Bildschirmgestaltung

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

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

Steuerung der Funktionstastenleiste — Terminalkommando %Y

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

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- Format der Funktionstastenleiste
- Positionierung der Funktionstastenleiste
- Cursor-Sensitivität

Format der Funktionstastenleiste

Die folgenden Terminalkommandos legen das Format der Funktionstastenleiste fest:

%YN

Die Funktionstastenleiste wird im tabularischen Software-AG-Format angezeigt:

```
  Command ===>

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
            Help        Exit                                          Canc
```

%YS

Die Funktionstastenleiste zeigt die einzelnen Tasten nacheinander, und zwar nur die Tasten, denen Namen zugewiesen wurden (PF1=*Wert*,PF2=*Wert* usw.):

```
Command ===>
PF1=Help,PF3=Exit,PF12=Canc
```

%YP

Die Funktionstastenleiste erscheint im PC-Format, d.h. die einzelnen Tasten werden nacheinander angezeigt, und zwar nur die Tasten, denen Namen zugewiesen wurden (F1=*Wert*,F2=*Wert* usw.):

```
Command ===>
F1=Help,F3=Exit,F12=Canc
```

Weitere Anzeige-Optionen

Es stehen verschiedene andere Optionen zur Anzeige der Funktionstastenleiste zur Verfügung, wie:

- ein- und zweizeilige Darstellung
- intensivierte Darstellung
- inverse Darstellung
- farbige Darstellung

Einzelheiten zu diesen Optionen entnehmen Sie dem Abschnitt *%Y - Steuerung der PF-Tastenleiste* in der *Terminalkommandos*-Dokumentation.

Positionierung der Funktionstastenleiste

%YB

Die Funktionstastenleiste wird am unteren Bildschirmrand angezeigt:

```
16:50:53                  *****  NATURAL  *****                  2002-12-18
User SAG                      - Main Menu -               Library XYZ


                  Function

               _ Development Functions
               _ Development Environment Settings
               _ Maintenance and Transfer Utilities
               _ Debugging and Monitoring Utilities
               _ Example Libraries
               _ Other Products
               _ Help
               _ Exit Natural Session




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

%YT

Die Funktionstastenleiste wird am oberen Bildschirmrand angezeigt:

```
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Canc
16:50:53                    *****  NATURAL  *****                   2002-12-18
User SAG                        - Main Menu -                Library XYZ


                    Function

                  _ Development Functions
                  _ Development Environment Settings
                  _ Maintenance and Transfer Utilities
                  _ Debugging and Monitoring Utilities
                  _ Example Libraries
                  _ Other Products
                  _ Help
                  _ Exit Natural Session









Command ===>
```

%Y*nn*

Die Funktionstastenleiste wird in der *nn*-ten Zeile des Bildschirms angezeigt. Im nachfolgenden Beispiel befindet sich die Funktionstastenleiste in Zeile 10:

```
16:50:53                    *****  NATURAL  *****                   2002-12-18
User SAG                        - Main Menu -                Library XYZ


                    Function

                  _ Development Functions
                  _ Development Environment Settings
                  _ Maintenance and Transfer Utilities
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Canc
                  - Debugging and Monitoring Utilities
                  _ Example Libraries
                  _ Other Products
                  _ Help
                  _ Exit Natural Session









Command ===>
```

Cursor-Sensitivität

%YC

Dieses Kommando macht die Funktionstastenleiste cursor-sensitiv. Sie reagiert dann wie eine Aktionsleiste auf einem PC-Bildschirm: der Benutzer wählt mit dem Cursor lediglich den Namen oder die Nummer der gewünschten Funktionstaste aus und drückt EINGABE, und Natural reagiert, als ob die betreffende Funktionstaste gedrückt worden wäre.

Durch nochmaliges Eingeben von `%YC` schalten Sie die Cursor-Sensitivität wieder aus.

Durch Verwendung von `%YC` in Verbindung mit tabellarischem Anzeigeformat (`%YN`) und einzeiliger Anzeige (`%YH`) können Sie Ihre Anwendungen mit einer sehr komfortablen Aktionsleisten-Verarbeitung ausstatten: der Benutzer wählt dann nur noch den Namen einer Funktion mit dem Cursor aus und drückt EINGABE, und die Funktion wird ausgeführt.

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
- Schützen der Meldungszeile
- Farbe der Meldungszeile

Positionierung der Meldungszeile

%MB

Die Meldungszeile wird am unteren Bildschirmrand angezeigt:

