auf, und wählen Sie den Befehl **Run to Cursor**.

Oder:

Drücken Sie STRG+F10.

Zum nächsten Statement gehen

Sie können den Debugger anweisen, Code zu überspringen und die Ausführung des Objekts ab der Sourcecode-Zeile wieder aufzunehmen, in die Sie den Cursor gestellt haben. Der übersprungene Code wird nicht ausgeführt.

Vorsicht: Abhängig vom zu überspringenden Code kann dieser Befehl zu unvorhergesehenen Ergebnissen führen. Benutzen Sie diesen Befehl mit Vorsicht.

Zum nächsten Statement gehen

1 Stellen Sie den Cursor in die Sourcecode-Zeile, in der Sie die Ausführung wieder aufnehmen möchten.

2 Rufen Sie das Kontextmenü im Editor auf, und wählen Sie **Set Next Statement** (Nächstes Statement setzen).

Anmerkung: Dieser Befehl ist nur für das Objekt verfügbar, das gerade verarbeitet wird.

6 Breakpoints und Watchpoints setzen

Über Breakpoints und Watchpoints

Zwei Typen von Einträgen können in einem Programm zu Debugging-Zwecken definiert werden:

- **Breakpoints**
  Ein Breakpoint ist ein Punkt, an dem während der Ausführung eines Natural-Objekts die Kontrolle an den Benutzer zurückgegeben wird.

  Wenn ein einzelnes Statement mehr als eine Zeile einnimmt, kann ein Breakpoint nur in der ersten Zeile dieses Statements gesetzt werden.

  Wenn Sie versehentlich versuchen, einen Breakpoint in einer nicht ausführbaren Zeile (zum Beispiel in einer Kommentarzeile) zu setzen, wird der Breakpoint automatisch in die nächste ausführbare Zeile verschoben.
- **Watchpoints**
  Wenn Sie Watchpoints benutzen, können Sie rasch unerwartete Änderungen an Natural-Variablen entdecken, die von Objekten verursacht werden, die Fehler enthalten.

  Standardmäßig werden Watchpoints benutzt, um den Debugger anzuweisen, die Ausführung von Natural-Objekten zu unterbrechen, wenn sich der Inhalt einer Variablen ändert. Sie können der Variablen beim Setzen eines Watchpoint aber auch einen bestimmten Wert zusammen mit einem Watchpoint-Operator mitgeben; in diesem Fall wird der Watchpoint nur dann aktiviert, wenn eine bestimmte Bedingung erfüllt wird.

  Eine Variable wird als geändert angesehen, wenn sich entweder ihr aktueller Wert von dem Wert unterscheidet, der aufgezeichnet wurde, als der Watchpoint zuletzt angestoßen wurde, oder wenn er sich vom Anfangswert unterscheidet.

Jeder Breakpoint oder Watchpoint wird in dem betreffenden Register des **Break- und Watchpoints**-Fensters angezeigt. Für jeden Breakpoint wird die Zeilennummer angezeigt, in der der Breakpoint definiert wurde. Jeder Watchpoint erhält einen Namen, der dem Namen der Variablen entspricht, zu der er gehört; außerdem wird die Unterbrechungsbedingung (Break Condition) angezeigt.

Mittels des Kontrollkästchens in der ersten Spalte eines Registers kann ein Breakpoint oder Watchpoint während einer Debugging-Session jederzeit aktiviert oder deaktiviert werden. Siehe *Breakpoints und Watchpoints zeitweise deaktivieren*.

Für jeden Breakpoint oder Watchpoint wird die Anzahl der gefundenen Treffer (Hit Count) angezeigt, die sich jedes Mal erhöht, wenn der Debug-Eintrag durchlaufen wird. Die Anzahl der Ausführungen eines Debug-Eintrags kann allerdings auf eine der folgenden Arten eingeschränkt werden:

- Sie können die Anzahl der Einträge angeben, die übersprungen werden sollen (Skips), bevor der Breakpoint oder Watchpoint ausgeführt wird. Der Debug-Eintrag wird dann solange ignoriert, bis der Event-Zähler größer ist als die Anzahl der angegebenen zu überspringenden Einträge.
- Sie können die maximale Anzahl von Ausführungen angeben, so dass der Breakpoint oder Watchpoint ignoriert wird, sobald der Event-Zähler größer ist als die angegebene Anzahl der Ausführungen.

Wenn ein Breakpoint oder Watchpoint innerhalb eines anderen Objekts als des gerade aktiven Objekts vorgefunden wird, öffnet sich ein neues Editor-Fenster mit der Source dieses neuen Objekts.

Breakpoint hinzufügen und entfernen

Sie können im **Breakpoints**-Register des Break- und Watchpoints-Fenster einen Breakpoint für die aktuelle Cursor-Position hinzufügen. Oder wenn ein Breakpoint für diese Cursor-Position bereits vorhanden ist, können Sie ihn aus dem **Breakpoints**-Register entfernen.

Ein Dialogfeld erscheint in diesem Fall nicht. Wenn Sie den Breakpoint ändern möchten (zum Beispiel, um die maximale Anzahl der Ausführungen zu definieren), siehe ***Breakpoint ändern***.

Siehe auch ***Breakpoints und Watchpoints löschen***.

Anmerkung: In der lokalen Umgebung und in einer Remote-Umgebung (SPoD) ist es auch möglich, Breakpoints zu setzen und zu entfernen (wie unten beschrieben), wenn der Debugger nicht aktiv ist. Jeder Breakpoint, der im Editor definiert wurde, wird in diesem Fall geprüft, wenn der Debugger gestartet wird. Wenn der Breakpoint an der definierten Position nicht zulässig ist, wird er in die nächste Zeile verschoben, in der es möglich ist, einen Breakpoint zu definieren. Siehe auch *Debugger-Fenster* in der Dokumentation *Natural Studio benutzen*.

Breakpoint ein- oder ausschalten

1 Markieren Sie die Zeile, in der Sie einen Breakpoint setzen oder entfernen möchten.

2 Rufen Sie das Kontextmenü im Editor auf, und wählen Sie **Toggle Breakpoint** (Breakpoint ein-/ausschalten).

Oder:

Drücken Sie F9.

Oder:

Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Oder:

Am linken Rand des Editor-Fensters (wo gewöhnlich die Symbole für die Breakpoints erscheinen), klicken Sie eine Position neben der erforderlichen Zeile an.

Wenn ein Breakpoint gesetzt wird, erscheint jetzt ein Symbol am linken Rand des Editor-Fensters. Siehe *Im Editor-Fenster benutzte Symbole*. Ein Eintrag für den Breakpoint erscheint auch im **Breakpoints**-Register des Break- und Watchpoints-Fensters.

Wenn ein Breakpoint entfernt wurde, erscheint das betreffende Symbol nicht mehr. Der Eintrag für den Breakpoint wird aus dem **Breakpoints**-Register des Break- und Watchpoints-Fensters entfernt.

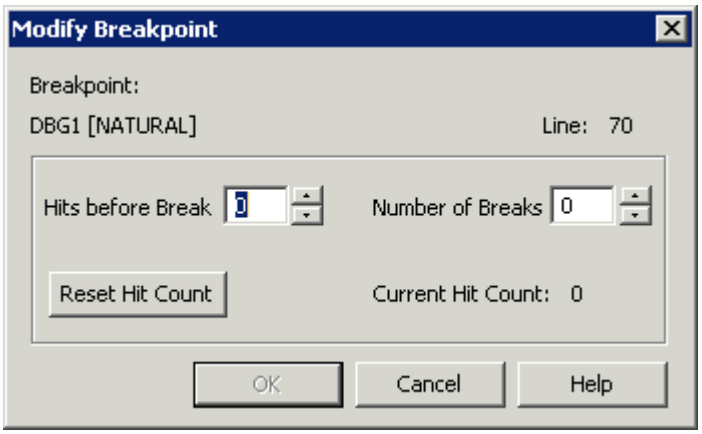
Breakpoint ändern

Sie können jeden Breakpoint ändern, der gerade im Break- und Watchpoints-Fenster angezeigt wird.

Breakpoint ändern

1 Markieren Sie den erforderlichen Breakpoint, rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Modify** (Ändern).

Es erscheint das folgende Dialogfeld:



2 Setzen Sie die erforderlichen Optionen:

Hits before Break (Treffer vor der Unterbrechung)
Die Anzahl der übersprungenen Einträge vor der Ausführung des Breakpoints, wenn er nicht ausgeführt werden soll, bis das Programm eine bestimmte Anzahl von Malen ausgeführt wurde. Die Voreinstellung ist 0.

Number of Breaks (Anzahl der Unterbrechungen)
Die maximale Anzahl der Ausführungen des Breakpoints. Nachdem diese Anzahl erreicht wurde, wird der Breakpoint ignoriert. Die Voreinstellung ist 0.

Reset Hit Count (Anzahl der gefundenen Treffer zurücksetzen)
Wenn Sie diese Befehlsschaltfläche wählen, wird die aktuelle Anzahl der Treffer (Hit Count) auf 0 zurückgesetzt.

3 Wählen Sie die Befehlsschaltfläche **OK**.

Watchpoint hinzufügen

Watchpoints erscheinen im **Watchpoints**-Register des Break- und Watchpoints-Fensters.

Sie können Watchpoints auf unterschiedliche Arten hinzufügen:

- Watchpoint im Editor-Fenster hinzufügen
- Watchpoint aus dem Variablen-Fenster hinzufügen
- Watchpoint über ein Dialogfeld hinzufügen

Watchpoint im Editor-Fenster hinzufügen

Sie können die Variable an der aktuellen Cursor-Position im Editor-Fenster zum **Watchpoints**-Register des Break- und Watchpoints-Fensters hinzufügen. Es erscheint kein Dialog in diesem Fall.

Variable als Watchpoint definieren

1 Markieren Sie die Variable im Editor, indem Sie den Cursor auf eine beliebige Position innerhalb des Variablennamens stellen.

2 Rufen Sie das Kontextmenü auf, und wählen Sie **Add to Watchpoints** (Zu Watchpoints hinzufügen).

Oder:

Drücken Sie STRG+UMSCHALT+W.

Oder:

Markieren Sie die Variable im Editor. Ziehen Sie die Variable mit der Maus auf das **Watchpoints**-Register und lassen Sie sie dort fallen.

Watchpoint aus dem Variablen-Fenster hinzufügen

Sie können eine Variable aus dem Variablen-Fenster zum **Watchpoints**-Register des Break- und Watchpoints-Fensters hinzufügen. Es erscheint kein Dialog in diesem Fall.

