

Bildschirmgestaltung

Dieses Kapitel beschreibt die Möglichkeiten zur Gestaltung des Bildschirm-Layouts.

- Steuerung der Meldungszeile — Terminalkommando %M
 - Zuweisen von Farben zu Feldern — Terminalkommando %=
 - Statistikzeile/Infoline — Terminalkommando %X
 - Fenster
 - Standard-/Dynamische Layout-Maps
 - Mehrsprachige Benutzeroberflächen
 - Kenntnisabhängige Benutzeroberflächen (Expertenmodus)
-

Steuerung der Meldungszeile — Terminalkommando %M

Mit dem Terminalkommando %M geben Sie an, wie und wo die Natural-Meldungszeile angezeigt werden soll.

Im folgenden finden Sie Informationen zu:

- Positionierung der Meldungszeile
- Farbe der Meldungszeile

Positionierung der Meldungszeile

%MB

Die Meldungszeile wird am unteren Bildschirmrand angezeigt:

%MT

Die Meldungszeile wird am oberen Bildschirmrand angezeigt:

Weitere Optionen zur Positionierung der Meldungszeile sind im Abschnitt *%M - Steuerung der Meldungszeile* in der *Terminalkommandos*-Dokumentation beschrieben.

Farbe der Meldungszeile

%M=color-code

Die Meldungszeile wird in der angegebenen Farbe angezeigt (eine Beschreibung der Farbcodes finden Sie unter Session-Parameter CD in der *Parameter-Referenz*-Dokumentation).

Zuweisen von Farben zu Feldern — Terminalkommando

%=

Mit dem Terminalkommando %= können Sie bestimmten Feldern bestimmte Farben zuweisen, und zwar für Programme, die ursprünglich ohne Berücksichtigung von Farbgebung geschrieben wurden. Sie geben einen Feldtyp und/oder ein Feldattribut an sowie eine Farbe. Alle Felder/Texte dieses Typs/Attributs werden dann in dieser Farbe angezeigt.

Außerdem können Sie bestehende Farbzuweisungen ändern, falls bereits vordefinierte Farbgebungen ungeeignet sind.

Darüber hinaus können Sie das Terminalkommando %= in den Natural-Editoren benutzen, um Farben dynamisch zuzuordnen, z.B. beim Erstellen einer Maske (Map).

Codes	Beschreibung
<i>leer</i>	Bestehende Farbzuweisungen werden gelöscht.
F	Neu definierte Farbzuweisungen gelten statt denen des Programms.
N	Im Programm definierte Farbzuweisungen behalten ihre Gültigkeit.
O	Ausgabefeld (Output).
M	Modifizierbares Feld (Aus- und Eingabe).
T	Textkonstante.
B	Blinkend.
C	Kursiv.
D	Standard (Default).
I	Intensiviert.
U	Unterstrichen.
V	Invers.
BG	Bildschirmhintergrund (Background).
BL	Blau.
GR	Grün.
NE	Neutral.
PI	Rosa (Pink).
RE	Rot (Red).
TU	Türkis.
YE	Gelb (Yellow).

Beispiel:

`%=TI=RE,OB=YE`

Dieses Beispiel ordnet die Farbe Rot allen intensivierten Text-Feldern und Gelb allen blinkenden Ausgabefeldern zu.

Statistikzeile/Infoline — Terminalkommando %X

Dieses Terminalkommando steuert die Anzeige der Natural-Infoline.

Weitere Informationen finden Sie in der Beschreibung des Terminalkommandos %X in der *Terminalkommandos*-Dokumentation.

Fenster

Im folgenden finden Sie Informationen zu:

- Was ist ein Fenster?
- DEFINE WINDOW-Statement
- INPUT WINDOW-Statement

Was ist ein Fenster?

Ein *Fenster* ist jener, von einem Programm aufgebaute Abschnitt einer logischen Seite, der auf dem Terminal-Bildschirm angezeigt wird.

