SIEMENS

FORMANT

Maskenunterstützung für Bildschirmeinheiten und Drucker

SINIX

für Siemens PC

Betriebssystem SINIX

FORMANT

Maskenunterstützung für Bildschirmeinheiten und Drucker

Ausgabe September 1985 (FORMANT Version 1.0B)

Bestell-Nr. U2420-J-Z95-1
Printed in the Federal Republic of Germany
5600AG5860.8(7000)

Im Laufe der Entwicklung des Produktes können aus technischen oder wirtschaftlichen Gründen Leistungsmerkmale hinzugefügt bzw. geändert werden oder entfallen. Entsprechendes gilt für andere Angaben in dieser Druckschrift.

Siemens Aktiengesellschaft

Vorwort

FORMANT ist eine Maskensteuerung für alle SINIX-Systeme. Dieses Handbuch beschreibt, was FORMANT leistet und wie man FORMANT einsetzt. Es enthält:

- eine Einführung in FORMANT,
- die Beschreibung der Bedienerschnittstelle von FORMANT,
- die Beschreibung von FORMANTGEN, dem Programm mit dem man Masken erstellt und verwaltet,
- die Beschreibung der Programmschnittstellen in COBOL und in C,
- viele Beispiele zur Programmierung.

Wie lernen Sie FORMANT kennen?

Die ersten 3 Kapitel erfordern keine besonderen Kenntnisse. Sie richten sich an alle, die FORMANT einsetzen oder sich über die Möglichkeiten von FORMANT nur informieren wollen. Wenn Sie diese Kapitel gelesen haben, kennen Sie FORMANT soweit, daß Sie eine FORMANT-Anwendung planen können.

Wenn Sie eine FORMANT-Anwendung schreiben wollen...

dann brauchen Sie zu den Informationen der ersten Kapitel die ausführliche Beschreibung der FORMANT-Funktionen. Diese finden Sie in den Kapiteln 5 und 6 in alphabetischer Reihenfolge.

Um eine FORMANT-Anwendung schreiben zu können, müssen Sie eine der Programmiersprachen COBOL oder C beherrschen. Außerdem brauchen Sie Grundkenntnisse des Betriebssystems SINIX.

Kapitel 4 enthält ausführliche Programmbeispiele in COBOL und in C. Sie zeigen die wichtigsten Funktionen von FORMANT und sollen Ihnen den Einstieg erleichtern.

Eine Bitte an Sie

Sagen Sie uns, wie Sie mit dem Manual zurechtkommen. Wenn Sie zufrieden sind, dann sind wir auf dem richtigen Weg. Teilen Sie uns aber auch Ihre 'Stolpersteine' mit, damit wir sie aus dem Weg räumen können.

Manualredaktion K D ST PM222
Otto-Hahn-Ring 6, 8 München 83

Inhalt

1 Eine Übersicht über FORMANT

FORMANT heißt "Format Manager für Terminals" und besteht aus zwei Teilen:

- Teil 1 ist der Interaktive Maskengenerator FORMANTGEN. Damit erstellt man Masken.

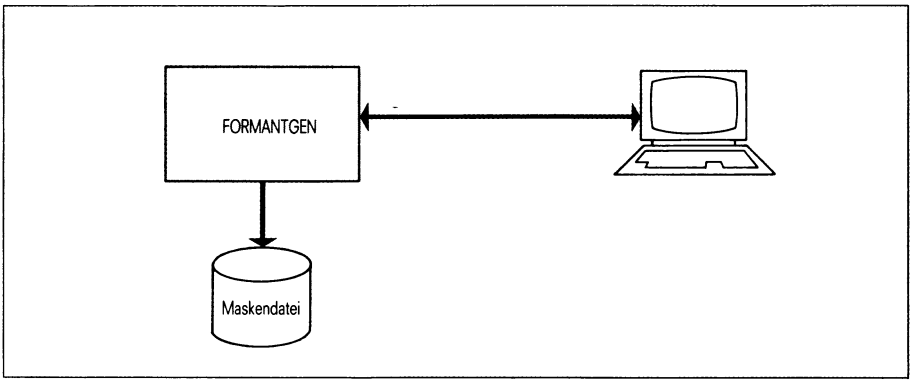


Bild 1-1: FORMANTGEN erzeugt Masken.

- Teil 2 ist die Menge der FORMANT-Funktionen. Die Funktionen ruft man in einem Anwendungsprogramm auf, um die Masken einzusetzen.

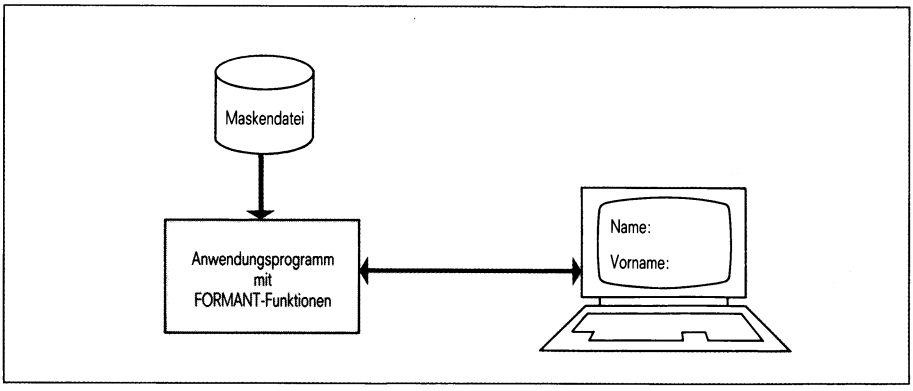


Bild 1-2: Einsetzen von Masken.