Variable als Watchpoint definieren

1 Markieren Sie die gewünschte Variable im Variablen-Fenster.

2 Rufen Sie das Kontextmenü auf, und wählen Sie **Add to Watchpoints** (Zu Watchpoints hinzufügen).

Oder:

Drücken Sie STRG+UMSCHALT+W.

Watchpoint über ein Dialogfeld hinzufügen

Sie können ein Dialogfeld benutzen, um einen Watchpoint zum **Watchpoints**-Register des Break- und Watchpoints-Fensters hinzuzufügen.

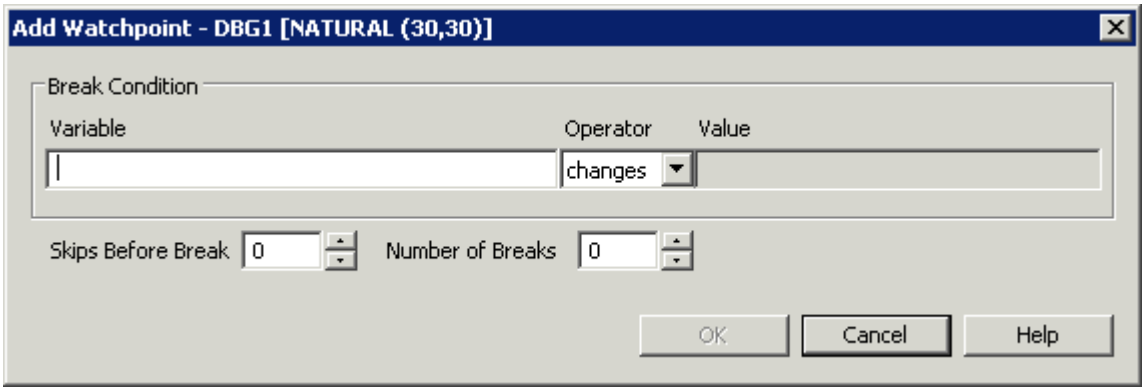
Watchpoint hinzufügen

1 Wählen Sie aus dem Menü **Debug** den Befehl **Add Watchpoint** (Watchpoint hinzufügen).

Oder:

Rufen Sie im **Watchpoints**-Register des Break- und Watchpoints-Fensters das Kontextmenü auf, und wählen Sie den Befehl **Add** (Hinzufügen). Stellen Sie sicher, dass kein anderer Eintrag markiert ist. Sonst erscheint dieser Befehl nicht im Kontextmenü.

Es erscheint das Dialogfeld **Add Watchpoint** (Watchpoint hinzufügen). Die Titelleiste gibt den Namen des aktuellen Programms und der aktuellen Library sowie die Datenbank-ID und Dateinnummer des aktuellen `FUSER` an.



2 Setzen Sie die erforderlichen Optionen:

Variable
Die Variable, die im ausgetesteten Programm überwacht werden soll.

Operator/Value (Operator/Wert)
Um eine Bedingung für den Watchpoint zu definieren, wählen Sie einen passenden Watchpoint-Operator aus, und geben Sie einen Wert für diesen Operator an. Wenn Sie keine Bedingung angeben, gilt die Voreinstellung ("changes").

Die Watchpoint-Operatoren sind Folgende:

Operator	Aktivierung des Watchpoints
changes (Änderungen)	Jedesmal, wenn die Variable geändert wird. Voreinstellung.
EQ (=)	Nur wenn der aktuelle Wert der Variablen gleich dem angegebenen Wert ist.
NE (!=)	Nur wenn der aktuelle Wert der Variablen ungleich dem angegebenen Wert ist.
GT (>)	Nur wenn der aktuelle Wert der Variablen größer als der angegebene Wert ist.
LT (<)	Nur wenn der aktuelle Wert der Variablen kleiner als der angegebene Wert ist.
GE (>=)	Nur wenn der aktuelle Wert der Variablen größer gleich dem angegebenen Wert ist.
LE (<=)	Nur wenn der aktuelle Wert der Variablen kleiner gleich dem angegebenen Wert ist.

Skips before Break (Übersprungene Einträge vor der Unterbrechung)
Die Anzahl der übersprungenen Einträge vor der Ausführung des Watchpoints, wenn er erst dann ausgeführt werden soll, wenn das Programm eine gewisse Anzahl von Malen abgelaufen ist. Die Voreinstellung ist 0.

Number of Breaks (Anzahl der Unterbrechungen)
Die maximale Anzahl der Ausführungen des Watchpoints. Nachdem diese Anzahl erreicht wurde, wird der Watchpoint ignoriert. Die Voreinstellung ist 0.

3 Wählen Sie die Befehlsschaltfläche **OK**.

Der Name der ausgewählten Variablen wird jetzt im **Watchpoints**-Register des Break- und Watchpoints-Fensters angezeigt.

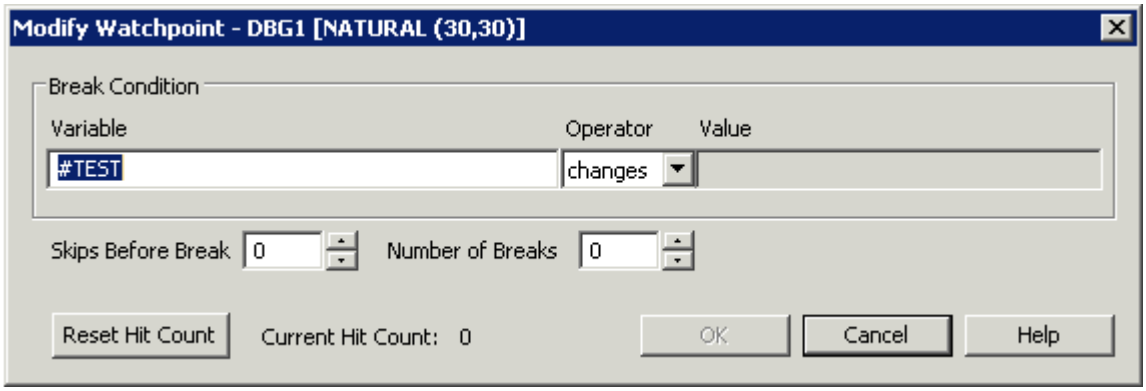
Watchpoint ändern

Sie können jeden Watchpoint ändern, der im Break- und Watchpoints-Fenster angezeigt wird.

Watchpoint ändern

1 Markieren Sie den erforderlichen Watchpoint, rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Modify** (Ändern).

Es erscheint das Dialogfeld **Modify Watchpoint** (Watchpoint ändern).



Dieses Dialogfeld bietet dieselben Optionen wie das Dialogfeld **Add Watchpoint** (Watchpoint hinzufügen). Eine Beschreibung der Optionen, die in diesem Dialogfeld angegeben werden können, finden Sie im Abschnitt *Watchpoint über ein Dialogfelds hinzufügen*.

Außerdem bietet dieses Dialogfeld die Befehlsschaltfläche **Reset Hit Count** (Anzahl der Treffer zurücksetzen). Wenn Sie diese Befehlsschaltfläche wählen, wird die aktuelle Anzahl der Treffer (Hit Count) auf 0 zurückgesetzt.

2 Machen Sie alle erforderlichen Änderungen, und wählen Sie die Befehlsschaltfläche **OK**.

Breakpoints und Watchpoints zeitweise deaktivieren

Jeder definierte Breakpoint oder Watchpoint kann zeitweise deaktiviert werden.

Breakpoint oder Watchpoint deaktivieren

1 Wählen Sie das erforderliche Register im Break- und Watchpoints-Fenster aus.

2 Wählen Sie für den gewünschten Eintrag die erste Spalte des Registers aus, um die Markierung zu entfernen.

Oder:

Rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Activate/Deactivate** (Aktivieren/Deaktivieren).

Wenn Sie einen Breakpoint deaktiviert haben, ändert sich das Symbol, das am linken Rand des Editor-Fensters angezeigt wird. Siehe *Im Editor-Fenster benutzte Symbole*.

Deaktivierten Breakpoint oder Watchpoint aktivieren

1 Wählen Sie das erforderliche Register im Break- und Watchpoints-Fenster aus.

2 Wählen Sie für den gewünschten Eintrag die erste Spalte des Registers aus, so dass erneut eine Markierung erscheint.

Oder:

Rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Activate/Deactivate** (Aktivieren/Deaktivieren).

Wenn Sie einen Breakpoint aktiviert haben, ändert sich das Symbol, das am linken Rand des Editor-Fensters angezeigt wird. Siehe *Im Editor-Fenster benutzte Symbole*.

Alle Breakpoints oder Watchpoints deaktivieren oder aktivieren

1 Wählen Sie das erforderliche Register im Break- und Watchpoints-Fenster aus.

2 Stellen Sie sicher, dass kein Eintrag markiert ist (sonst wird der erforderliche Befehl nicht im Kontextmenü angezeigt), rufen Sie das Kontextmenü auf, und wählen Sie entweder den Befehl **Deactivate All** (Alle deaktivieren) oder **Activate All** (Alle aktivieren).

Wenn Sie alle Breakpoints deaktiviert oder aktiviert haben, ändern sich die Symbole, die am linken Rand des Editor-Fensters angezeigt werden. Siehe *Im Editor-Fenster benutzte Symbole*.

Sourcecode für einen definierten Breakpoint oder Watchpoint zeigen

Für jeden definierten Breakpoint oder Watchpoint, der im Break- und Watchpoints-Fenster angezeigt wird (ganz gleich, ob er aktiv ist oder nicht), können Sie zur Source gehen, in der dieser Breakpoint oder Watchpoint definiert wurde.

Zur Source gehen, in der ein Breakpoint oder Watchpoint definiert wurde

1 Wählen Sie das erforderliche Register im Break- und Watchpoints-Fenster aus.

2 Markieren Sie den erforderlichen Eintrag, rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Go To Source Code** (Zum Sourcecode gehen).

Oder:

Klicken Sie den erforderlichen Eintrag doppelt an.

Für einen Breakpoint erscheint die Trace-Position neben der Sourcecode-Zeile, in der der Breakpoint definiert wurde.

Für einen Watchpoint erscheint der gesamte Sourcecode, in dem der Watchpoint definiert wurde.

Breakpoints und Watchpoints löschen

Sie können entweder markierte Breakpoints oder Watchpoints löschen, oder Sie können alle Breakpoints oder Watchpoints löschen.

Siehe auch *Breakpoint hinzufügen oder entfernen*.

Breakpoint oder Watchpoint löschen

1 Wählen Sie das erforderliche Register im Break- und Watchpoints-Fenster aus.

2 Markieren Sie den erforderlichen Eintrag.

3 Rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Delete** (Löschen).

Oder:

Drücken Sie ENTF.