Eine *logische Seite* ist der Ausgabebereich für Natural; mit anderen Worten enthält die logische Seite den/die vom Natural-Programm für die Anzeige erzeugte/n Report/Map. Diese logische Seite kann breiter als der physische Bildschirm sein.

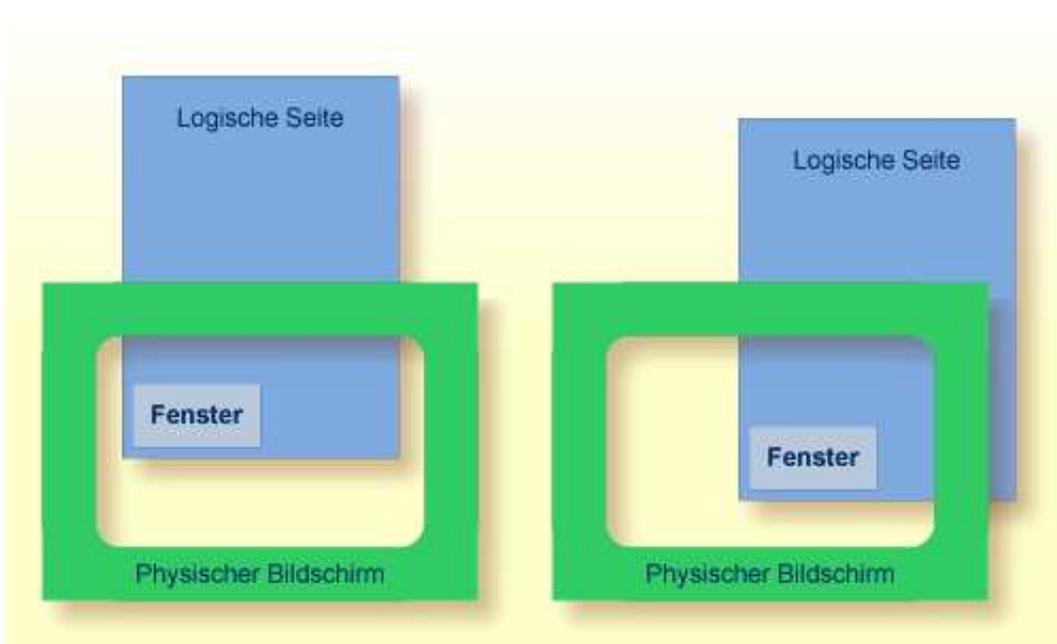
Es ist immer ein Fenster vorhanden, auch wenn dessen Vorhandensein Ihnen nicht bewusst sein mag. Wenn es (durch ein `DEFINE WINDOW`-Statement) nicht anders angegeben ist, ist die Größe des Fensters mit der physischen Größe Ihres Terminal-Bildschirms identisch.

Sie können ein Fenster auf zwei Arten handhaben:

- Sie können die Größe und Position des Fensters auf dem *physischen Bildschirm* steuern.
- Sie können die Position des Fensters auf der *logischen Seite* steuern.