FORMANT ist eine Maskensteuerung für alle SINIX Systeme. Sie können FORMANT in C-Programmen und in COBOL-Programmen einsetzen.

Was ist eine Maske?

Eine Maske erleichtert dem Bediener eines Programms die Arbeit. Eine Maske gleicht einem Formular, das man am Bildschirm ausfüllen kann. Es gibt auf dem Bildschirm Textfelder und variable Felder.

- Textfelder entsprechen den vorgedruckten Texten eines Formulars. Diese kann man nicht verändern. Sie gehören nicht zum Datensatz.
- Variable Felder entsprechen den Kästchen eines Formulars. Man kann darin etwas eintragen oder einen vorgegebenen Inhalt verändern. Die variablen Felder gehören zum Datensatz. Sie bilden die zusammen die Nettodaten.

Masken erstellen Sie im Dialog mit FORMANTGEN. Man kann aber auch im Anwendungsprogramm ein Maske abändern oder ganz neu erstellen. Dies geschieht dann mit FORMANT-Aufrufen.

Masken erleichtern dem Programmierer und dem Bediener die Arbeit. Der Programmierer kann:

- die Ein- und Ausgabe programmieren, ohne die physischen Eigenschaften der Datenstation zu kennen,

- unterschiedliche Zeichensätze und Darstellungsarten wählen,
- Daten von FORMANT prüfen lassen (Plausibilitätsprüfungen),
- strukturierte Datensätze aufbauen lassen,
- Funktionstasten abfragen,
- Formulare drucken (Masken für Drucker).

Das Programm kann unabhängig vom "Rahmen" der Ein- und Ausgabe bleiben. Die Texte, die Positionen der Felder und die Eigenschaften der Felder sind in der Maske festgelegt und nicht im Programm. Das heißt, im Programm braucht man nur mit Nettodaten umzugehen.

Die Vorteile für den Bediener sind:

- Daten in Masken einzugeben ist sehr einfach, sicher und übersichtlich.
- Man kann die Schreibmarke beliebig innerhalb der Eingabefelder bewegen.
- FORMANT prüft laufend die Eingabe und zeigt Fehler an.
- Funktionstasten und Systemsteuertasten erleichtern die Bedienung.
- Masken ermöglichen eine klare Bedienerführung.

Der folgende Abschnitt zeigt, wie FORMANT arbeitet.

1.1 Wie arbeitet FORMANT?

Eine FORMANT-Anwendung besteht aus

- einem Anwendungsprogramm in C oder in COBOL,
- den mit FORMANTGEN definierten Masken (jede Maske steht in einer Datei),
- den Benutzerdateien, auf die das Anwendungsprogramm zugreift.

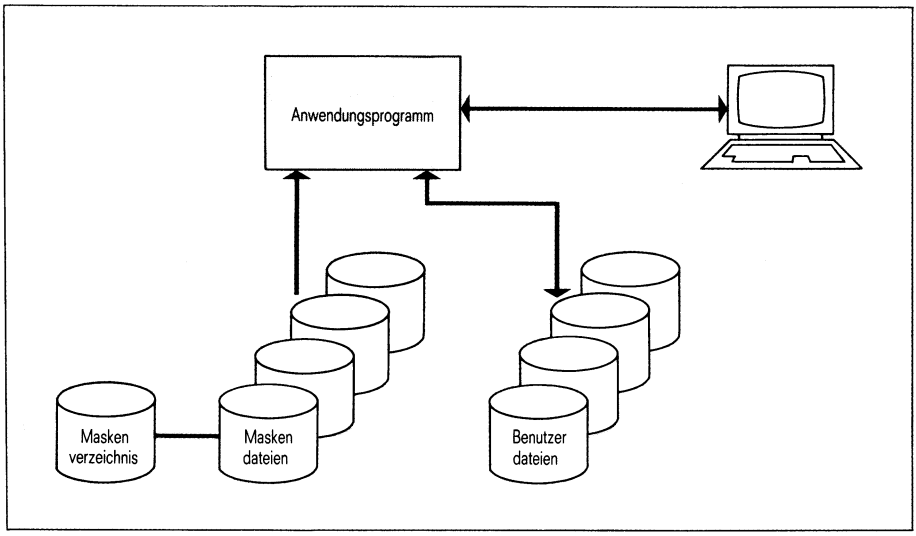


Bild 1-3: Teile einer FORMANT-Anwendung.

Das Anwendungsprogramm enthält FORMANT-Aufrufe, die die verschiedenen Funktionen von FORMANT auslösen.

Alle FORMANT-Funktionen sind Bestandteil des Anwendungsprogramms. Bei C-Programmen sind es Funktionen, die mit eingebunden werden. Bei COBOL-Programmen sind es Unterprogramme, die mit CALL aufgerufen werden.