```
16:50:53                    *****  NATURAL  *****                   2002-12-18
User SAG                        - Main Menu -                Library XYZ


                    Function

                  _ Development Functions
                  _ Development Environment Settings
                  _ Maintenance and Transfer Utilities
                  _ Debugging and Monitoring Utilities
                  _ Example Libraries
                  _ Other Products
                  _ Help
                  _ Exit Natural Session




Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                  Canc
Please enter a function.
```

%MT

Die Meldungszeile wird am oberen Bildschirmrand angezeigt:

```
Please enter a function.
16:50:53                    *****  NATURAL  *****                   2002-12-18
User SAG                        - Main Menu -                Library XYZ


                    Function

                  _ Development Functions
                  _ Development Environment Settings
                  _ Maintenance and Transfer Utilities
                  _ Debugging and Monitoring Utilities
                  _ Example Libraries
                  _ Other Products
                  _ Help
                  _ Exit Natural Session




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

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

Schützen der Meldungszeile

%MP

Der Schutz der Meldungszeile wird ein- bzw. ausgeschaltet. Ist die Meldungszeile nicht geschützt, kann sie auch für Bildschirmeingaben benutzt werden.

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 zuordnen, z.B. beim Erstellen einer Maske (Map).

Codes	Beschreibung
leer	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.

Outlining (Umrahmung) — Terminalkommando %D=B

"Outlining" (Boxing) ist die Möglichkeit, bestimmte Felder auf dem Bildschirm "eingerahmt" (d.h. von einer Linie umgeben) anzuzeigen. Diese Form der Anzeige ist eine weitere Möglichkeit, dem Benutzer Länge und Position von Feldern auf dem Bildschirm deutlich zu machen.

"Outlining" ist nur auf bestimmten Terminaltypen möglich, in der Regel auf solchen, die auch Doppelbyte-Zeichensätze (Kanji) unterstützen.

Mit dem Terminalkommando `%D=B` steuern Sie das "Outlining". Einzelheiten zu diesem Kommando entnehmen Sie dem entsprechenden Abschnitt in der *Terminalkommandos*-Dokumentation.

Statistikzeile/Infoline — Terminalkommando %X

Dieses Terminalkommando steuert die Anzeige der Natural-Statistikzeile/Infoline. Die Zeile kann als Statistikzeile oder als Infoline benutzt werden, aber nicht beides gleichzeitig.

Im Folgenden finden Sie Informationen zu:

- Statistikzeile
- Infoline

Statistikzeile

%X schaltet die Anzeige der Statistikzeile/Infoline ein bzw. wieder aus. Schalten Sie die Statistikzeile ein, so können Sie statistische Informationen sehen wie:

- die Anzahl der während der letzten Bildschirmoperation an den Bildschirm übergebenen Bytes,
- die logische Zeilenlänge der aktuellen logischen Seite,
- die physische Zeilenlänge des aktiven Natural-Fensters.

Weitere Einzelheiten zur Statistikzeile können Sie der Beschreibung des Terminalkommandos %X in der *Terminalkommandos*-Dokumentation.

Das nachstehende Beispiel zeigt die Statistikzeile, wie sie unten am Bildschirmrand angezeigt wird:

```
 >                                      > +  Program     POS      Lib SAG
 All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
   0010 SET CONTROL 'XT'
   0020 SET CONTROL 'XI+'
   0030 DEFINE PRINTER (2) OUTPUT 'INFOLINE'
   0040 WRITE (2) 'EXECUTING' *PROGRAM 'BY' *INIT-USER
   0050 WRITE 'TEST OUTPUT'
   0070 END
   0080
   0090
   0100
   0110
   0120
   0130
   0140
   0150
   0160
   0170
   0180
   0190
   0200
IO=264,AI =292,L=0 C= ,LS=80,P =23,PLS=80,PCS=24,FLD=82,CLS=1,ADA=0
```

Infoline

Sie können die Statistikzeile auch als *Infoline* benutzen, in der Sie Status-Informationen ausgeben können, z.B. bei der Fehlersuche. Alternativ können Sie die Infoline als Trennlinie (wie in den SAA-Standards definiert) verwenden.