Positionierung auf dem physischen Bildschirm

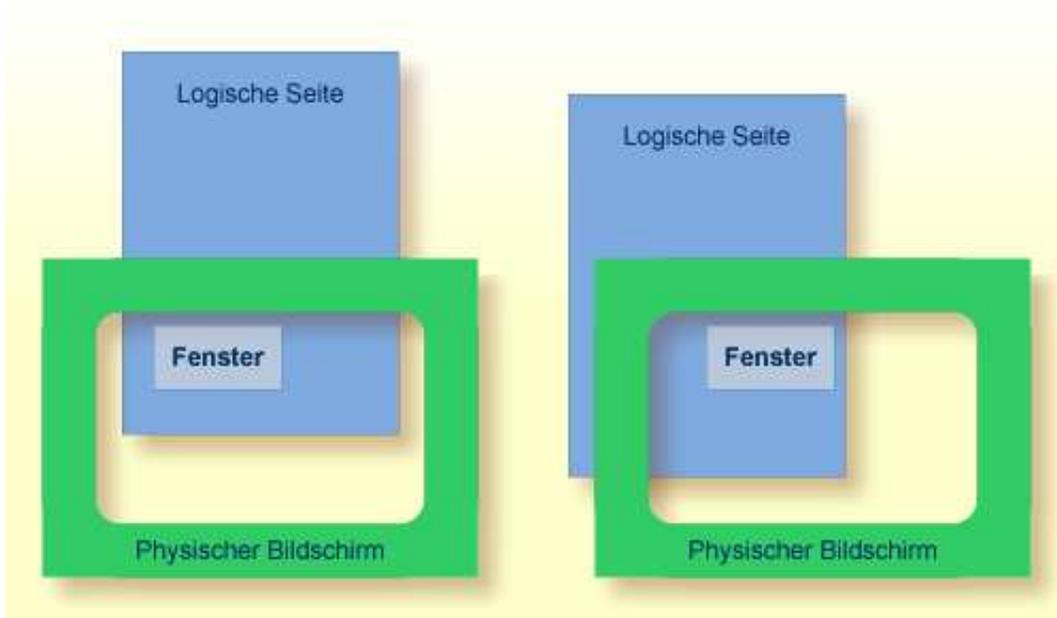
Die Abbildung unten veranschaulicht die Positionierung des Fensters auf dem physischen Bildschirm. Beachten Sie, dass in beiden Fällen der gleiche Ausschnitt der logischen Seite angezeigt wird; es wird lediglich die Position des Fensters auf dem physischen Bildschirm geändert.



Positionierung auf der logischen Seite

Die Abbildung unten veranschaulicht die Positionierung des Fensters auf der logischen Seite.

Wenn Sie die Position des Fensters auf der *logischen Seite* verändern, bleiben Position und Größe des Fensters auf dem *physischen Bildschirm* gleich; d.h. das Fenster wird nicht über der logischen Seite bewegt, sondern die logische Seite wird sozusagen "unter" dem Fenster verschoben.



DEFINE WINDOW-Statement

Das `DEFINE WINDOW`-Statement dient dazu, die Größe, Position und Attribute eines Bildschirmfensters auf dem *physischen Bildschirm* zu definieren.

Mit einem `DEFINE WINDOW`-Statement wird ein Fenster nicht aktiviert; dies geschieht mit einem `SET WINDOW`-Statement oder der `WINDOW`-Klausel eines `INPUT`-Statements.

Das `DEFINE WINDOW`-Statement bietet verschiedene Optionen. Diese werden anhand des nachstehenden Beispiels erläutert.

Das folgende Programm definiert ein Fenster auf dem physischen Bildschirm.

```
** Example 'WINDX01': DEFINE WINDOW
*****
DEFINE DATA LOCAL
1 COMMAND (A10)
END-DEFINE
*
DEFINE WINDOW TEST
    SIZE 5*25
    BASE 5/40
    TITLE 'Sample Window'
    CONTROL WINDOW
    FRAMED POSITION SYMBOL BOTTOM LEFT
*
INPUT WINDOW='TEST' WITH TEXT 'message line'
    COMMAND (AD=I'_' ) /
    'dataline 1' /
    'dataline 2' /
    'dataline 3' 'long data line'
*
IF COMMAND = 'TEST2'
    FETCH 'WINDX02'
ELSE
    IF COMMAND = '.'
        STOP
    ELSE
        REINPUT 'invalid command'
    END-IF
END-IF
END
```

Der Window-Name identifiziert das Fenster. Der Name darf bis zu 32 Stellen lang sein. Für Fensternamen gelten die gleichen Namenskonventionen wie für Benutzervariablen. Hier lautet der Name `TEST`.