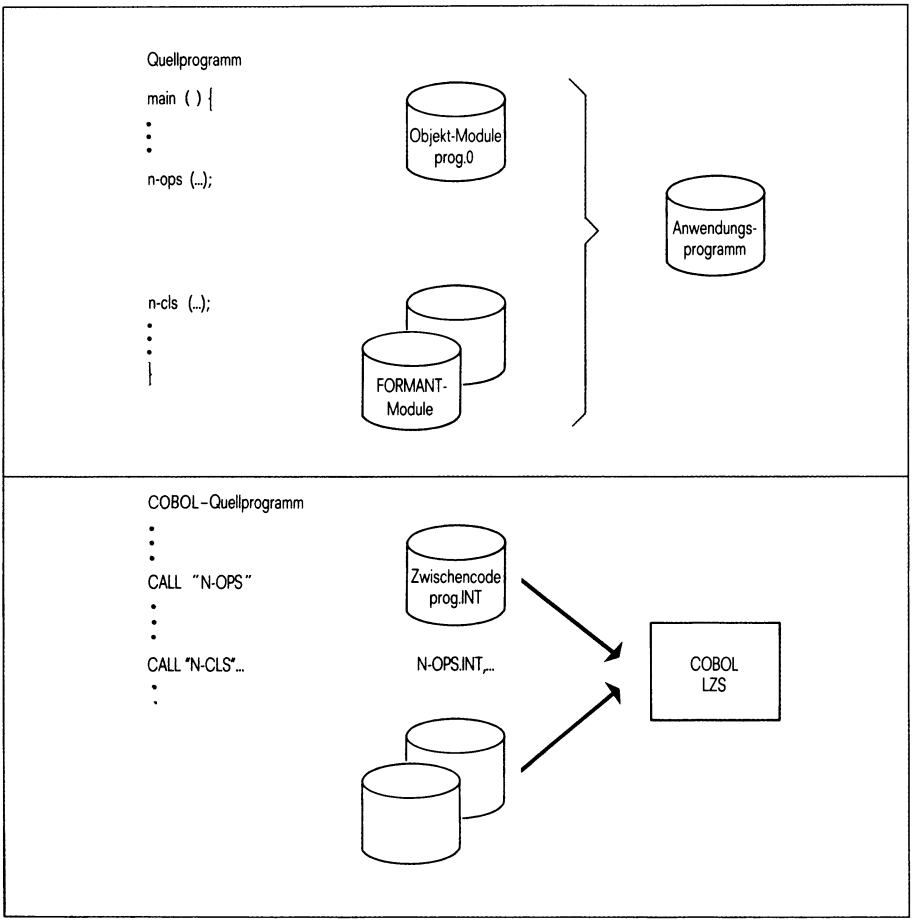


Bild 1-4: FORMANT-Funktionen im Anwendungsprogramm.

1.1.1 Die FORMANT-Sitzung

Im Anwendungsprogramm wird als erstes die FORMANT-Sitzung eröffnet. Die FORMANT-Sitzung ist ein Verarbeitungsabschnitt im Anwendungsprogramm. Alle FORMANT-Aufrufe liegen innerhalb der Sitzung. Das Programm erhält einen Sitzungs-Deskriptor. Das ist eine Nummer, die diese Sitzung identifiziert. Für die Sitzung stellt FORMANT Standard-Parameter bereit.

Sitzungs-Parameter und Umgebungs-Parameter beeinflussen den Ablauf der FORMANT-Anwendung. FORMANT legt beim Eröffnen der Sitzung Standardwerte dafür fest. Diese Standardwerte können Sie während der Sitzung abändern.

In Version 1.0 ist nur *eine* FORMANT-Sitzung möglich. In späteren Versionen können Sie mehrere Sitzungen eröffnen und auf diese Weise Masken für Bildschirm und Drucker gleichzeitig behandeln.

1.1.2 Ausgabe und Eingabe

Zuerst wird eine Maske geladen oder Speicherplatz bereitgestellt, wenn die Maske während des Programmlaufs neu aufgebaut werden soll.

Nun soll die Maske ausgegeben und Daten eingelesen werden. Wie läuft das ab? FORMANT übernimmt bei folgenden Aufrufen die Kontrolle über die Ein- und Ausgabe.

- write (schreiben)
- read (lesen) und
- write read (schreiben und lesen).

Schreiben heißt: die Maske wird mit den aktuellen Feldinhalten am Bildschirm ausgegeben. Die aktuellen Feldinhalte sind zunächst die Inhalte der Textfelder und Füllzeichen für die variablen Felder.

Lesen heißt: FORMANT nimmt die Daten entgegen, die der Bediener eingibt und prüft die Eingabe Zeichen für Zeichen. Bei Eingabefehlern reagiert FORMANT mit einer Fehlermeldung. Der ganze Ablauf der Dateneingabe ist in Kapitel 2 beschrieben. Dort finden Sie auch die Hinweise auf alle Prüfmöglichkeiten.

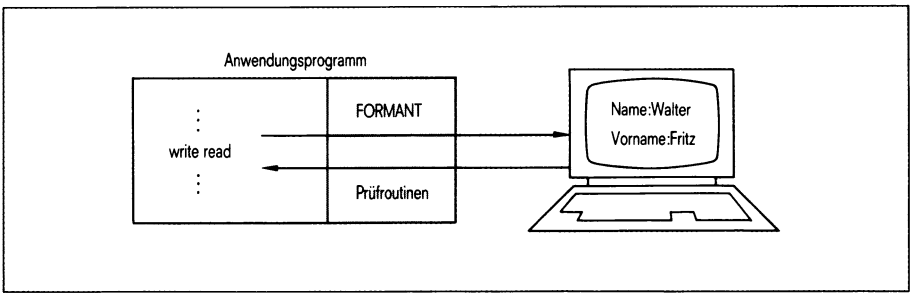


Bild 1-5: Ausgabe und Eingabe

In Version 1.0 ist die Funktion 'Schreiben' noch als leere Funktion implementiert. Das heißt, daß alle Funktionen, die den Bildschirm verändern, direkt wirken. Sie können und sollten aber schon jetzt die Schreibaufrufe einbauen, damit Sie Ihre Programme später nicht ändern müssen.

1.1.3 Verarbeitungsfunktionen

Nach der Eingabe haben die Felder der Maske neue aktuelle Inhalte. Die eingelesenen Daten sind im Programm jedoch noch nicht verfügbar! Sie können jetzt:

- die Inhalte einzelner Felder in eigene Datenbereiche im Programm übertragen oder
- die Inhalte aller variablen Felder auf einmal in einen strukturierten Datensatz im Programm übertragen,
- die aktuellen Feldinhalte verändern oder löschen,
- weitere FORMANT-Funktionen aufrufen, z.B. Maske abspeichern, ausdrucken, Schreibmarkenposition abfragen usw.
- die aktuelle Maske schließen und eine neue Maske anlegen (Maskenwechsel).