Alle Breakpoints oder Watchpoints löschen

1 Wählen Sie das erforderliche Register im Break- und Watchpoints-Fenster aus.

2 Stellen Sie sicher, dass kein Eintrag markiert ist (sonst wird der erforderliche Befehl nicht im Kontextmenü angezeigt), rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Delete All** (Alle löschen).

Im Editor-Fenster benutzte Symbole

Die folgenden Symbole können am linken Rand des Editor-Fensters erscheinen.

 Diese Zeile enthält einen aktiven Breakpoint.

 Diese Zeile enthält einen deaktivierten Breakpoint.

 Diese Zeile enthält einen aktiven Breakpoint. Sie enthält auch die Trace-Position.

 Diese Zeile enthält einen deaktivierten Breakpoint. Sie enthält auch die Trace-Position.

 Diese Zeile enthält einen Breakpoint, der noch nicht validiert wurde (d.h. der Debugger hat noch nicht die markierte Zeile erreicht). Der Status kann entweder als aktiv (roter Hintergrund) oder inaktiv (weißer Hintergrund) anzeigt werden.

 Diese Zeile enthält einen ungültigen Breakpoint (zum Beispiel, wenn der Breakpoint in einer Zeile nach dem `END`- Statement gesetzt wurde). Der Status kann entweder als aktiv (roter Hintergrund) oder inaktiv (weißer Hintergrund) angezeigt werden.

7 Variablen ändern und überwachen

Variable ändern

Sie können eine Variable auf unterschiedliche Arten ändern:

- Variable im Editor-Fenster ändern
- Variable im Variablen-Fenster ändern

Variable im Editor-Fenster ändern

Sie können die Variable ändern, die an der aktuellen Cursor-Position im Editor-Fenster angezeigt wird. Im daraufhin erscheinenden Dialogfeld können Sie auch den Namen einer anderen zu ändernden Variablen eingeben.

Variable an der Cursor-Position ändern

1 Markieren Sie die Variable im Editor aus, indem Sie den Cursor an eine beliebige Position innerhalb des Variablennamens stellen.

2 Rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Modify Variable** (Variable ändern).

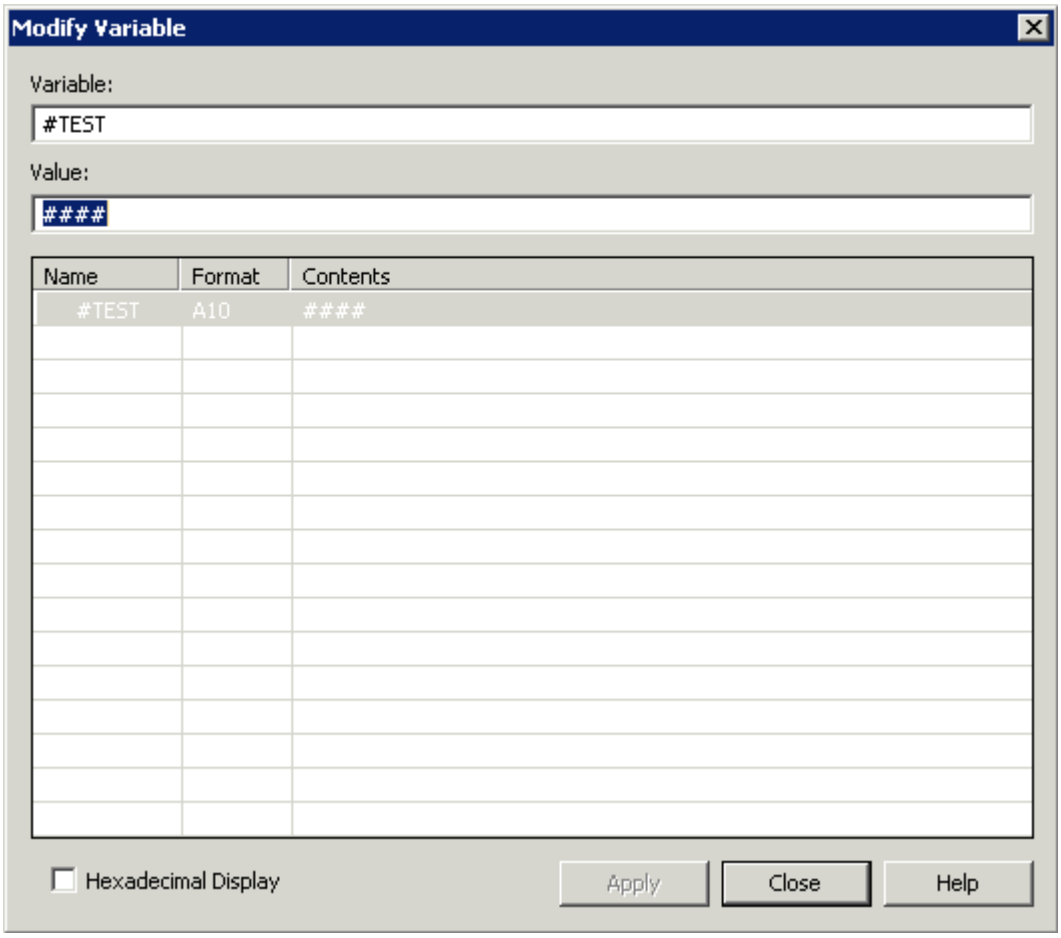
Oder:

Drücken Sie UMSCHALT+F9.

Oder:

Wenn die Debug-Symbolleiste angezeigt wird, wählen Sie die folgende Schaltfläche in der Symbolleiste:

Es erscheint das Dialogfeld **Modify Variable** (Variable ändern) mit dem Inhalt der markierten Variable. Im Falle eines Arrays ist der Knoten standardmäßig erweitert.



3 Um den Inhalt einer Variablen zu ändern, geben Sie den neuen Wert in das Textfeld **Value** (Wert) ein.

Anmerkung: Bei Variablen, die nicht geändert werden können (wie z.B. nicht änderbare Systemvariablen), wird das Textfeld **Value** grau dargestellt.

Sie können das Dialogfeld **Modify Variable** auch folgendermaßen benutzen:

- Wenn Sie den Namen einer anderen bereits vorhandenen Variablen in das Textfeld **Variable** eingeben, erscheint der Inhalt dieser Variablen sofort im Dialogfeld, und Sie können sie ändern.
- Im Falle eines Arrays können Sie die Ausprägungen dieses Arrays wie folgt ändern:
  - Derselbe Wert kann für alle Ausprägungen definiert werden: Markieren Sie den obersten Knoten und geben Sie den neuen Wert ein.
  - Jede Ausprägung kann separat geändert werden: Markieren Sie die erforderliche Ausprägung, und geben Sie den neuen Wert ein.
  - Bestimmte Ausprägungen können gleichzeitig geändert werden: Geben Sie die erforderlichen Ausprägungen im Textfeld **Variable** ein. Wenn Sie beispielsweise `#MYVAR(1:6)`

in `#MYVAR(2:3)` abändern, werden nur die zweite und die dritte Ausprägung im Dialogfeld angezeigt. Wenn Sie einen neuen Wert eingeben, gilt er nur für diese Ausprägungen.

Variablen (und Ausprägungen), die Sie geändert haben, sind Rot gekennzeichnet.

4 Wenn Sie das Kontrollkästchen **Hexadecimal Display** (Hexadezimale Anzeige) aktivieren, wird der Inhalt der Variablen in hexadezimalem Format angezeigt.

5 Wählen Sie die Befehlsschaltfläche **Apply** (Zuweisen).

Ihre Änderungen werden sofort gespeichert, wenn Sie die Befehlsschaltfläche **Apply** wählen. Sie werden aber noch nicht im Variablen-Fenster angezeigt.

6 Um das Dialogfeld zu schließen, wählen Sie die Befehlsschaltfläche **Close** (Schließen).

Ihre Änderungen werden jetzt im Variablen-Fenster angezeigt.

Variable im Variablen-Fenster ändern

Sie können eine im Variablen-Fenster aufgeführte Variable ändern.

Es ist nicht möglich, einen Eintrag zu ändern, der noch erweitert werden kann (wie z.B. ein View). Dies ist nur möglich für einzelne Variablen, nachdem der Eintrag erweitert wurde.

Die folgenden Farben werden für die Einträge im Variablen-Fenster benutzt:

- **Grau**
  Variablen, die nicht geändert werden können (wie z.B. nicht änderbare Systemvariablen) sind in Grau dargestellt.
- **Rot**
  Variablen, die Sie geändert haben, sind in Rot dargestellt.

Variable im Variablen-Fenster ändern

1 Wählen Sie das erforderliche Register im Variablen-Fenster aus.

2 Markieren Sie den erforderlichen Eintrag.

3 Rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Modify** (Ändern).

Es erscheint das Dialogfeld **Modify Variable** (Variable ändern) mit dem Inhalt der markierten Variablen.

Weitere Informationen zu diesem Dialogfeld finden Sie unter *Variable im Editor-Fenster ändern*.

Watch-Variable hinzufügen

Wenn Sie bestimmte Variablen überwachen möchten, können Sie sie zum **Watch**-Register des **Variablen-Fensters** hinzufügen.

Sie können eine Watch-Variable auf unterschiedliche Arten hinzufügen:

- Watch-Variable aus dem Editor-Fenster hinzufügen
- Watch-Variable aus dem Variablen-Fenster hinzufügen

Anmerkung: Es ist nicht möglich, den Inhalt einer Watch-Variable zu ändern.

Watch-Variable aus dem Editor-Fenster hinzufügen

Sie können die Variable an der aktuellen Cursor-Position im Editor-Fenster als eine Watch-Variable definieren.

Watch-Variable im Variablen-Fenster hinzufügen

1 Markieren Sie die Variable im Editor, indem Sie den Cursor auf eine beliebige Position im Namen stellen.

2 Rufen Sie das Kontextmenü im Editor auf, und wählen Sie den Befehl **Add to Watchvariables** (Zu Watch-Variablen hinzufügen).

Oder:

Drücken Sie STRG+UMSCHALT+T.

Oder:

Markieren Sie die Variable im Editor. Ziehen Sie die Variable mit der Maus auf das **Watch**-Register des Variablen-Fensters und lassen Sie sie dort fallen.

Watch-Variable aus dem Variablen-Fenster hinzufügen

Sie können Variablen von den ersten Registern des Variablen-Fensters zum Watch-Register desselben Fensters hinzufügen.

Variable als Watch-Variable definieren

1 Markieren Sie die gewünschte Variable im Variablen-Fenster.

2 Rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Add to Watchvariables** (Zu Watch-Variablen hinzufügen).

Oder:

Drücken Sie STRG+UMSCHALT+T.