`%XI+` definiert die Statistikzeile als Infoline.

Sobald Sie die Infoline mit dem oben erwähnten Kommando aktiviert haben, können Sie die Infoline als Ausgabemedium für Daten mit dem `DEFINE PRINTER`-Statement definieren, wie im folgenden Beispiel veranschaulicht:

```
SET CONTROL 'XT'
SET CONTROL 'XI+'
DEFINE PRINTER (2) OUTPUT 'INFOLINE'
WRITE (2) 'EXECUTING' *PROGRAM 'BY' *INIT-USER
WRITE 'TEST OUTPUT'
END
```

Wenn dieses Programm gestartet wird, werden die Status-Informationen in der Infoline am oberen Rand des Ausgabeschirms angezeigt:

```
EXECUTING POS      BY SAG
Page      1                                          2001-01-22  10:56:06

TEST OUTPUT
```

Weitere Einzelheiten zur Statistikzeile/Infoline, siehe das Terminalkommando `%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.

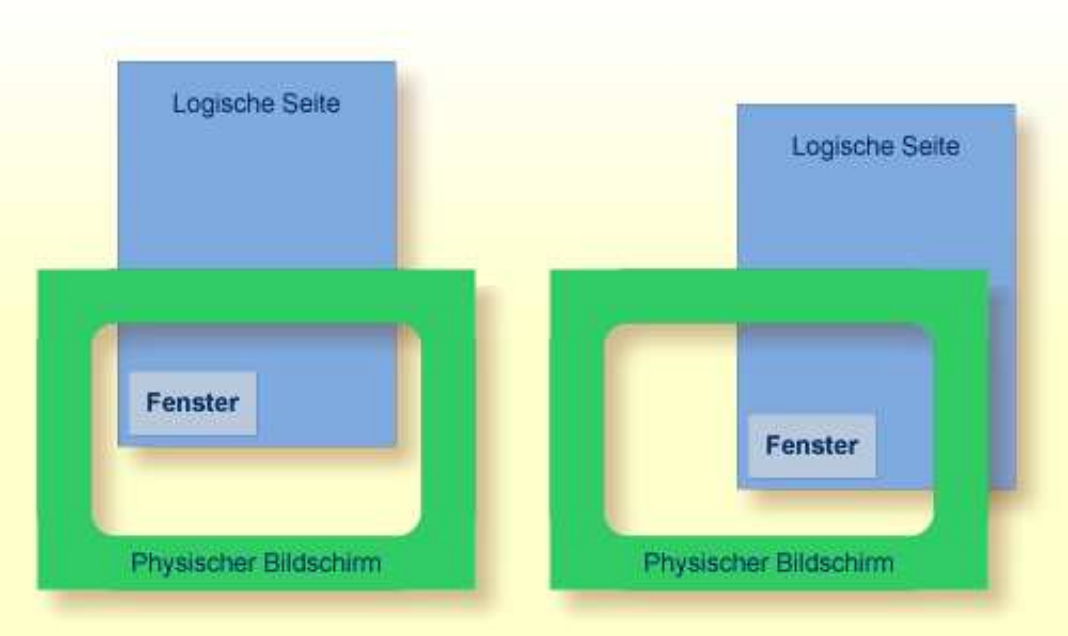
Es ist immer ein Fenster vorhanden, auch wenn dessen Vorhandensein Ihnen nicht bewusst sein mag. Wenn es (durch ein `DEFINE WINDOW`-Statement) nicht anders angegeben ist, ist die Größe des Fensters mit der physischen Größe Ihres Terminal-Bildschirms identisch.