In der Regel wird folgender Verarbeitungszyklus mehrfach durchlaufen werden:

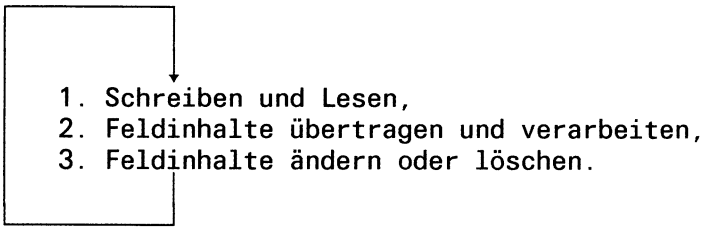


Bild 1-6: Verarbeitungszyklus

1.1.4 Benutzerroutinen

Außer den Prüfungen, die FORMANT durchführt, können Sie natürlich noch eigene Datenprüfungen vorsehen. Dazu können Sie Benutzerroutinen schreiben.

Benutzerroutinen sind Unterprogramme, die FORMANT ausführt, wenn

- der Bediener ein Datenfeld fertig eingegeben hat und dieses Feld als Triggerfeld definiert ist (siehe Abschnitt 3.4).
- der Bediener eine Funktionstaste oder eine Systemsteuertaste gedrückt hat, der Sie eine Benutzeroutine zugeordnet haben.

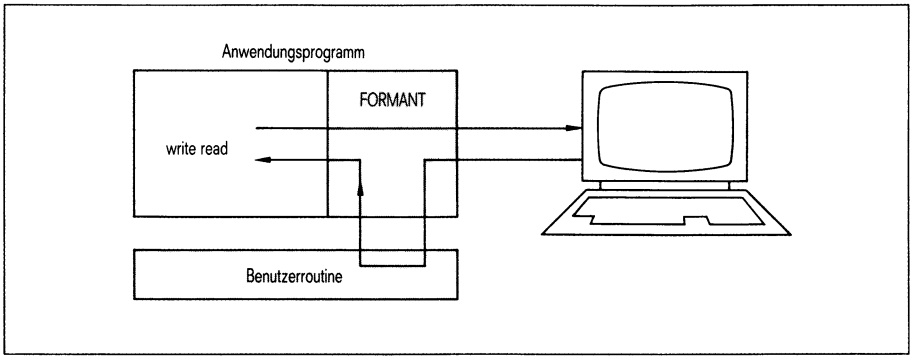


Bild 1-7: Benutzerroutinen

Sie können also auf die Eingabe bestimmter Felder (Triggerfelder) oder auf bestimmte Tasten (z.B. Funktionstasten) reagieren. Anschließend können Sie die Eingabe mit dem nächsten Feld der Maske fortsetzen oder auch die Eingabe abbrechen.

1.2 Was leistet FORMANT?

FORMANT unterstützt folgende Abläufe:

- Erstellen und Verwalten von Masken. Das ist die Aufgabe von FORMANTGEN.

  FORMANTGEN ist ein Programm, mit dem Sie im Dialog arbeiten. Sie können damit u.a. Masken erstellen, abändern, ausdrucken, ansehen usw. FORMANTGEN arbeitet mit logischen Abbildern der Masken, den Formaten. Aus dem Format generieren Sie mit FORMANTGEN die Maske.

  Sie definieren mit FORMANTGEN den Aufbau der Maske, die Eigenschaften der Felder und weitgehend den Ablauf der Eingabe.

  FORMANTGEN ist in Kapitel 3 beschrieben.

Die folgenden Abläufe realisiert man mit FORMANT-Aufrufen im Anwendungsprogramm.

- Ausgeben von Masken auf dem Bildschirm oder mit einem Drucker.

  Die Aufraufe sind:

  write
  write read print presentation image

- Eingeben von Daten über die Tastatur. Dabei kann FORMANT eine Reihe von Prüffunktionen übernehmen. Der Ablauf der Eingabe ist in Kapitel 2 beschrieben. Dort finden Sie auch die Hinweise, wie Sie den Ablauf der Eingabe beeinflußen können. Besondere Funkltionen sind:

  — Eingeben mit Gegentastprüfen.

  — Duplizieren, das heißt man kann vorhandene Inhalte in die aktuell einzugebenden Felder übernehmen.

- Eingegebene Daten in strukturierten Datensätzen ablegen. Mit FORMANTGEN legen Sie für die Maske fest, in welcher Reihenfolge die

variablen Felder im Datensatz stehen sollen. Der Aufruf 'get record' liefert den entsprechend aufgebauten Datensatz.

- Masken in Dateien abspeichern.

  Der Aufruf 'save presentation image' sichert die gerade bearbeitete Maske in einer Datei. Dazu gehört auch der aktuelle Inhalt aller variablen Felder.

- Masken während des Programmablaufs ändern. Dazu gehört:

  — das Einfügen oder Ausfügen von Feldern (insert field und delete field),

  — das Ändern von Darstellungseigenschaften der Felder, z.B. invers, kursiv, halbhell usw. (change field attributes),

  — das Ändern globaler Eigenschaften, wie satzweise oder feldweise Eingabe, Feldaufbereitung beim Überspringen von Feldern usw. (modify environment parameters).

- Masken während des Programmlaufs neu aufbauen.

  Dazu macht man eine entsprechende Angabe beim Aufruf 'create presentation image' und fügt neue Felder in eine leere Maske ein.

- Funktionstasten im Programm abfragen bzw. Benutzerroutinen dafür festlegen.