Variablen im Variablen-Fenster verwalten

Die folgenden Themen werden nachstehend behandelt:

- Letzte geänderte Variable zeigen
- Variable suchen
- Inhalt einer Variablen in alphanumerischem oder hexadezimalem Format zeigen
- Anzeige aktualisieren
- Watch-Variablen löschen

Siehe auch *Watchpoint aus dem Variablen-Fenster hinzufügen*.

Letzte geänderte Variable zeigen

Sie können festlegen, dass die beim Austesten geänderten Variablen stets im Variablen-Fenster sichtbar sind. Dies ist hilfreich, wenn Sie ein Programm austesten, das mehr Variablen hat als im Variablen-Fenster auf einmal angezeigt werden können. Wenn eine Variable beim Austesten geändert wird, die zur Zeit nicht im Variablen-Fenster sichtbar ist, wird im Variablen-Fenster ein Bildlauf durchgeführt, so dass die geänderte Variable zu sehen ist.

Diese Funktion ein- und ausschalten

1 Wählen Sie ein beliebiges Register im Variablen-Fenster.

2 Stellen Sie sicher, dass kein Eintrag im Register markiert wird (sonst wird der erforderliche Befehl nicht im Kontextmenü angezeigt), rufen Sie das Kontextmenü auf und wählen Sie **Show Last Modified** (Letzten geänderten Eintrag zeigen).

Wenn diese Funktion aktiv ist, wird ein Häkchen neben diesem Menübefehl angezeigt.

Variable suchen

Wenn das Variablen-Fenster aktiv ist, können Sie im aktuell ausgewählten Register nach einer Variablen suchen.

Anmerkung: Wenn ein Knoten im Variablen-Fenster nicht erweitert ist, wird sein Inhalt nicht in der Suche mit berücksichtigt.

Variable suchen

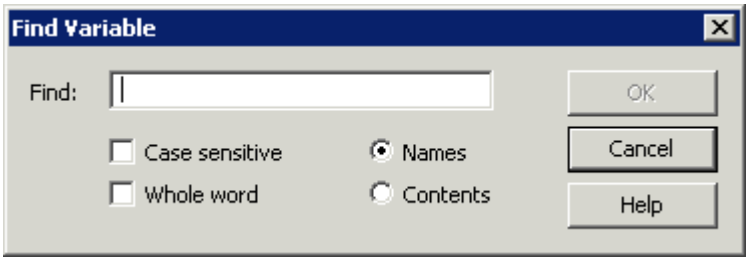
1 Wählen Sie das erforderliche Register im Variablen-Fenster aus.

2 Drücken Sie STRG+F.

Oder:

Stellen Sie sicher, dass kein Eintrag im Register markiert ist (sonst wird der erforderliche Befehl nicht im Kontextmenü angezeigt), rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Find** (Suchen).

Es erscheint das Dialogfeld **Find Variable** (Variable suchen).



3 Geben Sie Ihre Suchkriterien an:

Option	Beschreibung
Find (Suchen)	Die zu suchende Zeichenkette.
Case sensitive (Klein- oder Großschreibung)	Wenn dieses Kontrollkästchen markiert ist, werden nur Zeichenketten gefunden, die genau mit dem Eintrag im Textfeld **Find** übereinstimmen. Wenn es nicht markiert ist, werden alle Kombinationen von Groß- und Kleinbuchstaben gefunden.
Whole word (Ganzes Wort)	Wenn dieses Kontrollkästchen markiert ist, wird die Suche auf ganze Wörter eingeschränkt. Wenn es nicht markiert ist, werden alle Vorkommen der Zeichenkette gefunden.
Names (Namen)	Wenn dieses Optionsfeld markiert ist, bezieht sich die Zeichenkette im Textfeld **Find** auf einen Variablennamen.
Contents (Inhalt)	Wenn dieses Optionsfeld markiert ist, bezieht sich die Zeichenkette im Textfeld **Find** auf den Inhalt einer Variablen. Das heißt: Sie möchten eine Variable suchen, die den angegebenen Inhalt enthält.

4 Wählen Sie die Befehlsschaltfläche **OK**.

Wenn eine Variable, die den angegebenen Kriterien entspricht, im aktuellen Register gefunden werden kann, wird ihr Name hervorgehoben.

Anmerkung: Eine Meldung wird kurz angezeigt, die angibt, ob der angegebene Text gefunden wurde oder nicht.

Nächste Variable mit den angegebenen Suchkriterien suchen

- Drücken Sie F3.

  Oder:

  Stellen Sie sicher, dass kein Eintrag im Register markiert ist (sonst wird der erforderliche Befehl nicht im Kontextmenü angezeigt), rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Find Next** (weitersuchen).

Inhalt einer Variablen in alphanumerischem oder hexadezimalem Format zeigen

Sie können festlegen, ob der Inhalt der Variablen im Variablen-Fenster in alphanumerischem oder in hexadezimalem Format angezeigt werden soll.

Format umschalten

1. Wählen Sie das erforderliche Register im Variablen-Fenster.
2. Stellen Sie sicher, dass kein Eintrag im Register markiert ist (sonst wird der erforderliche Befehl nicht im Kontextmenü angezeigt), rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Hexadecimal Display** (Hexadezimale Anzeige).

   Wenn der Wert in der Spalte **Contents** (Inhalt) vorher in alphanumerischem Format angezeigt wurde, dann wird er jetzt in hexadezimalem Format angezeigt, und umgekehrt.

   Wenn das hexadezimale Format benutzt wird, wird ein Häkchen neben diesem Menübefehl angezeigt.

Anzeige aktualisieren

Wenn sich etwas in Natural Studio ändert, wird die Anzeige gewöhnlich automatisch aktualisiert. Im Debugger geschieht dies, wenn sich der Inhalt einer Variablen ändert. Für diese automatische Aktualisierung ist es erforderlich, dass die entsprechende Option in den Workspace-Optionen gesetzt wurde.

Wenn die automatische Aktualisierung in den Workspace-Optionen deaktiviert wurde, und sich der Inhalt einer oder mehrerer Variablen im aktuell ausgewählten Register ändert, müssen Sie die Anzeige manuell aktualisieren, damit Sie die aktuellen Werte sehen können.

Es gibt eine Ausnahme: Watch-Variablen werden stets automatisch aktualisiert, unabhängig von der Einstellung in den Workspace-Optionen.

Anzeige manuell aktualisieren

1 Wählen Sie ein beliebiges Register im Variablen-Fenster aus (außer dem **Watch**-Register).

2 Drücken Sie F5.

Oder:

Stellen Sie sicher, dass kein Eintrag im Register markiert ist (sonst wird der erforderliche Befehl nicht im Kontextmenü angezeigt), rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Refresh** (Aktualisieren).

Watch-Variablen löschen

Sie können entweder die markierten Watch-Variablen oder alle Watch-Variablen im Variablen-Fenster löschen.

Watch-Variable löschen

1 Markieren Sie die gewünschte Watch-Variable im **Watch**-Register des Variablen-Fensters.

2 Rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Delete** (Löschen).

Oder:

Drücken Sie ENTF.

Alle Watch-Variablen löschen

- Stellen Sie sicher, dass kein Eintrag im **Watch**-Register des Variablen-Fensters markiert ist (sonst wird der erforderliche Befehl nicht im Kontextmenü angezeigt), rufen Sie das Kontextmenü auf, und wählen Sie den Befehl **Delete All** (Alle Löschen).

8 Call-Stack benutzen

Über den Call-Stack

Im **Call-Stack-Fenster** sind die Objekte in hierarchischer Reihenfolge aufgeführt, die während der aktuellen Debugging-Session aufgerufen wurden.

Das neueste Objekt erscheint stets oben in der Liste. Im **Variablen-Fenster** werden standardmäßig alle Variablen angezeigt, die zu diesem Objekt gehören. Wenn Sie beispielsweise ein Subprogramm mit **Step Into** ausführen, erscheint dieses Subprogramm ganz oben in der Liste, und im Variablen-Fenster werden automatisch die Variablen für dieses Subprogramm angezeigt.

Sie können das Editor-Fenster für ein bestimmtes Objekt in den Vordergrund holen, indem Sie den entsprechenden Eintrag im Call-Stack-Fenster doppelt anklicken.

Anmerkungen:

1. Ein grauer (und nicht blauer) Pfeil im Editor-Fenster zeigt die Position an, an der das vorige Objekt in der Call-Stack-Hierarchie aufgerufen wurde.
2. Wenn Copycode ausgetestet wird, enthält der Call-Stack keinen zusätzlichen Eintrag für diesen Copycode.

Sourcecode eines anderen Objekts anzeigen

Für jedes im Call-Stack aufgeführte Objekt können Sie den Sourcecode anzeigen und somit sein Editor-Fenster in den Vordergrund holen. Zu diesem Zweck gibt es unterschiedliche Befehle:

- **Go To Source Code (Zum Sourcecode gehen)**
  Wenn Sie diesen Befehl wählen, werden die Variablen für das Objekt im aktivierten Editor-Fenster nicht im Variablen-Fenster berücksichtigt. Es enthält noch die Variablen des vorher aufgerufenen Objekts.
- **Switch To Call Level (Zur Aufrufebene wechseln)**
  Wenn Sie diesen Befehl wählen, erscheinen die Variablen für das Objekt im aktivierten Editor-Fenster im Variablen-Fenster.

Zum Sourcecode eines anderen Objekts wechseln

- Markieren Sie im Call-Stack das Objekt, für das Sie den Sourcecode anzeigen möchten, und wählen Sie im Kontextmenü den Befehl **Go To Source Code**.

  Das Editor-Fenster für dieses Objekt wird aktiviert.

Zum Sourcecode eines anderen Objekts wechseln und die Variablen dieses Objekts anzeigen

- Markieren Sie im Call-Stack das Objekt, für das Sie den Sourcecode anzeigen möchten, und wählen Sie im Kontextmenü den Befehl **Switch To Call Level**.

  Das Editor-Fenster für dieses Objekt wird aktiviert. Der Inhalt des Variablen-Fensters ändert sich; es erscheinen jetzt die Variablen dieses Objekts.

Zum Objekt an der aktuellen Trace-Position zurückkehren

Wenn Sie den Sourcecode eines anderen Objekts angezeigt haben, können Sie zum Objekt an der aktuellen (durch einen blauen Pfeil angezeigten) Trace-Position zurückkehren, und somit sein Editor-Fenster in den Vordergrund holen.

Zum Objekt an der aktuellen Trace-Position zurückkehren

- Wählen Sie aus dem Menü **Debug** den Befehl **Show Trace Position**.

  Anmerkung: Siehe auch *Trace-Position im Editor-Fenster*.

  Das die aktuelle Trace-Position enthaltende Editor-Fenster wird aktiviert. Der Inhalt des Variablen-Fensters ändert sich; es erscheinen jetzt die Variablen dieses Objekts.