Mit der `SIZE`-Klausel bestimmen Sie die Größe des Fensters. In dem Beispiel ist das Fenster 5 Zeilen hoch und 25 Spalten (Stellen) breit.

Mit der `BASE`-Klausel bestimmen Sie die Position des Fensters auf dem physischen Bildschirm. In dem Beispiel ist die obere linke Ecke des Fensters auf Zeile 5, Spalte 40 positioniert.

Mit der `TITLE`-Klausel können Sie eine Überschrift für das Fenster angeben. Die angegebene Überschrift wird zentriert in der oberen Rahmenzeile des Fensters angezeigt (natürlich nur, wenn ein Rahmen für das Fenster definiert ist).

Mit der CONTROL-Klausel können sie festlegen, ob die PF-Tastenzeile, die Meldungs- und die Statistikzeile in dem Fenster oder auf dem vollen physischen Bildschirm angezeigt werden. In diesem Fall bewirkt CONTROL WINDOW, dass die Meldungszeile im Fenster angezeigt wird. CONTROL SCREEN hingegen bewirkt, dass die Zeilen auf dem vollen physischen Bildschirm außerhalb des Fenster angezeigt werden. Wenn Sie die CONTROL-Klausel weglassen, gilt standardmäßig CONTROL WINDOW.

Mit der FRAMED-Option geben Sie an, dass das Fenster eingerahmt werden soll.

Dieser Rahmen ist dann cursor-sensitiv. Sie können gegebenenfalls im Fenster vor, zurück, nach rechts oder links blättern, indem Sie den Cursor einfach auf das entsprechende Symbol <, -, + oder > (vgl. POSITION-Klausel weiter unten) platzieren und EINGABE drücken. Anders ausgedrückt, bewegen Sie damit die logische Seite unter dem Fenster auf dem physischen Bildschirm. Werden keine Symbole angezeigt, können Sie vor- und zurückblättern, indem Sie den Cursor in die obere (zum Zurückblättern) bzw. untere (zum Vorblättern) Rahmenzeile platzieren und EINGABE drücken.

Mit der POSITION-Klausel der FRAMED-Option bestimmen Sie, dass Informationen über die Position des Fensters auf der logischen Seite in dem Fensterrahmen angezeigt werden. Diese Positionsangaben werden nur angezeigt, wenn die logische Seite größer ist als das Fenster; sonst wird die POSITION-Klausel ignoriert. Die Positionsangaben geben an, in welche Richtungen die logische Seite nach oben, unten, links und rechts über das aktuelle Fenster hinausgeht.

Wird keine POSITION-Klausel angegeben, gilt standardmäßig POSITION SYMBOL TOP RIGHT.

POSITION SYMBOL bewirkt, dass die Positionsangaben als Symbole More: < - + > angezeigt werden. Die Angaben werden in der oberen und/oder unteren Rahmenzeile angezeigt.

TOP/BOTTOM bestimmt, in welcher Rahmenzeile (oben oder unten) die Angaben erscheinen sollen.

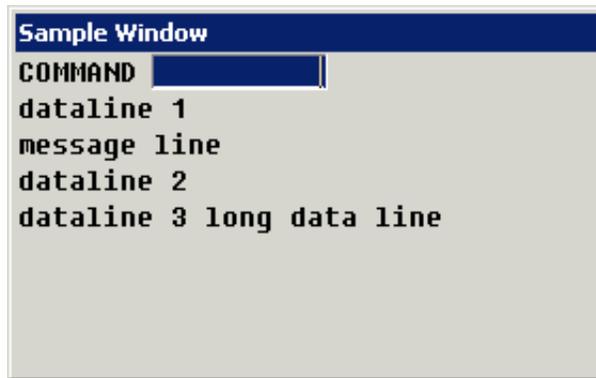
LEFT/RIGHT bestimmt, ob die Positionsangaben im linken oder rechten Teil der Rahmenzeile angezeigt werden.

