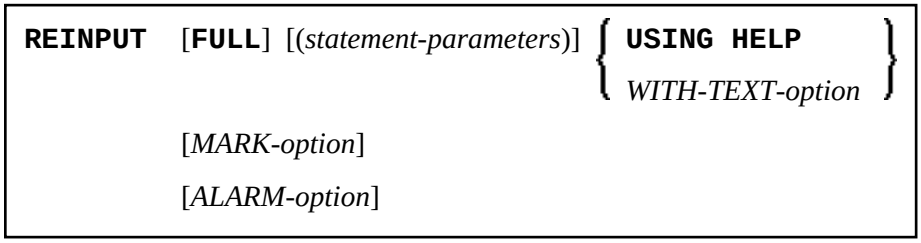
REINPUT



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Beispiele

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

Verwandte Statements: DEFINE WINDOW | INPUT | SET WINDOW

Gehört zur Funktionsgruppe: *Bildschirmgenerierung für interaktive Verarbeitung*

Funktion

Das Statement `REINPUT` dient dazu, zu einem `INPUT`-Statement zurückzukehren und dieses erneut auszuführen. Es wird in der Regel dazu benutzt, eine Fehlermeldung auszugeben, die dem Benutzer sagt, dass auf das `INPUT`-Statement hin ungültige Daten eingegeben wurden. Siehe *Beispiel 1*.

Zwischen einem `INPUT`-Statement und dem dazugehörigen `REINPUT`-Statement werden keine `WRITE`- oder `DISPLAY`-Statements ausgeführt. Im Batch-Betrieb ist das `REINPUT`-Statement nicht gültig.

Wenn das `REINPUT`-Statement ausgeführt wird, setzt es den Programmstatus, was die Verarbeitung von Unterprogrammen, besonderen Bedingungen und Verarbeitungsschleifen anbelangt, wieder auf den Stand zurück, der galt, als das `INPUT`-Statement ausgeführt wurde (vorausgesetzt das `INPUT`-Statement ist nach wie vor aktiv). Wurde nach der Ausführung des `INPUT`-Statements eine Verarbeitungsschleife gestartet und das `REINPUT`-Statement befindet sich innerhalb dieser Schleife, so wird die Schleife abgebrochen und erst dann neu gestartet, wenn das `INPUT`-Statement aufgrund des `REINPUT`-Statements erneut ausgeführt worden ist.

Wird nach der ersten Ausführung des `INPUT`-Statements eine Hierarchie von Unterprogrammen aufgerufen und das `REINPUT`-Statement steht in einem dieser Unterprogramme, so kehrt Natural automatisch zu dem Programm zurück, das bei der Ausführung des `INPUT`-Statements aktiv war.

Steht ein `INPUT`-Statement innerhalb einer Verarbeitungsschleife, eines Unterprogramms oder eines nur unter bestimmten Bedingungen verarbeiteten Statement-Blocks, so kann ein `REINPUT`-Statement nicht ausgeführt werden, wenn der Status, unter dem das `INPUT`-Statement ausgeführt wurde, bereits beendet ist. Eine derartige Situation würde einen Programmabbruch und eine entsprechende Fehlermeldung zur Folge haben.

Siehe auch die *Statements REINPUT/REINPUT FULL* im Abschnitt *Dialog-Gestaltung* im *Leitfaden zur Programmierung*.

Note:

Wird ein Eingabe-/Ausgabefeld (Option `(AD=M)`) durch ein `INPUT`-Statement angezeigt, werden die am Schirm sichtbaren Daten nur dann in die Variable zurück übertragen, wenn das Feld als "verändert" (modified) angesehen wird. Ein Feld erhält den Status `MODIFIED`, wenn eine der folgenden Operationen erfolgt ist:

- Der Inhalt des Feldes wurde verändert (d.h., es wurden *andere* Daten in das Feld eingegeben).
- Die Taste EEOF (Erase to End of Field) wird bei einem leeren Feld gedrückt.
- Leerzeichen werden in ein leeres Feld oder nach dem letzten Nicht-Leerzeichen im Feld eingegeben.
- Der Profilparameter `CVMIN` wurde auf `ON` gesetzt und die Daten im Feld werden durch Editiermaßnahmen verändert, die letzten Endes zur Wiederherstellung des Feldinhaltes führen (z.B. durch Überschreiben des ersten Zeichens mit dem vorhandenen Zeichen).

Der Inhalt eines Feldes, der letztendlich unverändert bleibt, wird nicht vom Bildschirmfeld in die Variable übertragen.