9 Alten Debugger benutzen

Der alte Debugger kann erscheinen, wenn eine alte Version von Natural auf dem Entwicklungs-Server installiert ist; siehe *Allgemeine Informationen*.

Natural-Objekte vorbereiten

Um den vollständigen Funktionalitätsbereich des Natural Debuggers auszunutzen, müssen Sie den folgenden Natural-Profilparameter entweder dynamisch oder in Ihrer Natural-Parameterdatei angeben:

`SYMGEN` ist auf "ON" gesetzt

Wenn ein Objekt katalogisiert oder in Source- und Objekt-Form gespeichert wird und `SYMGEN` auf "ON" gesetzt ist, wird eine Symboltabelle als Bestandteil des generierten Programms generiert. Da diese Tabelle die Informationen enthält, die sich auf die für dieses Objekt aktiven Variablen beziehen, können Variablen nicht aufgerufen werden, ohne dass `SYMGEN` angegeben wird, obwohl es noch möglich ist, das Objekt auszutesten.

Debugger starten

Der Debugger kann nur auf in Source- und Objekt-Form gespeicherte oder katalogisierte Natural-Programme und Dialoge angewandt werden.

Debugger starten

- Geben Sie das folgende Natural-Kommando ein:

  ```
  DEBUG objektname
  ```

  wobei *objektname* der Name des Natural-Objekts ist, das Sie austesten möchten.

In der Titelleiste erscheint einer der folgenden Einträge:

- **[break]**
  Wenn "[break]" in der Titelleiste erscheint, hat der Debugger die Kontrolle.
- **[waiting]**
  Wenn "[waiting]" in der Titelleiste erscheint, hat die gerade ausgetestete Natural-Anwendung die Kontrolle.

Wenn der Remote-Debugger im Betriebssystem Windows aktiv wird, erscheinen die folgenden Informationen in der Titelleiste: "Debugging Remote Natural Client (*knotenname*::*benutzername*::*prozess-id*)", wobei *knotenname* der Name des Computers ist, auf dem Natural läuft, *benutzername* der Name des Natural-Benutzers ist, und *prozess-id* die Natural Prozess-ID ist.

Das Debugger-Fenster enthält ein Child-Fenster mit einem Source-Listing des angegebenen Objekts, das ausgetestet werden soll.

In Verbindung mit dieser Objekt-Source werden die folgenden Informationen mittels Funktionsleisten angezeigt:

Funktionsleiste	Funktion
Breakpoints and Watchpoints	Diese Funktionsleiste besteht aus zwei Register-Bereichen. Ein Bereich verwaltet Breakpoints, während der andere Watchpoints verwaltet.
Variables	Diese Funktionsleiste zeigt die aktiven Variablen und ihren eigentlichen Inhalt an. Diese Variablen werden unter den folgenden Kategorien angezeigt: Locals, Globals, Systems, AIVs und Contexts (Lokale Variablen, Globale Variablen, Systemvariablen, AIV-Variablen und Kontext-Variablen).
Watchvariables	Diese Funktionsleiste zeigt die vom Benutzer ausgewählten Variablen einer Kategorie an, die in der Funktionsleiste Variables zur Verfügung steht.

Die einzelnen Funktionsleisten sind in allen Einzelheiten im Rest dieses Abschnitts beschrieben.

Debugger verlassen

Sie können den Debugger von einem beliebigen Punkt innerhalb einer Anwendung verlassen, indem Sie entweder Exit (siehe unten) oder die entsprechende Schaltfläche in der Symbolleiste wählen.

Der Debugger wird auch beendet, wenn die Anwendung ohne einen Fehler beendet wird, dann wird der Trace-Cursor (Cursor zur Ablaufverfolgung) in die zuletzt ausgeführte Sourcecode-Zeile platziert.

Im Falle eines Fehlers wird die entsprechende Source im Source-Fenster angezeigt, und der Trace-Cursor (Cursor zur Ablaufverfolgung) wird in die Zeile platziert, die den Fehler verursachte. Es erscheint ein Meldungsfenster mit der passenden Fehlermeldung und der Auswahlmöglichkeit, entweder mit der Verarbeitung fortzufahren oder die Debugging-Session zu beenden. Das Fortsetzen der Debugging-Session kann nützlich sein, wenn zum Beispiel Folgendes vorliegt:

- ihre Anwendung enthält eine beliebige Fehlerverarbeitung (einschließlich Fehlertransaktionen);
- Sie möchten beliebige Variablen anzeigen, bevor Sie Ihre Debugging-Session beenden.

Wenn Sie den Debugger verlassen, werden Ihre Breakpoint-, Watchpoint- und Watchvariablen-Einstellungen automatisch gespeichert, und zwar zusammen mit den Fenster- und Symbolleisten-Einstellungen. Alle diese Einstellungen werden das nächste Mal wiederhergestellt, wenn Sie den Debugger nochmals aufrufen.

Anmerkung: Wenn Sie im Falle einer Remote-Entwicklung den Debugger in einem Remote-System verlassen, wird die Programmausführung fortgesetzt, aber die Debugging-Kontrolle der Programmausführung wird angehalten.

Befehl Exit (Verlassen)

Mit **Exit** verlassen Sie die Debugging-Session, und die Kontrolle wird an Natural zurückgegeben. Der Befehl **Exit** (Verlassen) steht im ersten Menü im Hauptfenster für jede der fünf Debugger-Hauptfunktionen zur Verfügung.

Debugger bedienen

Bevor das Source-Fenster des Debuggers und andere Haupt-Funktionen detailliert behandelt werden, bietet Ihnen dieser Abschnitt allgemeine Informationen zum Debugger.

Windows und Menüs

Der Debugger bietet verschiedene Fenster, Funktionsleisten, Symbolleisten und Menüs.

Menübefehle, von denen angenommen wird, dass sie sehr oft benutzt werden, stehen auch als **Schaltflächen** in den betreffenden Symbolleisten zur Verfügung.

Anstatt die Menüs zu benutzen, können Sie Schaltflächen in der Symbolleiste wählen oder **Tastenkombinationen** benutzen.

Im Gegensatz zu Natural selbst:

- hat der Debugger keine Kommandozeile.
- enthält das **Tools**-Menü des Debuggers die folgenden Optionen:
  - **Customize** (Anpassen) - diese Option ermöglicht es Ihnen, das Aussehen Ihres Menüs und Ihrer Symbolleiste zu ändern, sowie Tastenkombinationen für häufig benutzte Befehle zu definieren;
  - **Fonts** (Schriftarten) - diese Option ermöglicht es Ihnen, die Schriftart des Source-Fensters zu ändern;
  - **Warning messages** (Warnungsmeldungen) - diese ermöglichen es Ihnen, festzulegen, ob Warnungsmeldungen zu fehlendem Sourcecode oder symbolische Informationen angezeigt

werden sollen oder nicht. Eine Meldung, die Sie darüber informiert, ob der gerade angezeigte Sourcecode neuer als das betreffende generierte Programm ist und auch davon beeinträchtigt ist.

Schaltflächen in der Symbolleiste

Die Symbolleisten bieten Ihnen einen schnellen Zugriff auf häufig benutzte Befehle. Um eine Kurzbeschreibung eines Befehls anzuzeigen, platzieren Sie den Mauszeiger auf die betreffende Schaltfläche. Die Beschreibung erscheint in der Statusleiste am unteren Rand des Hauptfensters des Debuggers. Wenn ein Befehl gerade nicht angewandt werden kann, wird die Schaltfläche ausgeschaltet.

Tastenkombinationen

Eine weitere Möglichkeit, einen Debugger-Befehl auszuführen, besteht darin, eine entsprechende Tastenkombination über die Tastatur einzugeben. Standardmäßig sind die folgenden Tastenkombinationen definiert:

Menü	Tastenkombination	Funktion
File (Datei)	STRG+O	Open (Öffnen)
Edit (Editieren)	STRG+F	Find (Suchen)
	F3	Find Next (Suchen Nächstes)
Debug (Debug)	F4	Close (Schließen)
	F5	Go (Weiter)
	F6	Step Over (Objekt mit Step Over überspringen)
	F7	Step In (Objekt mit Step In schrittweise ausführen)
	STRG+F7	Step Out (Objekt mit Step Out verlassen)
	STRG+F6	Run To Cursor (Zum Cursor)
	Alt+*	Show Trace Position (Position zur Ablaufverfolgung zeigen)
	F9	Toggle Breakpoint (Breakpoint ein/ausschalten)
Variables (Variablen)	STRG+M	Modify Variable (Variable ändern)
	STRG+D	Display Variable (Variable anzeigen)
	STRG+V	Add to Watch-Variablen (Zu Watch-Variablen hinzufügen)
	STRG+W	Add to Watchpoints (Zu Watchpoints hinzufügen)

Watchpoints und Breakpoints

Zwei Typen von Einträgen können in einem Programm zu Debugging-Zwecken definiert werden: **Watchpoints** und **Breakpoints**. Jeder Watchpoint oder Breakpoint wird in seiner entsprechenden Funktionsleiste angezeigt. Für jeden Watchpoint wird ein Name zugewiesen, der dem Namen der Variable entspricht, zu dem er gehört.

Jeder Watchpoint oder Breakpoint kann jederzeit während einer Debugging-Session über die entsprechenden Kontrollkästchen aktiviert oder deaktiviert werden.

Für jeden Watchpoint oder Breakpoint wird die Anzahl der Events (Ereignisse) angegeben, die sich jedes Mal erhöht, wenn der Debug-Eintrag überschritten wird. Die Anzahl der Ausführungen eines Debug-Eintrags kann aber auf zwei Arten eingeschränkt werden:

1. Eine Anzahl von übersprungenen Einträgen kann angegeben werden, bevor der Watchpoint oder Breakpoint ausgeführt wird. Der Debug-Eintrag wird dann ignoriert, bis die Anzahl der Events (Ereignisse) größer als die Anzahl der angegebenen übersprungenen Einträge ist.
2. Eine maximale Anzahl von Ausführungen kann angegeben werden, so dass der Breakpoint oder Watchpoint ignoriert wird, sobald die Anzahl der gefundenen Events (Ereignisse) die angegebene Anzahl der Ausführungen überschreitet.

Debugging-Session neu starten

Wenn Sie Ihre Debugging-Session neu starten, repositioniert der Debugger zum Anfang der Anwendung, während alle Ihre aktuellen Einstellungen (zum Beispiel Watchpoints oder Breakpoints) beibehalten werden, und alle Zähler sowie die Aufruf-Historie neu initialisiert werden. Folglich ist der erneute Start einer Debugging-Session nützlich, wenn Sie Ihre Anwendung neu starten möchten, ohne die für das Debugging relevanten Einstellungen erneut angeben zu müssen. Sie können Ihre Debugging-Session von einem beliebigen Punkt innerhalb der Anwendung neu starten, indem Sie entweder den Befehl **Restart** oder das entsprechende Symbol in der Symbolleiste wählen. Der Befehl **Restart** ist im Debug-Menü verfügbar.