INPUT WINDOW-Statement

Das INPUT WINDOW-Statement aktiviert das in dem DEFINE WINDOW-Statement definierte Fenster. In dem Beispiel wird das Fenster TEST aktiviert. Wollen Sie Daten in einem Fenster ausgeben (z.B. mit einem WRITE-Statement), benutzen Sie das SET WINDOW-Statement.

Wenn das obige Programm ausgeführt wird, wird das Fenster mit einem Eingabefeld COMMAND angezeigt. Mit dem Session-Parameter AD legen Sie fest, dass der Wert des Feldes intensiviert dargestellt und ein Unterstrich als Füllzeichen benutzt wird.

Ausgabe des Programms WINDX01:



Mehrere Fenster

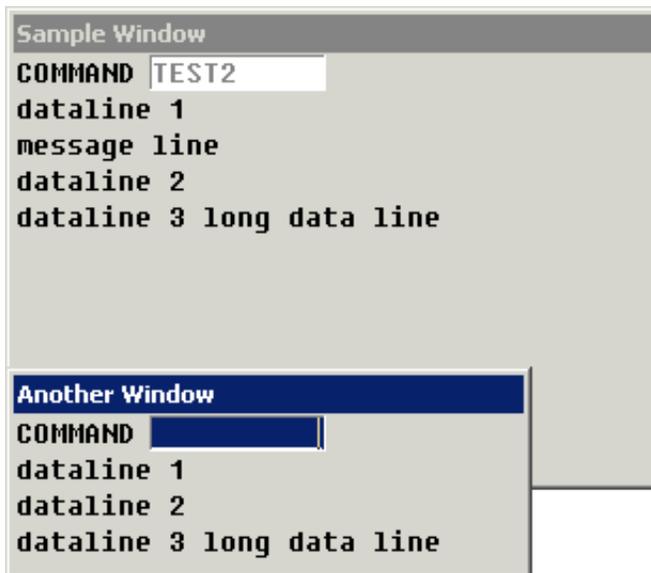
Sie können mehrere Fenster öffnen. Allerdings ist jeweils nur ein Natural-Fenster aktiv, und zwar das letzte. Vorherige Fenster mögen auf dem Bildschirm noch sichtbar sein, sind aber nicht mehr aktiv und werden von Natural ignoriert. Sie können Eingaben nur im jeweils letzten Fenster machen. Sollte der Platz hierzu nicht ausreichen, müssen Sie das Fenster vorher entsprechend vergrößern.

Wird TEST2 in das COMMAND-Feld eingegeben, wird das Programm WINDX02 ausgeführt.

```

** Example 'WINDX02': DEFINE WINDOW
*****
DEFINE DATA LOCAL
1 COMMAND (A10)
END-DEFINE
*
DEFINE WINDOW TEST2
    SIZE 5*30
    BASE 15/40
    TITLE 'Another Window'
    CONTROL SCREEN
    FRAMED POSITION SYMBOL BOTTOM LEFT
*
INPUT WINDOW='TEST2' WITH TEXT 'message line'
    COMMAND (AD=I'_' ) /
    'dataline 1' /
    'dataline 2' /
    'dataline 3' 'long data line'
*
IF COMMAND = 'TEST'
    FETCH 'WINDX01'
ELSE
    IF COMMAND = '.'
        STOP
    ELSE
        REINPUT 'invalid command'
    END-IF
END-IF
END
  
```

Ein zweites Fenster wird nun geöffnet. Das andere Fenster ist zwar noch sichtbar, jedoch inaktiv.



Beachten Sie, dass die Meldungszeile (message line) für das neue Fenster außerhalb des Fensters (unten auf dem physischen Bildschirm) angezeigt wird. Dies wurde mit der `CONTROL SCREEN`-Klausel im Programm `WINDX02` definiert.