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

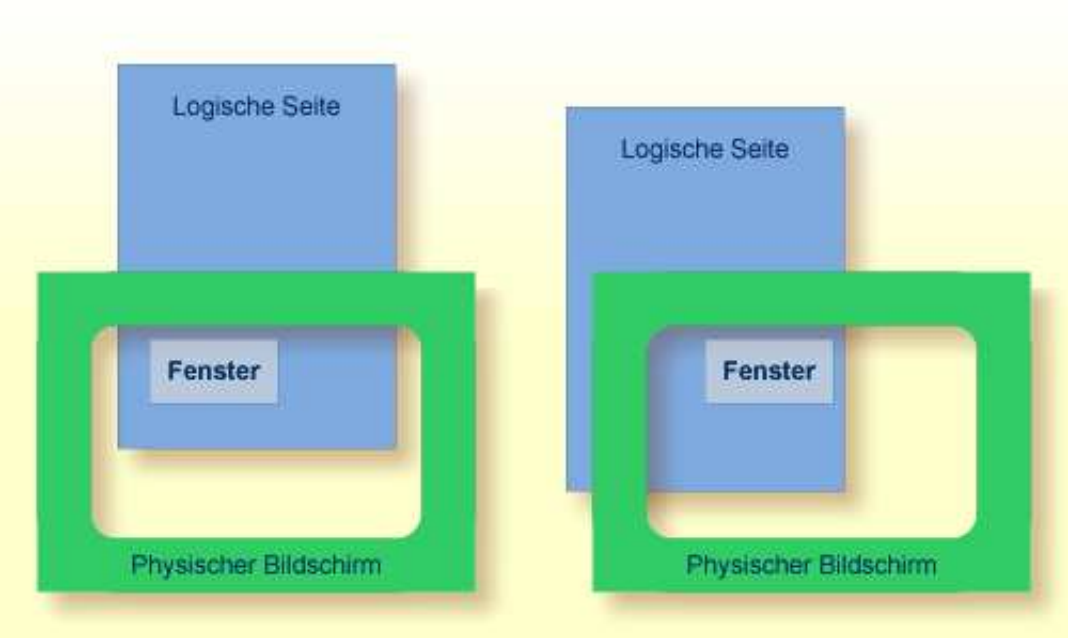
Die Abbildung unten veranschaulicht die Positionierung des Fensters auf dem physischen Bildschirm. Beachten Sie, dass in beiden Fällen der gleiche Ausschnitt der logischen Seite angezeigt wird; es wird lediglich die Position des Fensters auf dem physischen Bildschirm geändert.



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. Es bezieht sich auf die standardmäßige Terminaltyp-Einstellung von Natural; siehe auch Terminalkommando `%T=` und Profilparameter `TTYPE`).

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.

Mit dem Terminalkommando %F=*chv* bestimmen Sie, aus welchen Zeichen der Fensterrahmen zusammengesetzt werden soll.

c	Das erste Zeichen wird für die vier *Ecken* (**c**orners) des Rahmens verwendet.
h	Das zweite Zeichen wird für die *horizontalen* Linien des Rahmens verwendet.
v	Das dritte Zeichen wird für die *vertikalen* Linien des Rahmens verwendet.

Beispiel:

```
%F=+-!
```

Der mit dem obigen Kommando definierte Fensterrahmen sieht wie folgt aus:

```
+------------------------+
!                        !
!                        !
!                        !
!                        !
+------------------------+
```

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:

```
> r                                    > +  Program     WINDX01  Lib SYSEXPG
Top    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 ** Example 'WINDX01': DEFINE WINDOW
  0020 ****************************** +----Sample Window-----+ ***************
  0030 DEFINE DATA LOCAL               ! message line         !
  0040 1 COMMAND (A10)                 ! COMMAND _____________ !
  0050 END-DEFINE                      ! dataline 1           !
  0060 *                               +More:     + >---------+
  0070 DEFINE WINDOW TEST
  0080        SIZE 5*25
  0090        BASE 5/40
  0100        TITLE 'Sample Window'
  0110        CONTROL WINDOW
  0120        FRAMED POSITION SYMBOL BOTTOM LEFT
  0130 *
  0140 INPUT WINDOW='TEST' WITH TEXT 'message line'
  0150       COMMAND (AD=I'_') /
  0160       'dataline 1' /
  0170       'dataline 2' /
  0180       'dataline 3' 'long data line'
  0190 *
  0200 IF COMMAND = 'TEST2'
       ....+....1....+....2....+....3....+....4....+....5....+... S 29   L 1
```

Die in der unteren Rahmenzeile erscheinenden Positionsangaben `More + >` zeigen an, dass die logische Seite mehr Informationen enthält, als im Fenster zu sehen sind.