Anmerkung: Wenn Sie eine Debugging-Session in einer Remote-Umgebung starten, steht der Befehl **Restart** nicht zur Verfügung, und wenn Sie einen DCOM- oder RPC-Server austeste, werden über den Befehl **Restart** die aufgerufene Methode oder das aufgerufene Subprogramm neu gestartet.

Source-Fenster des Debuggers

Wenn der Debugger aufgerufen wird, erhält er die Kontrolle des angegebenen Natural-Objekts und zeigt die betreffende Source im Source-Fenster an. Wenn die Source nicht zur Verfügung steht, bleibt das Fenster leer. Der Trace-Cursor (Cursor zur Ablaufverfolgung) wird in die erste ausführbare Sourcecode-Zeile platziert.

Wenn ein Benutzer ein neues Objekt öffnet, oder wenn ein Watchpoint oder Breakpoint innerhalb eines anderen Objekts außer dem gerade aktiven Objekt als Treffer (Hit Count) angezeigt wird, wird ein neues Source-Fenster geöffnet, das die Source dieses neuen Objekts anzeigt.

Die folgenden Themen sind im Folgenden erörtert:

Debug-Menü

Die folgenden Befehle des **Debug**-Menüs stehen in Zusammenhang mit dem Source-Fenster zur Verfügung:

Step Into (Objekt mit Step Into schrittweise ausführen)
Wenn Sie den Befehl **Step Into** wählen, wird der nächste Programmschritt ausgeführt, und der Trace-Cursor (Cursor zur Ablaufverfolgung) wird in die betreffende Sourcecode-Zeile platziert.

Wenn in dieser Sourcecode-Zeile ein weiteres Natural-Objekt aufgerufen oder aufgenommen wird, führt der Debugger in diesem Objekt den nächsten Programmschritt aus.

Step Over (Objekt mit Step Over überspringen)
Wenn Sie den Befehl **Step Over** wählen, wird der nächste Programmschritt ausgeführt, und der Trace-Cursor (Cursor zur Ablaufverfolgung) wird in die betreffende Sourcecode-Zeile platziert. Dieses Mal überspringt der Debugger allerdings ein aufgerufenes oder aufgenommenes Natural-Objekt, hält aber an, wenn dieses Objekt Watchpoints oder Breakpoints enthält.

Step Out (Objekt mit Step Out verlassen)

Wenn Sie den Befehl **Step Out** wählen, kehrt der Debugger zur vorherigen Programmebene zurück, hält aber an, wenn er einen Watchpoint oder Breakpoint vorfindet, bevor diese vorherige Ebene erreicht ist.

Animated Step Into (Schrittweises Ausführen des Programms in Objekt)

Wenn Sie den Befehl **Animated Step Into** wählen, wird das Programm bis zum Ende automatisch Schritt für Schritt ausgeführt. Der Debugger greift auf ein aufgerufenes oder aufgenommenes Natural-Objekt zu.

Animated Step Over (Schrittweises Ausführen des Programms und Objekt überspringen)

Wenn Sie den Befehl **Animated Step Over** wählen, wird das Programm bis zum Ende automatisch Schritt für Schritt ausgeführt. Der Debugger überspringt dabei ein aufgerufenes oder aufgenommenes Natural-Objekt; wenn ein Watchpoint oder Breakpoint gesetzt ist, springt er zu der betreffenden Statement-Zeile und fährt mit der automatischen Abarbeitung fort.

Go (Weiter)

Wenn Sie den Befehl **Go** wählen, wird das Programm bis zum nächsten aktiven Watchpoint oder Breakpoint ausgeführt, und der Trace-Cursor (Cursor zur Ablaufverfolgung) wird in der entsprechenden Sourcecode-Zeile platziert.

Go Until Next Event (Weiter bis zum nächsten Event)

Wenn Sie den Befehl **Go Until Next Event** wählen, dann hat dies dieselben Auswirkungen wie der Befehl **Go** in einer nicht ereignisgesteuerten Anwendung. In einer ereignisgesteuerten Anwendung wird das Objekt aber ausgeführt, bis der nächste Event an die Anwendung gesandt wird; es hält an, wenn ein aktiver Watchpoint oder Breakpoint auftritt, bevor der nächste Event gesandt wird.

Run to Cursor (Weiter zum Cursor)

Wenn Sie den Befehl **Run to Cursor** wählen, wird das Programm ausgeführt, bis die Source-Zeile an der aktuellen Cursor-Position erreicht ist.

Show Trace Position (Position zur Ablaufverfolgung zeigen)

Wenn Sie den Befehl **Show Trace Position** wählen, wird der aktuelle Trace-Cursor (Cursor zur Ablaufverfolgung) angezeigt.

Toggle Breakpoint (Breakpoint ein/ausschalten)

Wenn Sie den Befehl **Toggle Breakpoint** (Breakpoint ein/ausschalten) wählen, wird ein Breakpoint für die aktuelle Trace-Position (Position zur Ablaufverfolgung) zur Funktionsleiste für Breakpoints hinzugefügt. Wenn ein Breakpoint für diese Cursor-Position bereits vorhanden ist, wird er von der Funktionsleiste für Breakpoints entfernt.

Calls (Aufrufe)

Das Untermenü **Calls** (Aufrufe) bietet eine Liste (Historie) mit den zuletzt aufgerufenen Natural-Objekten, einschließlich Copycodes und Inline-Subroutinen. Bis zu 20 Objekte können aufgelistet werden; das zuletzt aufgerufene Objekt erscheint oben in der Liste.

Die Objekt-Liste besteht aus den folgenden Informationen:

- Die Programmebene des aufgerufenen Objekts ohne Copycodes und Inline-Subroutinen.

- Die Programmebene des aufgerufenen Objekts mit Copycodes und Inline-Subroutinen.
- Der Name des aufgerufenen Objekts.
- Der Typ des aufgerufenen Objekts.
- Die Event- und Kontroll-Handle des abzuarbeitenden Event-Handlers (nur bei ereignisgesteuerten Anwendungen).

Die Statusleiste am unteren Rand des Hauptfensters des Debuggers zeigt zusätzliche Informationen zum aufgerufenen Objekt an:

- Der Name des aufrufenden Objekts:

  "Natural" wird als aufrufendes Objekt angezeigt, wenn das aufgerufene Objekt das Start-Programm der Anwendung oder ein Programm ist, das vom Natural-Stack aktiviert wurde (einschließlich Fehlertransaktionsprogrammen und Programmen, die von einem `RUN`-Statement innerhalb der Anwendung aktiviert wurden).

- Die Sourcecode-Zeile, in der das Objekt aufgerufen wurde:

  Wenn Sie ein Objekt (außer Natural) aus der Liste wählen, wird die Source des aufrufenden Programms in der Mitte des Source-Fensters angezeigt, wobei der Cursor am Anfang der Zeile platziert wird, in der der Aufruf erfolgte.

Variables-Menü

Das **Variables**-Menü dient dazu, um:

- Den Inhalt von ausgewählten Variablen anzuzeigen.
- Den Inhalt von ausgewählten Variablen zu ändern.
- Den Inhalt der Variable an der aktuellen Trace-Position (Position zur Ablaufverfolgung) per Quick Watch schnell zu überwachen.
- Variablen zur Funktionsleiste für Watch-Variablen hinzufügen.
- Variablen zur Funktionsleiste für Watchpoints hinzufügen.

Dialogfelder

Wenn Sie den Befehl **Display** (Anzeigen), **Modify** (Ändern), **Add Watch** (Watch hinzufügen) oder **Add Watchpoint** (Watchpoint hinzufügen) wählen, erscheint ein Dialogfeld, in dem eine Liste aller lokalen Variablen, globalen Variablen, AIV-Variablen oder Systemvariablen angezeigt wird, die im aktuellen Debugging-Kontext aktiv sind. Die folgenden Kontrollelemente sind Bestandteil dieses Dialogfelds:

- Das Textfeld **Variable**, das die gerade ausgewählte Variable zeigt.

- Das Feld **Line Reference** or **Context ID** (Zeilenreferenz oder Kontext-ID), das die Sourcecode-Zeilennummer der Variable oder Kontext-Variable zeigt, die im Textfeld **Variable** gerade enthalten ist.

  Das Feld **Line Reference** (Zeilenreferenz) wird nur angezeigt, wenn die Zeilenreferenz erforderlich ist, um die Auswahl der Variablen eindeutig zu gestalten. Dies ist der Fall, wenn:

  - die Variable zu einer Map gehört; dann enthält das Feld die Sourcecode-Zeilennummer des betreffenden, vom Map-Editor generierten `RULEVAR`-Syntaxelements;
  - die Variable entweder eine Datenbank-Variable (nur Reporting Mode) oder eine der folgenden Variablen ist: `*ISN`, `*COUNTER`, `*NUMBER`; dann enthält das Feld die Sourcecode-Zeilennummer der betreffenden Datenbank- Schleife oder des entsprechenden Datenzugriffs-Statements;
  - die Variable im Reporting Mode definiert ist, aber ohne ein `DEFINE DATA`-Statement.

  Das Feld **Context ID** (Kontext-ID) wird nur angezeigt, wenn die Variable eine Kontext-Variable ist; dann enthält das Feld die „ctx-Id“ (Kontext-ID).
- Das Listenfeld **History List** (Historien-Liste) - nur bei **Display**-Befehl - das die zuletzt ausgewählten Variablen (bis zu 20) enthält, die einen FIFO-Mechanismus (first-in first-out) nutzen.

  Die Historien-Liste unterstützt Sie dabei, eine Variable schnell zu lokalisieren, die bereits vorher ausgewählt war. Variablen können von der Historien-Liste genauso ausgewählt werden wie von der Variablen-Liste.
- Das Listenfeld **Variable**, das die betreffende Variablen-Liste enthält.

Variablen auswählen

Wenn Sie eine Variable in der Variablen-Liste eines Dialogfeldes wählen, erscheint sie im entsprechenden **Variable**-Textfeld.

Wenn Sie eine Variable in der Variablen-Liste eines Dialogfeldes wählen, wird ein weiteres Dialogfeld angezeigt (außer beim **Watch**-Befehl).

Wenn Sie eine Array- oder Variablen-Gruppe wählen:

- die einzelnen Array- oder Gruppen-Elemente werden im zweiten Dialogfeld angezeigt (Anzeigen),
- das Array wird im zweiten Dialogfeld zum Ändern angezeigt; Gruppen können nicht geändert werden (Ändern),
- eine entsprechende Fehlermeldung wird angezeigt (Watchpoint).