Weitere Einzelheiten zu den Statements `DEFINE WINDOW`, `INPUT WINDOW` und `SET WINDOW` finden Sie in den entsprechenden Beschreibungen in der *Statements*-Dokumentation.

Standard-/Dynamische Layout-Maps

Standard-Layout-Maps

Im Map-Editor kann ein *Standard-Layout* definiert werden. Wird dieses Layout bei der Erstellung aller Maps verwendet, so wird gewährleistet, dass die gesamte Anwendung ein einheitliches Erscheinungsbild aufweist.

Wenn eine Map, die ein Standard-Layout referenziert, initialisiert wird, wird dieses Layout zum festen Bestandteil der Map. Falls das Standard-Layout nachträglich geändert wird, bedeutet dies allerdings, dass alle Maps neu katalogisiert werden müssen, damit die Änderungen greifen.

Dynamische Layout-Maps

Im Gegensatz zum Standard-Layout, wird ein *dynamisches Layout* nicht zum festen Bestandteil einer sich darauf beziehenden Map; vielmehr wird das Layout jeweils zur Laufzeit generiert.

Wenn Sie also auf dem Define Map Settings For MAP-Schirm im Map-Editor die Layout-Map als "dynamisch" definieren, werden alle Änderungen der Layout-Map automatisch auch bei allen Maps, die sich darauf beziehen, durchgeführt. Die Maps müssen nicht neu katalogisiert werden.

Weitere Einzelheiten über Layout-Maps finden Sie in der *Map Editor* in the *Editors*-Dokumentation.

Mehrsprachige Benutzeroberflächen

Mit Natural können Sie mehrsprachige Anwendungen für den internationalen Einsatz erstellen.

Maps, Helproutinen, Fehlermeldungen, Programme, Functions, Subprogramme und Copycodes können in bis zu 60 verschiedenen Sprachen (inklusive Sprachen, die einen Doppelbyte-Zeichensatz benutzen) definiert werden.

Im folgenden finden Sie Informationen zu:

- Sprachcodes
- Definition der Sprache eines Natural-Objektes
- Definition der Benutzersprache
- Referenzieren von mehrsprachigen Objekten
- Programme
- Fehlermeldungen
- Editiermasken für Datums- und Uhrzeitfelder

Sprachcodes

In Natural hat jede Sprache einen *Sprachcode* (von 1 bis 60). Die folgende Tabelle ist ein Auszug der Gesamttabelle der Sprachcodes.

Eine vollständige Aufstellung der Sprachcodes finden Sie in der Beschreibung der Systemvariablen *LANGUAGE in der *Systemvariablen*-Dokumentation.

Sprachcode	Sprache	Mapcode in Objekt-Namen
1	Englisch	1
2	Deutsch	2
3	Französisch	3
4	Spanisch	4
5	Italienisch	5
6	Niederländisch	6
7	Türkisch	7
8	Dänisch	8
9	Norwegisch	9
10	Albanisch	A
11	Portugiesisch	B

Der Sprachcode (linke Spalte) ist der Code, der in der Systemvariable *LANGUAGE enthalten ist. Dieser Code wird von Natural intern benutzt. Es ist der Code, den Sie zur Definition der Benutzersprache benutzen (siehe *Definition der Benutzersprache* weiter unten). Der Code, den Sie zur Identifikation der Sprache eines Natural-Objekts angeben, ist der *Mapcode* in der rechten Spalte der Tabelle.

Beispiel:

Der Sprachcode für Portugiesisch ist 11. Der Code, den Sie beim Katalogisieren eines portugiesischen Natural-Objekts angeben, ist B.

Definition der Sprache eines Natural-Objektes

Sie definieren die Sprache eines Natural-Objektes (Map, Helproutine, Programm, Function, Subprogramm oder Copycode), indem Sie den entsprechenden Mapcode dem Objektnamen hinzufügen. Bis auf diesen Mapcode muss der Objektname identisch für alle Sprachen sein.