  Wie FORMANT Funktionstasten behandelt, legen Sie mit dem Aufruf 'modify input reaction table' fest. Dabei können Sie aich einzelne Tasten sperren oder neu zuordnen.

- Daten feldweise über Benutzerroutinen prüfen.

  Benutzerroutinen schließen Sie mit dem Aufruf 'define user exits' an.

1.3 Die Schnittstellen von FORMANT.

FORMANT hat folgende Schnittstellen:

- Programmschnittstellen für Anwendungsprogramme und Benutzerroutinen in COBOL und in C.

  Diese sind als Unterprogramme bzw. Funktionen realisiert. Die Beschreibung finden Sie in den Kapiteln 5 und 6.

- Die Bedienerschnittstelle.

  Viele Abläufe bei der Dateneingabe sind von FORMANT vorgegeben. Sie können sie jedoch im Anwendungsprogramm beeinflussen. Die Bedienerschnittstelle ist in Kapitel 2 beschrieben.

- Die Schnittstelle zur Maskengenerierung.

  Hierfür gibt es FORMANTGEN als eigenes Werkzeug. FORMANTGEN ist in Kapitel 3 beschrieben.

- Die Geräteschnittstellen.

  FORMANT unterstützt in Version 1.0 die Datensichtstation 97801 und die Drucker 9001 und 9004.

- Die Schnittstelle zum Dateisystem.

  FORMANT erzeugt und benutzt Dateien im SINIX-Dateisystem. Eine Übersicht finden Sie in Abschnitt 8.2.1.

1.4 Was müssen Sie tun, um mit FORMANT arbeiten zu können?

In den vorangegangenen Abschnitten haben Sie eine Übersicht bekommen, welche Möglichkeiten FORMANT bietet und was eine FORMANT-Anwendung ist. Hier sollen Sie einen Wegweiser erhalten, der Sie möglichst ohne Umwege zum Ziel führt, eine funktionierende FORMANT-Anwendung zu erstellen.

Ausgangspunkte sind Ihre Anforderungen an die Anwendung:

- Aufbau und Inhalt der Masken,
- die Bedienungsoberfläche einschließlich Funktionstasten,
- Prüffunktionen,
- Datenstrukturen usw.

Folgende Arbeitsschritte sind nun der Reihe nach nötig:

1. Machen Sie sich mit der Bedienerschnittstelle von FORMANT vertraut. Kapitel 2 beschreibt die Bedieneroberfläche. Zugleich erhalten Sie die Hinweise, wie Sie die Bedieneroberfläche mit FORMANT-Aufrufen beeinflussen können.
2. FORMANT muß an Ihrem SINIX-Rechner installiert sein. Alle Hinweise hierzu finden Sie in Kapitel 8.
3. Sie erstellen mit FORMANTGEN ihre Masken. Dabei gehen Sie so vor, wie in Kapitel 3 beschrieben. Jede fertige Maske steht in einer Datei.

   Beim Erstellen der Maske legen Sie u.a. bereits fest:

   — einen Teil der Bedieneroberfläche, z.B. die Reihenfolge in der die Felder einzugeben sind,

   — die Eigenschaften der Felder,

   — welche Prüfungen FORMANT durchführt,

   — wie der Datensatz aufgebaut ist.
4. Sie schreiben Ihr Anwendungsprogramm.

   Mit FORMANT-Aufrufen steuern Sie den Ablauf der Anwendung. Dabei legen Sie zum Beispiel fest, wann welche Masken verwendet werden oder Sie können die Bedeutung der Tasten festlegen bzw.

ändern. Hier können Sie auch die Angaben, die Sie beim Erstellen der Maske gemacht haben abfragen und abändern.

Informationen und Beispiele zum Anwendungsprogramm finden Sie in Kapitel 4. Kapitel 5 und 6 beschreiben die FORMANT-Aufrufe in COBOL und in C.

5. Sie schreiben eigene Benutzerroutinen, wenn Sie solche verwenden wollen.

   Alles über Benutzerroutinen finden Sie in Kapitel 7.

6. Anwendungsprogramm und evtl. Benutzerroutinen müssen übersetzt werden.

   Was Sie dabei beachten müssen, steht in Kapitel 8. Dort erfahren Sie auch, wie Sie in C geschriebene Anwendungsprogramme binden und was Sie in der SINIX-Umgebung beachten müssen.