Um die Informationen anzuzeigen, die sich weiter unten auf der logischen Seite befinden, platzieren Sie den Cursor in die untere Rahmenzeile auf das +-Symbol und drücken EINGABE.

Dadurch wird das Fenster nach unten bewegt. Beachten Sie, dass der Text `long data line` (lange Datenzeile) nicht in das Fenster passt und daher nicht vollständig angezeigt wird.

```
> r                                    > +  Program     WINDX01  Lib SYSEXPG
Top    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 ** Example 'WINDX01': DEFINE WINDOW
  0020 ****************************** +----Sample Window-----+ ***************
  0030 DEFINE DATA LOCAL               ! message line         !
  0040 1 COMMAND (A10)                 ! dataline 3 long data !
  0050 END-DEFINE                      ! dataline 2           !
  0060 *                               +More:   -   >---------+
  0070 DEFINE WINDOW TEST
  0080        SIZE 5*25
  0090        BASE 5/40
  0100        TITLE 'Sample Window'
  0110        CONTROL WINDOW
  0120        FRAMED POSITION SYMBOL BOTTOM LEFT
  0130 *
  0140 INPUT WINDOW='TEST' WITH TEXT 'message line'
  0150       COMMAND (AD=I'_') /
  0160       'dataline 1' /
  0170       'dataline 2' /
  0180       'dataline 3' 'long data line'
  0190 *
  0200 IF COMMAND = 'TEST2'
       ....+....1....+....2....+....3....+....4....+....5....+... S 29   L 1
```

Um diese *verborgenen* Informationen auf der rechten Seite zu sehen, platzieren Sie den Cursor in die untere Rahmenzeile auf das >-Symbol und drücken EINGABE. Das Fenster wird dadurch auf der logischen Seite nach rechts bewegt; das vorher unsichtbare Wort `line` (Zeile) wird angezeigt:

```
> r                                    > +  Program     WINDX01  Lib SYSEXPG
Top    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 ** Example 'WINDX01': DEFINE WINDOW
  0020 ****************************** +----Sample Window-----+ ***************
  0030 DEFINE DATA LOCAL               ! message line         !
  0040 1 COMMAND (A10)                 !  line                ! <==
  0050 END-DEFINE                      !                      !
  0060 *                               +More: < -    ---------+
  0070 DEFINE WINDOW TEST
  0080        SIZE 5*25
  0090        BASE 5/40
  0100        TITLE 'Sample Window'
  0110        CONTROL WINDOW
  0120        FRAMED POSITION SYMBOL BOTTOM LEFT
  0130 *
  0140 INPUT WINDOW='TEST' WITH TEXT 'message line'
  0150       COMMAND (AD=I'_') /
  0160       'dataline 1' /
  0170       'dataline 2' /
  0180       'dataline 3' 'long data line'
  0190 *
  0200 IF COMMAND = 'TEST2'
       ....+....1....+....2....+....3....+....4....+....5....+... S 29   L 1
```

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.

```
message line
> r                                    > +  Program     WINDX01  Lib SYSEXPG
Top    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 ** Example 'WINDX01': DEFINE WINDOW
  0020 ***************************** +----Sample Window-----+ ***************
  0030 DEFINE DATA LOCAL               ! message line         !  Inactive
  0040 1 COMMAND (A10)                 ! COMMAND TEST2_____   !  Window
  0050 END-DEFINE                      ! dataline 1           !  <==
  0060 *                               +More:     + >---------+
  0070 DEFINE WINDOW TEST
  0080        SIZE 5*25
  0090        BASE 5/40
  0100        TITLE 'Sample Window'
  0110        CONTROL WINDOW
  0120        FRAMED POSITION SYMBOL B +------Another Window-------+  Currently
  0130 *                               ! COMMAND _____________        !  Active
  0140 INPUT WINDOW='TEST' WITH TEXT ' ! dataline 1                   !  Window
  0150       COMMAND (AD=I'_') /       ! dataline 2                   !  <==
  0160       'dataline 1' /            +More:   +------------------+
  0170       'dataline 2' /
  0180       'dataline 3' 'long data line'
  0190 *
  0200 IF COMMAND = 'TEST2'
```