Beim folgenden Beispiel wurden zwei Maps erzeugt, und zwar eine englische und eine deutsche. Um die Sprachen der Maps zu identifizieren, wurde der entsprechende Mapcode jeweils den Mapnamen hinzugefügt.

Beispiel für Mapnamen bei einer mehrsprachigen Anwendung:

DEMO1 = englische Map (Mapcode 1)

DEMO2 = deutsche Map (Mapcode 2)

Definition von Sprachen mit alphabetischen Mapcodes

Mapcodes können im Bereich 1–9, A–Z oder a–y liegen. Die alphabetischen Mapcodes bedürfen einer besonderen Handhabung.

Normalerweise ist es nicht möglich, ein Objekt zu katalogisieren, das einen Kleinbuchstaben im Namen hat — alle Buchstaben werden automatisch in Großbuchstaben umgewandelt.

Genau dies ist aber erforderlich, wenn Sie z.B. ein Objekt als japanisch (Kanji) definieren wollen. Japanisch hat den Sprachcode 59 und den Mapcode x.

Um ein solches Objekt zu katalogisieren, müssen Sie zuerst den Sprachcode (in diesem Fall Sprachcode 59) richtig setzen, indem Sie das Terminalkommando %L=nn (nn entspricht dem Sprachcode) benutzen.

Jetzt können Sie das Objekt katalogisieren, wobei Sie das Und-Zeichen (&) anstelle des eigentlichen Mapcodes im Objektnamen benutzen. Um eine japanische Version der Map DEMO zu erhalten, katalogisieren Sie die Map also unter dem Namen DEMO&.

Wenn Sie jetzt in der Liste der Natural-Objekte nachschauen, wird die Map richtigerweise als DEMOx aufgelistet.

Sie können Objekte mit Sprachcode 1 bis 9 und A bis Z direkt katalogisieren, d.h. ohne die &-Notation.

Im nachfolgenden Beispiel sehen Sie die drei Maps DEMO1, DEMO2 und DEMOx. Um die Map DEMOx zu löschen, benutzen Sie die gleiche Technik wie bei der Definition des Mapnamens, d.h. Sie setzen zuerst die richtige Sprache mit dem Terminalkommando %L=59, dann bestätigen Sie den Löschvorgang mit der &-Notation (DEMO&).

Definition der Benutzersprache

Sie können pro Benutzer bestimmen, welche Sprache (wie in der Systemvariablen `*LANGUAGE` definiert) benutzt wird, und zwar mit dem Profilparameter `ULANG` (der in der *Parameter-Referenz*-Dokumentation beschrieben ist) oder mit dem Terminalkommando `%L=nn` (wobei `nn` der Sprachcode ist).

Referenzieren von mehrsprachigen Objekten

Um in einem Programm mehrsprachige Objekte zu referenzieren, benutzen Sie das `&`-Zeichen im Namen des Objektes.

Das Programm unten benutzt die zwei Maps `DEMO1` und `DEMO2`. Das `&`-Zeichen am Ende des Mapnamens steht anstelle des Mapcodes und bedeutet, dass die Map mit der Sprache, die dem Wert in der Systemvariable `*LANGUAGE` entspricht, benutzt werden soll.

```
DEFINE DATA LOCAL
1 PERSONNEL VIEW OF EMPLOYEES
  2 NAME (A20)
  2 PERSONNEL-ID (A8)
1 CAR VIEW OF VEHICLES
  2 REG-NUM (A15)
1 #CODE (N1)
END-DEFINE
*
INPUT USING MAP 'DEMO&' /* <--- INVOKE MAP WITH CURRENT LANGUAGE CODE
...
```

Wird dieses Programm ausgeführt, so wird die englische Map (`DEMO1`) ausgegeben, denn der Wert von `*LANGUAGE` ist `1` = englisch.

```
MAP DEMO1

SAMPLE MAP

Please select a function!

1.) Employee information
2.) Vehicle information

Enter code here: _
```

Im Beispiel unten wird der Sprachcode auf `2` = deutsch umgesetzt und zwar mit dem Terminalkommando `%L=2`.