Sie können auch eine Variable wählen, indem Sie sie zuerst direkt im Source-Fenster markieren und dann entsprechend den Befehl **Display** (Anzeigen), **Modify** (Ändern), **Add Watch** (Watch hinzufügen) oder **Add Watchpoint** (Watchpoint hinzufügen) auswählen. Dann zeigt das Textfeld **Variable** des betreffenden Dialogfeldes genau das Stück Sourcecode an, das Sie markiert haben, welches dann geändert werden kann.

Text im Source-Fenster markieren

Im Source-Fenster können Sie Variablen oder Zeichenketten entweder mit der Maus oder der Tastatur zur Auswahl markieren.

Wenn Sie Text mit der Maus markieren, stellen Sie den Mauszeiger auf das erste auszuwählende Zeichen, ziehen Sie den Zeiger zum letzten Zeichen, das Sie auswählen möchten, und lassen Sie die Maustaste wieder los. Um eine Auswahl abzubrechen, treffen Sie diese Wahl irgendwo im Dokument.

Wenn Sie die Tastatur benutzen, um Text zu markieren, werden Cursor-Bewegungstasten benutzt. Stellen Sie den Cursor zuerst auf ein Zeichen, indem Sie eine Pfeiltaste benutzen, drücken Sie dann die UMSCHALT-Taste, halten Sie sie gedrückt, und benutzen Sie die folgenden Tasten zur Textauswahl:

- den NACH-LINKS, um den Bereich links von Ihrer Cursor-Position zu markieren,
- den NACH-RECHTS, um den Bereich rechts von Ihrer Cursor-Position zu markieren,
- die ENDE-Taste, um den Bereich bis zum Ende der Sourcecode-Zeile zu markieren,
- die POS1-Taste, um den Bereich bis zum Anfang der Sourcecode-Zeile zu markieren.

Display (Anzeigen)

Mit diesem Befehl kann eine Variable von der Liste im Dialogfeld zur Anzeige ausgewählt werden, und zwar zusammen mit ihrem aktuellen Inhalt in einem zweiten **Display Variable**-Dialogfeld (Dialogfeld Variable anzeigen), wo Sie zwischen alphanumerischer und binärer Darstellung des Variablenwertes wählen können.

Wenn Sie ein Array, eine Handle-Variable oder eine Gruppe von Variablen auswählen, werden die einzelnen Elemente und ihre Werte im zweiten Dialogfeld aufgelistet. Bei Arrays wird ein beliebiger Variablenindex-Ausdruck ausgewertet.

Die Element-Liste kann erweitert oder verkürzt werden, indem Sie die Schaltfläche **Expand/Contract** (Erweitern/Verkürzen) wählen. Immer wenn die Anzahl der Arrays, Gruppen oder Dialog-Elemente in der Liste ein bestimmtes Anzeige-Limit überschreitet, erscheint eine More-Zeile (Mehr-Zeile), die zur Anzeige weiterer Objekte benutzt werden kann. Als Alternative dazu kann auch der Befehl **Expand** (Erweitern) benutzt werden.

Eine Variable, ein Array oder eine Gruppe von Variablen kann auch zur Anzeige im Dialogfeld **Display Variable** (Variable anzeigen) ausgewählt werden, indem Sie die Variable, das Array oder die Gruppe von Variablen mit der linken Maustaste direkt im Source-Fenster wählen.

Modify (Ändern)

Mit diesem Befehl kann eine Variable von der Liste im Dialogfeld zur Anzeige ausgewählt werden, und zwar zusammen mit ihrem aktuellen Wert in einem zweiten **Modify Variable**-Dialogfeld (Variable ändern), wo ihr Wert geändert werden kann.

Wenn Sie eine Systemvariable ändern möchten, werden nur Systemvariablen im ersten Dialogfeld angezeigt, die geändert werden können.

Wenn Sie ein Array ändern möchten, wird nur sein Name angezeigt, aber es erscheinen keine Werte im zweiten Dialogfeld. Der Wert, den Sie eingeben, ist dann gültig für alle Array-Elemente.

Gruppen von Variablen können nicht zum Ändern ausgewählt werden.

Quick Watch (Schnell überwachen)

Bei diesem Befehl erscheint ein Dialogfeld, das den Inhalt der Variablen an der aktuellen Cursor-Position anzeigt.

Add Watch (Watch-Variablen hinzufügen)

Bei diesem Befehl können Variablen, Arrays oder Gruppen von Variablen von der Liste im Dialogfeld ausgewählt werden, um sie zur Funktionsleiste Watchvariables (Watch-Variablen) hinzuzufügen.

Add Watchpoint (Watchpoint hinzufügen)

Mit diesem Befehl können einzelne Variablen und einzelne Gruppen oder Array-Elemente von der Liste im Dialogfeld zur Definition eines Watchpoints in einem zweiten Set Watchpoint-Dialogfeld (Watchpoint setzen) ausgewählt werden; Arrays und Gruppen von Variablen können nicht ausgewählt werden.

Das zweite **Set Watchpoint**-**Dialogfeld** zeigt den Namen des Watchpoints an (der dem Namen der ausgewählten Variablen entspricht), und zwar zusammen mit seiner Zeilenreferenz (wenn vorhanden), und die Namen des betreffenden Natural-Objekts und der entsprechenden Natural-Library.

Anmerkung: Bei Systemvariablen ist der betreffende Watchpoint nicht an eine spezifische Library und ein spezifisches Objekt angebunden; deshalb sind der Objekt- und Library-Name stets `SYSTEM`.

Um einen Watchpoint zu definieren, geben Sie die folgenden Elemente in den entsprechenden Feldern an:

- den Status des Watchpoints,
- eine Bedingung für den zu aktivierenden Watchpoint (optional),

- die Anzahl der Sprünge vor der Ausführung des Watchpoints,
- die maximale Anzahl der Ausführungen des Watchpoints.

File Menu (Menü Datei)

Die folgenden Befehle des **File**-Menüs sind in Verbindung mit dem Source-Fenster verfügbar:

Open (Öffnen)
Mit dem Befehl **Open** (Öffnen) können Sie ein weiteres Source-Programm angeben, das in das Source-Fenster geladen werden soll. Es erscheint das Dialogfeld **Open Source** (Source öffnen), in dem Sie den Programm-Namen und den passenden Library-Namen angeben, wenn das Programm nicht in der aktuellen Library (Voreinstellung) enthalten ist.

Sie können auch eine in das Dialogfeld **Open Source** (Source öffnen) zu stellende Zeichenkette auswählen, indem Sie ihren Namen im Source-Fenster **markieren** und dann den Befehl **Open** (Öffnen) wählen.

Close (Schließen)
Mit dem Befehl **Close** (Schließen) schließen Sie das gerade aktive Source-Fenster. Wenn das Source-Fenster, das Sie gerade schließen, die Trace-Bar (Ablaufverfolgungsleiste) enthält, wird das Fenster auf Symbolgröße verkleinert.

Exit (Verlassen)
Mit dem Befehl **Exit** (Verlassen) verlassen Sie den Debugger und beenden die aktuelle Programmausführung.

Edit-Menü

Die folgenden Befehle des **Edit**-Menüs sind in Verbindung mit dem Source-Fenster verfügbar:

Find (Suchen)
Mit dem Befehl **Find** (Suchen) können Sie im aktiven Fenster nach oben oder unten absuchen, um jede Ausprägung eines angegebenen Wortes oder einer spezifizierten Zeichenkette zu lokalisieren.

Es erscheint das Dialogfeld **Find** (Suchen), in dem Sie den in dem Textfeld Find zu lokalisierenden Text eingeben können. Außerdem können Sie die Optionen **Match Upper/Lower Case** (Groß/Kleinschreibung) und **Whole Words Only** (Nur ganze Wörter) ein- oder ausschalten.

Wenn Sie den Text gefunden haben, wird die erste Ausprägung des angegebenen Textes hell hervorgehoben (ausgewählt), währenddessen Sie in einer Meldung darüber informiert werden, wenn der Text nicht gefunden werden konnte.

Mit der Option **Match Upper/Lower Case** (Groß/Kleinschreibung) können Sie angeben, ob Sie mit der Find-Operation nach einer genauen Übereinstimmung (ON) oder nur nach denselben Zeichen suchen möchten, ungeachtet der Groß- oder Kleinschreibung (OFF).

Mit der Option **Whole Words Only** (Nur ganze Wörter) können Sie angeben, ob Sie mit der Find-Operation nach Ausprägungen suchen möchten, die nur aus ganzen Wörtern bestehen, und nicht Bestandteil einer Zeichenkette sind (ON), oder aber nach allen Ausprägungen des angegebenen Textes, ganzen Wörtern und Teilen einer Zeichenkette (OFF).

Um die Such-Richtung zu ändern, wählen Sie die Schaltfläche **Up** (Nach oben), um in Aufwärtsrichtung zum Anfang des Textes hin zu suchen, oder wählen Sie die Schaltfläche **Down** (Nach unten), um in Abwärtsrichtung zum Ende des Textes hin zu suchen; **Down** ist die Voreinstellung.

Wenn die Find-Operation nicht am Anfang (oder Ende) des Textes beginnt, und der angegebene Text nicht gefunden werden kann, erscheint ein Dialog. Sie können **Yes** wählen, um die Such-Operation am Anfang (oder Ende) des Textes fortzusetzen, oder **No**, um die Suche abzubrechen.

Sie können auch eine Zeichenkette auswählen, die in das Textfeld **Find** (Suchen) gestellt werden soll, indem Sie sie direkt im Source-Fenster markieren und dann den Befehl **Find** (Suchen) wählen.

Find Next (Suchen Nächstes)
Mit diesem Befehl können Sie die vorherige Such-Operation wiederholen und die nächste Ausprägung des mit dem Befehl **Find** angegebenen Textes lokalisieren.

Funktionsleiste Watchvariables

Die Funktionsleiste Watchvariables dient in erster Linie dazu, vorher ausgewählte Variablen zur genaueren und permanenten Beobachtung ihres Inhalts anzuzeigen.

Sie bietet ein Kontextmenü, das entweder die Befehle anzeigt, die in Verbindung mit der gesamten Funktionsleiste benutzt werden können, oder sie zeigt die Befehle an, die mit jeder einzelnen Watch-Variablen benutzt werden können.

Um das Kontextmenü zu öffnen, wählen Sie mit der rechten Maustaste entweder die Funktionsleisten-Titelleiste oder eine bestimmte Watch-Variable.