Die Ausführung eines `REINPUT`-Statements (ohne `FULL`-Option) hat keinen Einfluss auf den `MODIFIED`-Status eines Eingabe-/Ausgabefeldes. Ein Feld gilt weiterhin als *nicht verändert*, wenn es nicht über das `INPUT`-Statement mittels einer der oben aufgelisteten Operationen verändert wurde. Anders ausgedrückt wird ein Feld als *verändert* Behandelt, wenn mindestens eine der erwähnten Operationen durchgeführt wurde, und zwar unabhängig davon, wie oft das `INPUT`-Statement durch `REINPUT`-Statements (ohne `FULL`-Option) neu gesendet wurde.

Mit anderen Worten wird ein mittels `INPUT`-Statement angezeigter Feldwert, der von einem `REINPUT`-Statement (ohne `FULL`-Option) getriggert wurde, nur dann in die Variable übernommen, wenn der Feldinhalt durch eine der genannten Operationen verändert wurde.

Der `MODIFIED`-Status kann geprüft werden, wenn im Programmcode eine Attributkontrollvariable (Option `CV`) zu dem Feld zugeordnet wurde, das mit der `MODIFIED`-Option z.B. des `IF`-Statements nach dem `INPUT`-Statement geprüft wird.

Syntax-Beschreibung

<table>
<tr><td>REINPUT FULL</td><td>Wenn Sie die Option FULL in einem REINPUT-Statement angeben, wird das entsprechende INPUT-Statement vollständig neu ausgeführt:
Bei einem normalen REINPUT-Statement (ohne FULL-Option) werden Inhalte von Variablen, die zwischen INPUT- und REINPUT-Statement geändert wurden, nicht angezeigt; d.h. alle Variablen auf dem Schirm zeigen den Inhalt, den sie hatten, als das INPUT-Statement ursprünglich ausgeführt wurde.
Bei einem REINPUT FULL-Statement werden alle nach der ersten Ausführung des INPUT-Statements gemachten Änderungen sichtbar, wenn das INPUT-Statement erneut ausgeführt wird; d.h. alle Variablen auf dem Schirm haben den Inhalt, den sie zum Zeitpunkt der Ausführung des REINPUT-Statements hatten.
Anmerkung:
Der Inhalt reiner Eingabefelder (AD=A) wird durch REINPUT FULL wieder gelöscht.

Eine andere Eigenschaft des REINPUT FULL-Statements besteht darin, dass der Status der Kontrollvariable auf NOT MODIFIED (nicht geändert) zurückgesetzt wird. Dies erfolgt nicht mittels des normalen REINPUT-Statements. Um zu überprüfen, ob einer Attribut- Kontrollvariablen der Status MODIFIED (geändert) zugewiesen wurde, benutzen Sie die MODIFIED-Option.

Siehe auch Beispiel 3 - REINPUT FULL mit MARK POSITION.</td></tr>
<tr><td>statement-parameters</td><td>Mit einem REINPUT-Statement gesetzte Parameter gelten für alle Felder, die im Statement angegeben sind.

Auf Feldebene gesetzte Parameter (siehe MARK-Option) haben für das betreffende Feld Gültigkeit vor auf Statement-Ebene gesetzten.
<table>
<tr><td colspan="2">Parameter, die mit REINPUT-Statement angegeben werden können:</td><td>Spezifikation</td></tr>
<tr><td colspan="2"></td><td>S=auf Statement-Ebene</td></tr>
<tr><td colspan="2"></td><td>E=auf Element-Ebene</td></tr>
<tr><td>AD</td><td>Attribute Definition *</td><td>SE</td></tr>
<tr><td>CD</td><td>Color Definition</td><td>S</td></tr>
</table>
* Wird AD=P auf Statement-Ebene gesetzt, so sind alle Felder geschützt, außer den in der MARK-Option angegebenen.

Informationen zu den o.g. Session-Parametern finden Sie in der Parameter-Referenz.</td></tr>
</table>

USING HELP	`USING HELP` bewirkt, dass die für die `INPUT`-Map definierte Helproutine aufgerufen wird. `USING HELP` in Verbindung mit der `MARK`-Option (siehe unten) bewirkt, dass die für das erste in der `MARK`-Option angegebene Feld definierte Helproutine aufgerufen wird. Ist für das Feld keine Helproutine definiert, wird die Helproutine für die Map aufgerufen. Beispiel: `REINPUT USING HELP MARK 3` In diesem Beispiel wird die für das dritte Feld der `INPUT`-Map definierte Helproutine aufgerufen.
WITH-TEXT-option	Mit der Option `WITH TEXT` können Sie einen Text angeben, der in der Meldungszeile angezeigt werden soll. Siehe *WITH TEXT-Option* weiter unten.
MARK-option	Mit der `MARK`-Option können Sie ein bestimmtes Feld markieren, d.h. bei der Ausführung des `REINPUT`-Statements wird der Cursor in dieses Feld plaziert. Siehe *MARK-Option* weiter unten.
ALARM-option	Diese Option bewirkt, dass der Warnton des Terminals ausgelöst wird, wenn das `REINPUT`-Statement ausgeführt wird. Siehe *ALARM-Option* weiter unten.