7. Ihr Anwendungsprogramm ist dann fertig und Sie können es starten und bedienen, wie in Kapitel 2 beschrieben.

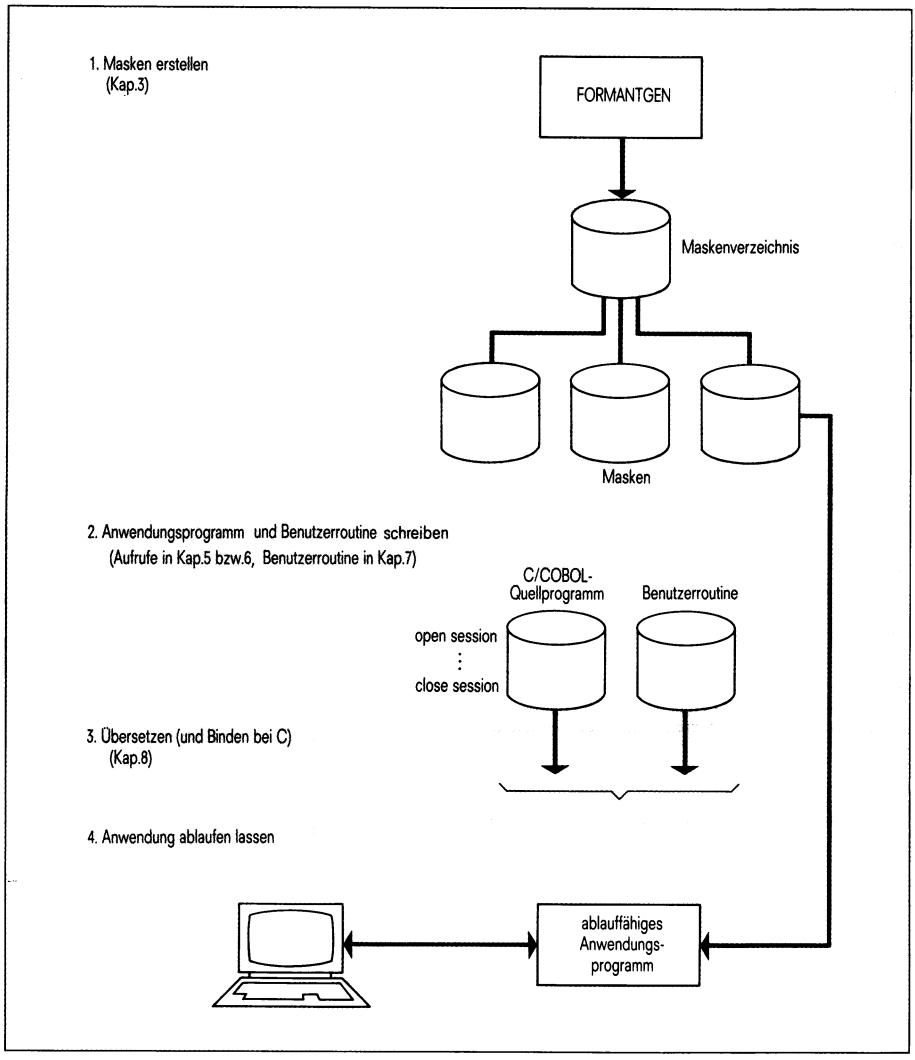


Bild 1-8: Wie eine FORMANT-Anwendung entsteht

2 Bedienen einer FORMANT-Anwendung

FORMANT ist sehr vielseitig. Deshalb kann die Bedieneroberfläche einer FORMANT-Anwendung ganz unterschiedlich gestaltet sein. Das hängt eben davon ab, wie das Anwendungsprogramm die FORMANT-Funktionen nutzt. Sie finden in den folgenden Abschnitten die Möglichkeiten beschrieben, wie die Bedieneroberfläche gestaltet sein kann. Dazu erhalten Sie Hinweise, mit welchen FORMANT-Funktionen Sie die Bedieneroberfläche beeinflussen.

Dies ist also keine Beschreibung für den Benutzer. Sie gestalten dieBedieneroberfläche durch das Anwendungprogramm und müssen dann den Benutzer entsprechend informieren.

2.1 Anwendungsprogramm aufrufen und beenden

Eine FORMANT-Anwendung ist ein normales Programm in C oder in COBOL und wird ebenso aufgerufen. Da FORMANT allerdings bei der Ein- und Ausgabe im 'cbreak-Modus' arbeitet, ist eine Vorbereitung nötig. Nach dem Programmaufruf muß man wieder 'aufräumen'.

2.1.1 Vorbereitungen zum Programmaufruf

FORMANT arbeitet mit 25 Zeilen. Um Fehler zu vermeiden, müssen Sie den Bildschirm in den 25-Zeilen-Modus schalten. Das geht mit der Steuerzeichenfolge:

ESC [0 u

2.1.2 Anwendungsprogramm aufrufen

Eine FORMANT-Anwendung rufen Sie auf wie jedes andere Programm:

programmname [parameter]	bei C-Programmen
cbrun programmname [parameter]	bei COBOL-Programmen

Ob Parameter anzugeben sind, hängt von Ihrem Programm ab. FORMANT braucht selbst keine Aufrufparameter. Sprachabhängige Einzelheiten zum Programmaufruf finden Sie in Abschnitt 8.2.2 für COBOL-Programme und in Abschnitt 8.2.3 für C-Programme.

2.1.3 Anwendungsprogramm beenden

Im Anwendungsprogramm muß eine Ende-Bedingung definiert sein, z.B. Drücken der Taste [END]. Sie können aber auch eine andere Ende-Bedingung gewählt haben. Siehe dazu das Beispiel in Abschnitt 4.3.

Nach dem Programmende sind einige Aufräumarbeiten nötig.

Meist stehen Schreibmarke und "$" auf der Zeile 25. Das hat folgenden Effekt: Sie können die Schreibmarke nicht mehr aus dieser Zeile herausbewegen und jede Ausgabe geht nur in diese Zeile. Diese unangenehme Erscheinung beenden Sie mit der Steuerzeichenfolge:

ESC [2 J ESC [1 u

Damit haben Sie gleichzeitig den Bildschirm gelöscht und wieder auf den 24-Zeilen-Modus gesetzt.

Nach dem Beenden einer FORMANT-Anwendung kann der Zeichenspeicher, in dem normalerweise der Zeichensatz 'Klammern' liegt, einen anderen Inhalt haben. Das beheben Sie mit der Eingabe [ESC] [)] [w].

Bei Programmfehlern kann es vorkommen, daß Sie das Programm nicht normal beenden können. Als "Notausgang" können Sie das Anwendungsprogramm auch mit der Taste [DEL] abbrechen. Dann wird allerdings die FORMANT-Sitzung nicht ordnungsgemäß geschlossen, wenn die Aufrufe 'destroy presentation image' und 'close session' nicht durchlaufen werden.