Wird das Programm nun ausgeführt, wird die deutsche Map (`DEMO2`) ausgegeben.

BEISPIEL-MAP

Bitte wählen Sie eine Funktion!

1.) Mitarbeiterdaten

2.) Fahrzeugdaten

Code hier eingeben: _

Programme

Bei manchen Anwendungen kann es nützlich sein, mehrsprachige Programme zu definieren. Ein Fakturierungsprogramm könnte z.B. verschiedene Unterprogramme benutzen, um gewisse steuerliche Aspekte zu berücksichtigen, je nachdem in welchem Land die Rechnung erstellt werden soll.

Mehrsprachige Programme werden genauso definiert wie Maps (siehe Beschreibung oben).

Fehlermeldungen

Mit der Natural-Utility *SYSERR* können Sie die Natural-Fehlermeldungen in bis zu 60 Sprachen übersetzen. Sie können aber auch eigene Fehlermeldungen erstellen.

Die Sprache der ausgegebenen Fehlermeldungen wird durch den Inhalt der Systemvariable `*LANGUAGE` bestimmt.

Einzelheiten über Fehlermeldungen finden Sie in der *SYSERR Utility* in der *Utilities*-Dokumentation.

Editiermasken für Datums- und Uhrzeitfelder

Die Sprache für Datums- und Uhrzeitfelder, die mit Editiermasken definiert wurden, wird auch durch die Systemvariable `*LANGUAGE` festgelegt.

Weitere Einzelheiten zu Editiermasken entnehmen Sie dem Session-Parameter `EM` in der *Parameter Reference*.

Kenntnisabhängige Benutzeroberflächen (Expertenmodus)

Es kann sinnvoll sein, Benutzern mit unterschiedlicher Erfahrung bei der Benutzung derselben Anwendung verschiedene Maps mit unterschiedlichem Informationsgehalt zu bieten.

Ist Ihre Anwendung *nicht* für den internationalen Einsatz bestimmt, können Sie die gleichen Techniken, die zur Unterstützung von mehrsprachigen Anwendungen benutzt werden, auch zur Definition von unterschiedlich detaillierten Maps verwenden.

Sie können z.B. Sprachcode 1 als Kenntnisstufe 1 = Kenntnisstand eines Anfängers, und Sprachcode 2 als Kenntnisstufe 2 = Kenntnisstand eines fortgeschrittenen Benutzers definieren. Die Anwendung dieser einfachen Technik wird im Folgenden veranschaulicht.

Die folgende Map (PERS1) enthält ausführliche Anweisungen, die dem Endbenutzer sagen, wie eine Funktion ausgewählt wird. Die Informationen sind sehr detailliert. Der Name dieser Map enthält den Mapcode 1.

```
MAP PERS1

SAMPLE MAP

Please select a function

1.) Employee information _
2.) Vehicle information _

Enter code: _

To select a function, do one of the following:

- place the cursor on the input field next to desired function and press ENTER
- mark the input field next to desired function with an X and press ENTER
- enter the desired function code (1 or 2) in the 'Enter code' field and press
  ENTER
```

Die gleiche Map - aber ohne die ausführlichen Anweisungen - wird unter dem gleichen Namen, aber mit Mapcode 2 gespeichert.

```
MAP PERS2

SAMPLE MAP

Please select a function

1.) Employee information _
2.) Vehicle information _

Enter code: _
```

In dem obigen Beispiel wird die Map mit den vollständigen Anweisungen dann ausgegeben, wenn der ULANG-Profilparameter auf 1 gesetzt ist und die Map ohne die Anweisungen, wenn er auf 2 gesetzt ist.