WITH TEXT-Option

Mit der Option `WITH TEXT` können Sie einen Text angeben, der in der Meldungszeile angezeigt werden soll. Der Text ist in der Regel eine Meldung, die angibt, welche Maßnahme zum Abarbeiten des Bildschirms getroffen werden sollte, oder wie der Fehler korrigiert werden kann.

[**WITH**] [**TEXT**] { ***** *operand1* | *operand2* } [(*attributes*)] [,*operand3*]**...** *7*

Operanden-Definitionstabelle:

Operand	Mögliche Struktur					Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1	C	S						N	P	I		B *							ja	nein
operand2	C	S				A	U												ja	nein
operand3	C	S				A	U	N	P	I	F	B	D	T	L				ja	nein

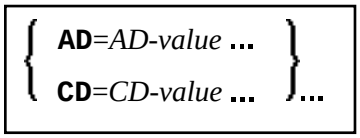
* Format B von *operand1* kann nur mit einer Länge von kleiner gleich 4 benutzt werden.

Syntax-Element-Beschreibung:

operand1	**Meldungstext aus der Natural-Fehlermeldungsdatei:** Als *operand1* geben Sie eine Natural-Fehlernummer an. Natural liest dann die entsprechende Fehlermeldung von der Natural- Fehlermeldungsdatei. Es können entweder benutzerdefinierte Meldungen oder Natural- Systemmeldungen gelesen werden. • Wenn Sie einen positiven Wert von bis zu vier Stellen (z.B.: `0954`) angeben, werden benutzerdefinierte Meldungen gelesen. • Wenn Sie einen negativen Wert von bis zu vier Stellen (z.B.: `–0954`) angeben, werden Natural-Systemmeldungen gelesen. Siehe auch *Beispiel 4 — WITH TEXT-Option*. Natural-Meldungsdateien werden mit der `SYSERR`-Utility erstellt und gepflegt.
operand2	**Meldungstext:** Als *operand2* geben Sie den Text an, der in der Meldungszeile ausgegeben werden soll. Siehe auch *Beispiel 4 — WITH TEXT-Option*.
attributes	**Ausgabe-Attribute:** Als *attributes* können Sie *operand1* oder *operand2* bestimmte Anzeige- und Farbattribute zuordnen. Diese Attribute und die Syntax, die benutzt werden kann, sind im Abschnitt *Ausgabe-Attribute* weiter unten beschrieben.
operand3	**Dynamische Ergänzung im Meldungstext:** *operand3* kann in Form einer numerischen Konstanten oder Textkonstanten oder als Name einer Variablen angegeben werden. Der angegebene Wert dient dazu, einen Teil einer mit *operand1* oder *operand2* angegebenen Meldung dynamisch zu generieren. Innerhalb der Fehlermeldung dient die Notation `:`*n*`:` zur Referenzierung von *operand3*, wobei *n* die Ausprägung (1 – 7) von *operand3* darstellt. Siehe auch *Beispiel 4 — WITH TEXT-Option*. **Anmerkung:** Werden mehrere *operanden3* angegeben, müssen diese mit einem Komma voneinander getrennt werden. Falls das Komma als Dezimalzeichen verwendet wird (wie mit dem Session-Parameter `DC` definiert) und es sich bei *operand3* um numerische Konstanten handelt, setzen Sie Leerzeichen vor und nach dem Komma, damit es nicht als Dezimalkomma missinterpretiert wird. Alternativ können mehrere *operanden3* auch mit dem Eingabebegrenzungszeichen (Input Delimiter Character, wie mit dem Session-Parameter `ID` definiert) voneinander getrennt werden; dies geht jedoch nicht im Falle von `ID=/` (Schrägstrich). Nicht signifikante Nullen oder Leerzeichen werden aus dem Feldwert entfernt, bevor er in einer Meldung angezeigt wird.