Beachten Sie, dass die Meldungszeile (message line) für das neue Fenster außerhalb des Fensters (oben 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

Wie im *Map Editor Tutorial* beschrieben, kann im Map-Editor 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 (siehe Beispiel unten), werden alle Änderungen der Layout-Map automatisch auch bei allen Maps, die sich darauf beziehen, durchgeführt. Die Maps müssen nicht neu katalogisiert werden.

```
08:46:18              Define Map Settings for MAP                  2001-01-22

 Delimiters           Format                          Context
 -----------------    --------------------------      -------------------------
 Cls Att CD  Del      Page Size ..... 23              Device Check .... ________
  T   D      BLANK    Line Size ..... 79              WRITE Statement   _
  T   I      ?        Column Shift ... 0 (0/1)        INPUT Statement   X
  A   D      _        Layout ......... STAN1___       Help ____________________
  A   I      )         dynamic ....... Y (Y/N)         as field default N (Y/N)
  A   N      a        Zero Print ..... N (Y/N)
  M   D      &        Case Default ... UC (UC/LC)
  M   I      :        Manual Skip .... N (Y/N)        Automatic Rule Rank  1
  O   D      +        Decimal Char ... .              Profile Name .... SYSPROF
  O   I      (        Standard Keys .. Y (Y/N)
                      Justification .. L (L/R)        Filler Characters
                      Print Mode ..... __             ------------------------
                                                      Optional, Partial .... _
                      Control Var .... ________       Required, Partial .... _
                                                      Optional, Complete ... _
 Apply changes only to new fields?     N (Y/N)        Required, Complete ... _

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                  Let
```