In diesem Fall ist der Bildschirm in einem Zustand, in dem "nichts mehr geht". Das beheben Sie mit dem Kommando:

stty echo -nl -cbreak -tandem

2.1.4 Beispiel für eine Aufrufprozedur

Alle nötigen Kommandos können Sie in einer Shell-Prozedur zusammenfassen. Diese Prozedur könnte aussehen wie folgt:

```
display ESC[0u
programmaufruf
display ESC[2JESC[1u
```

Beachten Sie bitte, daß Sie das Zeichen Escape (ESC) mit dem ced nicht direkt eingeben können. Das geht aber z.B. mit cat oder ed.

2.2 Der Bildschirm

Der Bildschirm hat 80 Spalten und 25 Zeilen. 24 Zeilen sind für die Maske verfügbar, die 25. Zeile benutzt FORMANT als Systemzeile für Fehlermeldungen und Statusmeldungen (siehe auch Abschnitte 2.4 und 2.5).

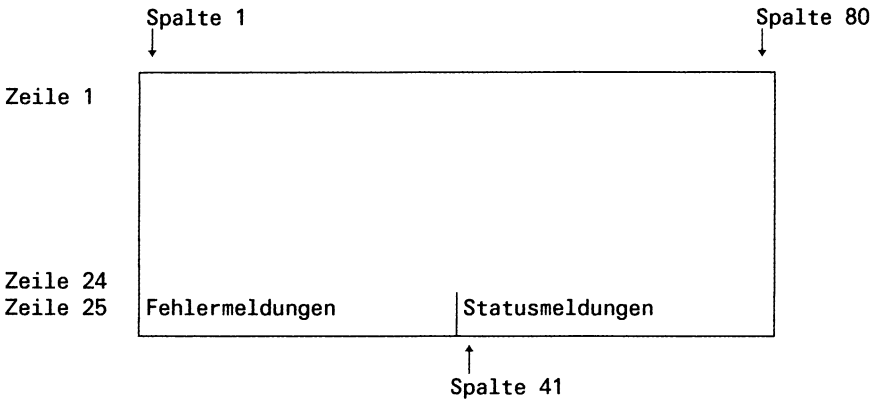


Bild 2-1: Wie der Bildschirm aufgebaut ist

2.3 Daten eingeben - die Tastatur

FORMANT hat eine eigene Tastaturbelegung für die Systemsteuertasten und die Funktionstasten. Sie finden diese Belegung in einem Faltblatt im Anhang, das Sie herausklappen können.

Diese Tastaturbelegung ist Standard. Im folgenden sind alle Tastaturfunktionen so beschrieben, wie sie standardmäßig definiert sind. Sie können aber für viele Tasten die Funktion umdefinieren oder sperren.

Eine Reihe von Tasten, die FORMANT standardmäßig nicht benutzt, können Sie mit eigenen Funktionen belegen (Funktionstasten).

Die Tastenfunktionen ändern oder sperren Sie mit dem Aufruf:

modify input reaction table

Eine Übersicht, welche Tasten Sie ändern oder sperren können, finden Sie in Abschnitt 2.3.8.

Hat FORMANT eine Maske ausgegeben, füllen Sie Feld für Feld aus, wie in den folgenden Abschnitten beschrieben.

Wann erhält das Anwendungsprogramm die Steuerung zurück?

FORMANT kennt zwei Betriebsarten:

- satzweise Eingabereaktion (Standard). Hierbei erhält das Anwendungsprogramm die Steuerung zurück, wenn
  - alle Eingabefelder ausgefüllt sind, das heißt das erste Feld in der Reihenfolge der Bearbeitung wieder erreicht wurde und zwar durch Vorwärtspositionieren.
  - Sie die Taste [F14] (SEND: Satzende) gedrückt haben oder
  - Sie eine Funktionstaste gedrückt haben, deren Wirkung entsprechend programmiert wurde oder
  - eine Benutzerroutine durchlaufen wurde.
- feldweise Eingabereaktion. Das Anwendungsprogramm erhält die Steuerung jedesmal, wenn Sie ein Eingabefeld ausgefüllt haben, das heißt, wenn Sie es vorwärts verlassen.

Der Unterschied liegt also nur darin, wieviele Daten das Anwendungsprogramm mit einem Leseaufruf erhält. Sie können die Betriebsart im Programm ändern mit dem Aufruf:

modify environment parameters, Parameter FORMANTCONTROL

Beachten Sie bitte: Diese Betriebsarten haben nichts damit zu tun, wie FORMANT bei der Eingabe in ein Feld reagiert. Hier reagiert FORMANT immer sofort.

Fehler rücksetzen

Erkennt FORMANT eine fehlerhafte Eingabe, erscheint in Zeile 25 eine Fehlermeldung und die Tastatur ist gesperrt (siehe auch Abschnitt 2.4).

Drücken Sie [WORD] (ERS: Fehler rücksetzen) oder
[q] oder [Q]

Dann können Sie weiter eingeben.

Aktuelle Maske ausdrucken

Sie können das Bild der aktuellen Maske jederzeit auf dem Drucker ausdrucken lassen (Hardcopy-Funktion). Drücken Sie die Taste [PRINT].

FORMANT druckt den aktuellen Bildschirminhalt auf dem Drucker aus. FORMANT erzeugt eine temporäre Datei, druckt sie mit 'lpr' aus und löscht sie.

Dateiname: _ppi_#.tmp (# von 0 bis 9).

2.3.1 Feld ausfüllen

Nach dem Programmaufruf steht die Schreibmarke im ersten Eingabefeld. Dieses Feld wird mit Feldleerzeichen angezeigt, falls es nicht vorbelegt ist. Alle anderen nicht vorbelegten Eingabefelder füllt FORMANT mit Bildschirmfüllzeichen (standardmäßig das Leerzeichen). Sie können nun in das erste Feld Daten eingeben. FORMANT prüft bei der Eingabe jedes Zeichen und reagiert entsprechend den Feldeigenschaften, die Sie für jedes Feld definiert haben. Zum Beispiel können Sie für ein Feld Großschreibung festlegen. Dann zeigt FORMANT jeden eingegebenen Buchstaben sofort groß an.