Ausgabeattribute

attributes sind zur Text-Anzeige verwendete Ausgabeattribute. Sie können folgende *attributes* angeben:



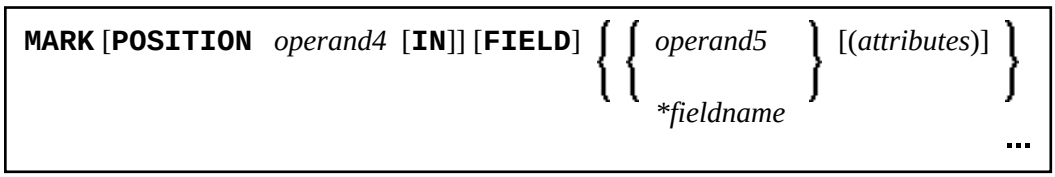
Die möglichen Parameterwerte sind in der *Parameter-Referenz* aufgeführt:

- *AD - Attribute-Definition*, Abschnitt *Feldanzeige*
- *CD - Farbdefinition*

Anmerkung:
Der Compiler akzeptiert mehr als einen Attributwert für ein Ausgabefeld. Beispielsweise können Sie angeben: `AD=BDI`. In einem solchen Fall gilt allerdings nur der letzte Wert. In dem vorliegenden Beispiel greift nur der Wert `I`, und das Ausgabefeld wird intensiviert dargestellt.

MARK-Option

Mit der `MARK`-Option können Sie ein bestimmtes Feld markieren, so dass bei der Ausführung des `REINPUT`-Statements der Cursor in dieses Feld plaziert wird. Sie können auch eine bestimmte Stelle innerhalb eines Feldes markieren. Außerdem können Sie Felder gegen Eingabe schützen sowie ihre Anzeige- und Farbattribute ändern.



Operanden-Definitionstabelle:

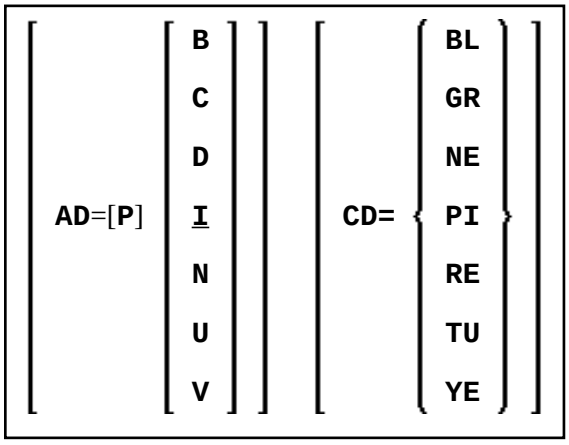
Operand	Mögliche Struktur					Mögliche Formate											Referenzierung erlaubt	Dynam. Definition
operand4	C	S						N	P	I							ja	nein
operand5	C	S	A					N	P	I							ja	nein

Syntax-Element-Beschreibung:

operand5	**Das zu markierende Feld:**
	
	Jedes mit einem INPUT-Statement angegebene Eingabefeld (AD=A oder AD=M) wird durchnumeriert (beginnend mit 1). Sie können als *operand5* die Nummer des Feldes angeben, in das der Cursor plaziert werden soll.
	
	Die Notation **fieldname* wird verwendet, um den Cursor in ein Feld zu positionieren, und zwar mittels des (im INPUT-Statement verwendeten) Namens des Feldes.
	
	Ist das betreffende INPUT-Feld ein Array, kann zur Markierung einer oder mehrerer Ausprägungen des Arrays ein eindeutiger Index oder ein Indexbereich angegeben werden.
	
	`INPUT #ARRAY (A1/1:5)`
	`...`
	`REINPUT (AD=P) 'TEXT' MARK *#ARRAY (2:3)`
	
	Ist *operand5* ebenfalls ein Array, so werden die Werte von *operand5* als Feldnummern für das INPUT-Array benutzt.
	
	`RESET #X(N2/1:2)`
	`INPUT #ARRAY ...`
	`...`
	`REINPUT (AD=P) 'TEXT' MARK #X (1:2)`

MARK POSITION	**POSITION-Option:** Mit der MARK POSITION-Option können Sie den Cursor an eine bestimmte Stelle, die Sie mit *operand4* angeben, innerhalb eines Feldes plazieren. Siehe auch *Beispiel 3 - REINPUT FULL mit MARK POSITION*.
operand4	**Cursor-Position:** *operand4* gibt die Cursor-Position an, *operand4* darf keine Dezimalziffern enthalten.
attributes	**Attribut-Zuweisungen:** Siehe *Attribut-Zuweisungen* weiter unten.