Funktionsleiste Variables

Die Funktionsleiste Variables zeigt alle Variablen an, die im aktuellen Status der Programmausführung zur Verfügung stehen. Alle Variablen sind in unterschiedliche Kategorien unterteilt. Diese Kategorien sind lokale Variablen, globale Variablen, Systemvariablen, AIV-Variablen und Kontext-Variablen. Sie können zwischen diesen Kategorien wechseln, indem Sie das entsprechende Register am unteren Rand der Funktionsleiste wählen. Um den Inhalt zu ändern, wählen Sie das Inhaltsfeld einer bestimmten Variablen aus. Für einige Systemvariablen gilt nur Lesezugriff (read-only), und sie können deshalb nicht geändert werden.

Die Funktionsleiste Variables bietet ein Kontextmenü, das entweder die Befehle anzeigt, die in Verbindung mit der gesamten Funktionsleiste benutzt werden können, oder aber die Befehle anzeigt, die mit jeder einzelnen Variablen benutzt werden können.

Um das Kontextmenü zu öffnen, wählen Sie mit der rechten Maustaste entweder die Funktionsleisten-Titelleiste oder eine bestimmte Variable.

Funktionsleiste Watchpoints and Breakpoints

Die Funktionsleiste Watchpoints and Breakpoints dient dazu, Watchpoints und Breakpoints hinzuzufügen und zu verwalten. Sie können zwischen den Watchpoints und Breakpoints umschalten, indem Sie das betreffende Register am unteren Rand der Funktionsleiste wählen.

Watchpoints

Wenn Sie Watchpoints benutzen, können Sie rasch „unzulässige“ Änderungen an Natural-Variablen durch Fehler enthaltende Objekte entdecken.

Standardmäßig dienen Watchpoints dazu, den Debugger anzuweisen, die Ausführung von Natural-Objekten zu unterbrechen, wenn sich der Inhalt einer Variablen ändert. Wenn Sie aber einen bestimmten Wert für die Variable zusammen mit einem Watchpoint-Operator beim Setzen eines Watchpoints angeben, kann eine Bedingung gesetzt werden, die den Watchpoint nur aktiviert, wenn die Bedingung wahr ist.

Eine Variable gilt als geändert, wenn entweder ihr aktueller Wert vom Wert abweicht, der aufgezeichnet wurde, als der Watchpoint zuletzt gestartet wurde, oder wenn er vom Anfangswert abweicht.

Um einen Watchpoint zeitweise zu deaktivieren, entfernen Sie das Häkchen aus dem Kontrollkästchen des betreffenden Watchpoint-Eintrags.

Das Watchpoint-Register dieser Funktionsleiste bietet ein Kontextmenü, das entweder die Befehle anzeigt, die in Verbindung mit dem gesamten Register benutzt werden können, oder das die Befehle anzeigt, die mit jedem einzelnen Watchpoint benutzt werden können.

Um das Kontextmenü zu öffnen, wählen Sie mit der rechten Maustaste entweder die Register-Titelleiste oder einen bestimmten Watchpoint.

Watchpoint hinzufügen

Ein neuer Watchpoint kann hinzugefügt werden, indem Sie entweder den Befehl **Add Watchpoint** (Watchpoint hinzufügen) vom Menü Variables oder den Befehl **Add** (Hinzufügen) vom Kontextmenü des Watchvariables-Registers auswählen.

Das Dialogfeld **Add Watchpoint** (Watchpoint hinzufügen) ermöglicht es Ihnen, einzelne Variablen, Arrays und einzelne Gruppen-Elemente von der Liste der verfügbaren Variablen auszuwählen. Wenn Sie den Dialog mit der Schaltfläche **OK** schließen, bewirkt, dass das Dialogfeld **Set Watchpoint** (Watchpoint setzen) geöffnet wird, das es Ihnen ermöglicht, eine Bedingung für diesen Watchpoint anzugeben.

Anmerkung: Bei Systemvariablen ist der betreffende Watchpoint nicht an eine spezifische Library und ein bestimmtes Objekt angebunden; deshalb sind der Objekt- und Library-Name stets `SYSTEM`.

Dialogfeld Set Watchpoint (Watchpoint setzen)

Das Dialogfeld **Set Watchpoint** (Watchpoint setzen) zeigt den Namen des Watchpoints an (der dem Namen der ausgewählten Variablen entspricht), und zwar zusammen mit seiner Zeilenreferenz/Kontext-ID (wenn vorhanden) und den Namen des betreffenden Natural-Objekts und der entsprechenden Natural-Library.

Anmerkung: Bei Systemvariablen ist der betreffende Watchpoint nicht an eine spezifische Library und ein bestimmtes Objekt angebunden; deshalb ist der Objekt- und Library-Name stets `SYSTEM`.

Um einen Watchpoint zu definieren, können Sie die folgenden Elemente in den entsprechenden Feldern angeben:

- Den Status des zu setzenden Watchpoints; gültige Stati sind "active" (aktiv) - Voreinstellung - und "pending" (anstehend).
- Eine Bedingung für den zu aktivierenden Watchpoint (optional).

Sie können einen passenden Wert und Watchpoint-Operator angeben; wenn kein Operator und Wert (d.h. eine Bedingung) angegeben werden, gilt die Voreinstellung (MOD). Eine Beschreibung der einzelnen Watchpoint-Operatoren finden Sie unten).

- Die Anzahl der übersprungenen Werte vor der Ausführung des Watchpoints, wenn er nicht ausgeführt werden soll, bis das Programm eine bestimmte Anzahl von Malen abgelaufen ist; die Voreinstellung ist 0.
- Die maximale Anzahl der Ausführungen des Watchpoints; die Voreinstellung ist 0.

Ein Watchpoint wird erst gesetzt, wenn Sie entweder mit **OK** bestätigen oder EINGABE drücken. Wenn Sie die Schaltfläche **Cancel** (Abbrechen) wählen oder ESC drücken, wird kein Watchpoint gesetzt.

Sobald ein Watchpoint angegeben wurde, bleibt er stehen, bis Sie ihn explizit löschen.

Watchpoint-Operatoren

Watchpoint-Operatoren werden über Optionsfelder gesetzt; die folgenden Watchpoint-Operatoren sind verfügbar:

Operator	Steht für	Beschreibung
MOD	Modifikation	Der Watchpoint wird jedesmal aktiviert, wenn eine Änderung der Variablen auftritt. Voreinstellung.
LT	Kleiner als	Der Watchpoint wird nur aktiviert, wenn der aktuelle Wert der Variablen kleiner als der angegebene Wert ist.
LE	Kleiner gleich	Der Watchpoint wird nur aktiviert, wenn der aktuelle Wert der Variablen kleiner gleich dem angegebenen Wert ist.
GT	Größer als	Der Watchpoint wird nur aktiviert, wenn der aktuelle Wert der Variablen größer als der angegebene Wert ist.
GE	Größer gleich	Der Watchpoint wird nur aktiviert, wenn der aktuelle Wert der Variablen größer gleich dem angegebenen Wert ist.
EQ	Gleich	Der Watchpoint wird nur aktiviert, wenn der aktuelle Wert der Variablen gleich dem angegebenen Wert ist.
NE	Ungleich	Der Watchpoint wird nur aktiviert, wenn der aktuelle Wert der Variablen ungleich dem angegebenen Wert ist.

Breakpoints

Ein Breakpoint ist ein Punkt, an dem die Kontrolle an den Benutzer zurückgegeben wird, während ein Natural-Objekt ausgeführt wird.

Um einen Breakpoint zeitweise zu deaktivieren, entfernen Sie das Häkchen aus dem Kontrollkästchen des betreffenden Breakpoint-Eintrags.

Das Breakpoint-Register dieser Funktionsleiste bietet ein Kontextmenü, das entweder die Befehle anzeigt, die in Verbindung mit dem gesamten Register benutzt werden können, oder das die Befehle anzeigt, die mit jedem einzelnen Breakpoint benutzt werden können.

Um das Kontextmenü zu öffnen, wählen Sie mit der rechten Maustaste entweder die Register-Titelleiste oder einen bestimmten Breakpoint.

Breakpoint hinzufügen

Mit dem Befehl **Add** (Hinzufügen) können Sie einen neuen Breakpoint definieren. Das Dialogfeld **Add Breakpoint** (Breakpoint hinzufügen) wird angezeigt, in dem Sie den Breakpoint definieren, indem Sie die folgenden Elemente in den betreffenden Feldern angeben:

- Den Status des zu setzenden Breakpoints; gültige Stati sind "active" (aktiv) - Voreinstellung - und "pending" (anstehend).
- Der Name des Natural-Objekts, das den Breakpoint enthalten soll; der standardmäßige Objekt-Name ist der Name des aktuell sich im Source-Fenster befindlichen Objekts.
- Der Name der Natural-Library, die das Objekt mit dem Breakpoint enthält; der standardmäßige Library-Name ist der Name der Library, die das sich aktuell im Source-Fenster befindliche Objekt enthält.
- Die Zeilennummer des Sourcecodes des Objekts, in der der Breakpoint ausgeführt werden soll.

Begin (Anfang) bedeutet in diesem Zusammenhang, dass der Breakpoint an der ersten ausführbaren Code-Zeile des angegebenen Objekts gesetzt werden soll; **End**(Ende) bedeutet, dass der Breakpoint an der letzten ausführbaren Code-Zeile des angegebenen Objekts gesetzt werden soll.

- Die Anzahl der übersprungenen Elemente vor der Ausführung des Breakpoints, wenn er erst ausgeführt werden soll, wenn das Programm eine bestimmte Anzahl von Malen abgelaufen ist; die Voreinstellung ist 0.
- Die maximale Anzahl der Ausführungen des Breakpoints; die Voreinstellung ist 0.

Ein Breakpoint wird erst gesetzt, wenn Sie entweder mit **OK** bestätigen oder EINGABE drücken. Wenn Sie die Schaltfläche **Cancel** (Abbrechen) wählen oder ESC drücken, wird kein Breakpoint gesetzt.

Breakpoints können auch direkt im gerade im Source-Fenster enthaltenen Programm gesetzt werden, indem Sie die passende Statement-Zeile mit der rechten Maustaste doppelt anklicken. Auf diese Art wird ein Breakpoint mit allen Standardwerten und der entsprechenden Sourcecode-Zeilennummer definiert. Er kann angezeigt und/oder geändert werden, indem Sie die betreffenden Funktionen benutzen.

Breakpoints können nicht in Kommentarzeilen oder in einer Statement-Zeile außer der ersten Zeile gesetzt werden, wenn ein einzelnes Statement mehr als eine Zeile einnimmt.

Sobald ein Breakpoint definiert wurde, bleibt er erhalten, bis Sie ihn explizit löschen.