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, 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, 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&).

```
08:41:14              ***** NATURAL LIST COMMAND *****             2001-01-25
User SAG            LIST * *                                  Library SAG

Cmd  Name      Type         S/C  SM  Vers  Level  User-ID   Date      Time
---  --------  -----------  ---  --  ----  -----  --------  --------  --------
__   COM3      Program      S/C  S   2.2   0002   SAG       92-01-21  14:34:39
__   CUR       Program  +--------- DELETE ---------+        92-01-22  09:37:02
__   CURS      Map      !                           !        92-01-22  09:37:41
__   D         Program  !  Please confirm deletion !        92-01-21  14:13:14
__   DARL      Program  !  with name DEMOx          !        91-06-03  12:08:30
__   DARL1     Local    !            DEMO&___       !        91-06-03  12:03:52
__   DAV       Program  +---------------------------+        92-01-29  09:07:52
de   DEMOx     Map          S/C  S   2.2   0002   SAG       92-02-25  08:41:04
__   DEMO1     Map          S/C  S   2.2   0002   SAG       92-01-22  08:38:32
__   DEMO2     Map          S/C  S   2.2   0002   SAG       92-01-22  08:07:32
__   DOWNCOM   Program      S    S   2.2   0001   SAG       91-08-12  14:01:10
__   DOWNCOMR  Program      S    S   2.2   0001   SAG       91-08-12  14:01:32
__   DOWNCOM2  Program      S    S   2.2   0001   SAG       91-08-15  13:02:20
__   DOWNDIR   Program      S    S   2.2   0001   SAG       91-08-16  08:03:56
From _________ (New start value)                                 0

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

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).

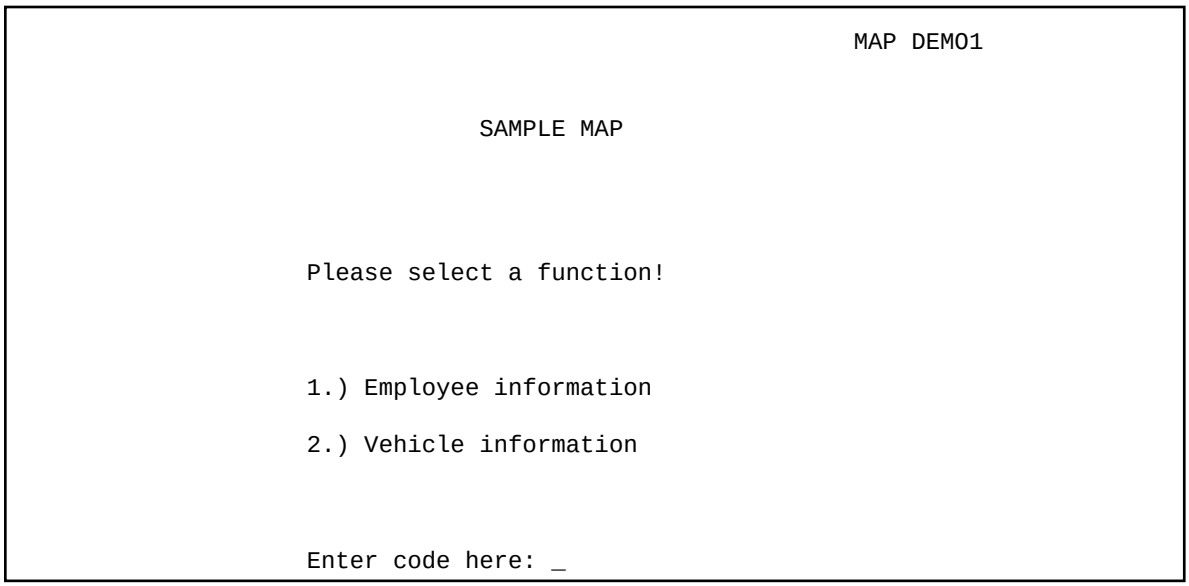
Referenzieren von mehrsprachigen Objekten

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.

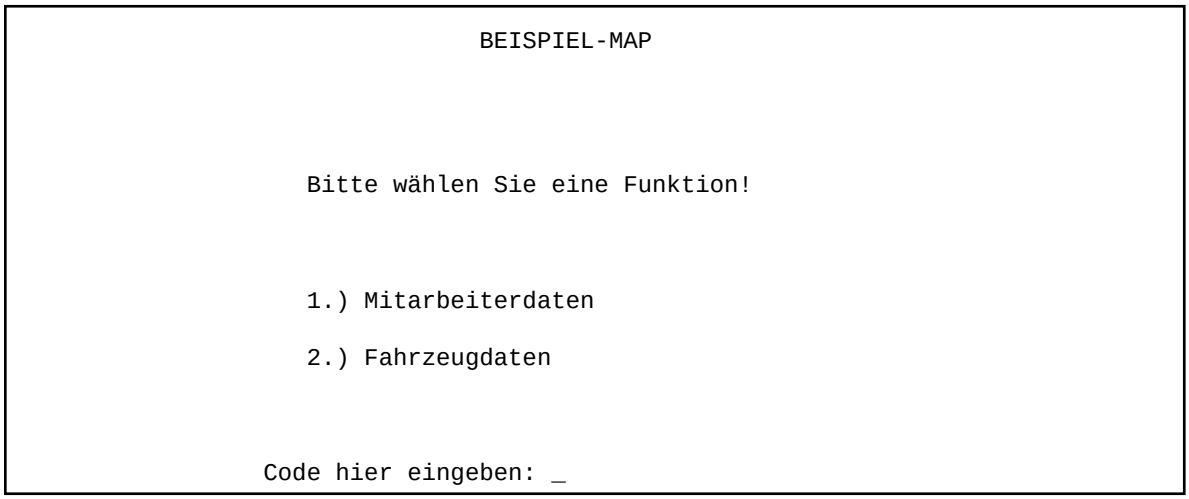
```
DEFINE DATA LOCAL
1 PERSONNEL VIEW OF EMPLOYEES
  2 NAME (A20)
  2 PERSONNEL-ID (A8)
1 CAR VIEW OF VEHICLES
   2 REG-NUM (A15)
1 #CODE (N1)
END-DEFINE
*
INPUT USING MAP 'DEMO&' /* <--- INVOKE MAP WITH CURRENT LANGUAGE CODE
...
```

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



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.



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.