Attribut-Zuweisungen:



Mit dem Attribut AD=P können Sie ein Eingabefeld (AD=A oder AD=M) gegen Eingaben schützen.

Anmerkung:
Reine Ausgabefelder (AD=O) können nicht durch ein entsprechendes Attribut zu Eingabefeldern gemacht werden.

Wird AD=P auf Statement-Ebene gesetzt, so sind alle Felder geschützt außer den in der MARK-Option angegebenen.

Außerdem können Sie Anzeige- und Farbattribute von Feldern ändern. Informationen zu diesen Attributen finden Sie unter Session-Parameter AD bzw. CD in der *Parameter-Referenz*.

Siehe auch *Beispiel 2 - REINPUT mit Attribut-Zuweisung*.

ALARM-Option

[**AND**] [**SOUND**] **ALARM**

Diese Option bewirkt, dass der Warnton des Terminals ausgelöst wird, wenn das REINPUT-Statement ausgeführt wird. Voraussetzung ist, dass die verwendete Terminal-Hardware dies ermöglicht.

Beispiele

- Beispiel 1 — REINPUT-Statement
- Beispiel 2 — REINPUT mit Attribut-Zuweisung
- Beispiel 3 — REINPUT FULL mit MARK POSITION
- Beispiel 4 - mit TEXT-Option

Beispiel 1 — REINPUT-Statement

```
** Example 'REIEX1': REINPUT
************************************************************************
DEFINE DATA LOCAL
1 #FUNCTION (A1)
1 #PARM     (A1)
END-DEFINE
*
INPUT #FUNCTION #PARM
*
DECIDE FOR FIRST CONDITION
  WHEN #FUNCTION = 'A' AND #PARM = 'X'
    REINPUT 'Funktion A with parameter X selected.'
            MARK *#PARM
  WHEN #FUNCTION = 'C' THRU 'D'
    REINPUT 'Funktion C or D selected.'
  WHEN #FUNCTION = 'X'
    STOP
  WHEN NONE
    REINPUT 'Please enter a valid function.'
```

```
            MARK *#FUNCTION
END-DECIDE
*
END
```

Ausgabe des Programms REIEX1:

```
#FUNCTION A #PARM Y
```

Nach Drücken von EINGABE:

```
PLEASE ENTER A VALID FUNCTION
#FUNCTION A #PARM Y
```

Beispiel 2 — REINPUT mit Attribut-Zuweisung

```
** Example 'REIEX2': REINPUT  (with attributes)
************************************************************************
DEFINE DATA LOCAL
1 #A (A20)
1 #B (N7.2)
1 #C (A5)
1 #D (N3)
END-DEFINE
*
INPUT (AD=A) #A  #B  #C  #D
*
IF #A = ' ' OR #B = 0
  REINPUT (AD=P) 'RETYPE VALUES'
                 MARK *#A (AD=I CD=RE)    /* put cursor on first field
                      *#B (AD=U CD=PI)    /* and change colours
END-IF
*
END
```

Beispiel 3 — REINPUT FULL mit MARK POSITION

```
** Example 'REIEX3': REINPUT  (with FULL and POSITION option)
************************************************************************
DEFINE DATA LOCAL
1 #A (A20)
1 #B (N7.2)
1 #C (A5)
1 #D (N3)
END-DEFINE
*
INPUT (AD=M) #A  #B  #C  #D
*
IF #A = ' '
  COMPUTE #B = #B + #D
  RESET #D
END-IF
*
IF #A = SCAN 'TEST' OR = ' '
REINPUT FULL 'RETYPE VALUES' MARK POSITION 5 IN *#A
END-IF
*
END
```

Ausgabe des Programms REIEX3:

```
RETYPE VALUES
#A                    #B        0.00 #C       #D    0
```

Beispiel 4 - mit TEXT-Option

```
** Example 'REIEX4': REINPUT (with TEXT option)
************************************************************************
DEFINE DATA LOCAL
01 #NAME   (A8)
01 #TEXT   (A20)
END-DEFINE
*
*
INPUT WITH TEXT 'Enter a program name.' 'Program name:' #NAME
*
IF #NAME = ' '
  REINPUT WITH TEXT 'Input missing. Enter a name.'
END-IF
*
IF #NAME NE MASK (A)
  MOVE 'Invalid input.' TO #TEXT
  REINPUT WITH TEXT ':1: Name must start with a letter.',#TEXT
ELSE
  /* Using Natural error message 7600 for demonstration
  COMPRESS *INIT-USER 'on' *DAT4I INTO #TEXT
  INPUT WITH TEXT *-7600,#NAME,#TEXT 'Input accepted.'
END-IF
END
```