Das Feldleerzeichen ist standardmäßig das Zeichen '_' (Unterstrich). Sie können es abändern mit dem Aufruf:

modify environment parameters, Parameter FELDLEERZ

Die Feldeigenschaften (Attribute) bestimmen:

- die Felddarstellung am Bildschirm, z.B. (halbhell, blinkend usw.),
- den Zeichensatz (z.B. National, Klammern usw.),
- die Bearbeitungseigenschaften
  (z.B. Vollständigkeitsfeld, Großschreibung),
- Feldausrichtung (links oder rechts),
- zulässige Eingabezeichen (z.B. nur numerisch),
- Eingabeeigenschaften
  (z.B. Gegentastprüfen, Korrekturmöglichkeiten),
- Füllzeichen (Feldfüllzeichen und Datenfüllzeichen).

Die Feldeigenschaften legen Sie fest:

beim Erstellen der Maske mit FORMANTGEN (Funktion Attribute setzen) oder

beim Einfügen eines Feldes mit dem Aufruf 'insert field'.

Folgende weitere Aufrufe beziehen sich auf Feldeigenschaften:

get field attribute	Feldeigenschaften abfragen
change field attribute	Feldeigenschaften ändern
change field type	Feldtyp ändern (Textfeld bzw. var. Feld)

Die vollständige Beschreibung der Feldeigenschaften finden Sie in Abschnitt 3.4.

Eingabe in numerische Felder

In numerische Felder können Sie nur Ziffern eingeben oder, falls Nachkommastellen definiert sind, auch das Zeichen Punkt. Das Zeichen Komma können Sie ebenfalls eingeben, es wird aber als Punkt abgebildet.

Felder mit Nachkommastellen:

FORMANT prüft bei der Eingabe, ob die definierte Anzahl Nachkommastellen noch in das Feld paßt und fordert evtl. auf, ein Komma einzugeben: 'E5 Zuwenig Nachkommastellen'.

Wenn Sie versuchen, mehr Nachkommastellen einzugeben, meldet FORMANT 'E7 Zuviele Nachkommastellen'.

Beim Aufbereiten eines Feldes mit Nachkommastellen setzt FORMANT wenn nötig den Dezimalpunkt und füllt fehlende Nachkommastellen mit Nullen auf. Anschließend richtet FORMANT das Feld wie gewünscht aus.

Beispiel: Sie bearbeiten ein Feld, für das 2 Nachkommastellen definiert sind und das 6 Zeichen lang ist (Ausrichtung links).

Eingabe	1.3___	↵
aufbereitetes Feld	1.30$$	

Vorzeichenfelder:

Vorzeichenfelder sind genauso einzugeben wie andere numerische Felder. Sie schließen aber die Eingabe ab mit einer Vorzeichentaste: ↵ = FE+, CE = FE-.

Beispiel: Sie bearbeiten ein Vorzeichenfeld, für das 3 Nachkommastellen definiert sind und das 9 Zeichen lang ist (Ausrichtung rechts).

Eingabe	2.15_____	CE
aufbereitetes Feld	$$$2.150	

Das Vorzeichen steht immer an letzter Stelle. Nur das Minuszeichen wird eingesetzt. Bei positivem Vorzeichen steht an letzter Stelle ein Leerzeichen.

In beiden Beispielen ist das Feldfüllzeichen das Zeichen '$'.

Eingabe mit Ausweisleser

Wenn ein Ausweisleser in die Tastatur integriert ist, kann FORMANT Ausweisdaten in ein Feld der Maske eintragen. Das Feld muß die Feldeigenschaft 'Ausweislesefeld' haben.

Steht ein Ausweislesefeld zur Eingabe an, prüft FORMANT, ob ein Ausweisleser an der Datenstation vorhanden ist. Wenn ja, fordert FORMANT auf, eine Ausweiskarte einzuziehen. Einziehen des Ausweises bewirkt:

1. FORMANT liest die Ausweisdaten und prüft sie entsprechend den definierten Feldeigenschaften.
2. FORMANT trägt die Ausweisdaten in das Ausweislesefeld ein.
3. FORMANT fordert das nächste Feld zur Eingabe an. Das Ausweislesefeld ist also nicht mit einer Systemsteuertaste zu verlassen. FORMANT trägt den logischen Code der Taste ↵ (FE+) ein.

Ist kein Ausweisleser vorhanden, behandelt FORMANT ein Ausweislesefeld wie ein normales Eingabefeld.

Fehler beim Lesen einer Ausweiskarte:

Bei einem Bedienungsfehler gibt FORMANT eine Fehlermeldung aus. Um die Fehlermeldung zu löschen, muß man die Ausweiskarte aus dem Leser nehmen. Anschließend kann man weiter eingeben. Möglich sind die Fehlermeldungen:

E4 Zeichen unzulaessig !
E14 Lesegeschwindigkeit falsch !

E15 Ausweisleser oder Karte defekt !
E16 Timeout !

Wenn der Ausweis nicht gelesen werden kann oder wenn versucht wird, mit der Tastatur in ein Ausweislesefeld einzugeben, unterbricht FORMANT die Eingabe. Der Leseaufruf liefert den logischen Code 'CARDBAD' (siehe write read).

Was passiert, wenn die Ausweiskarte aus dem Leser genommen wird?

Wenn Sie den Ausweis aus dem Leser nehmen, unterbricht FORMANT die Eingabe und der Leseaufruf liefert den logischen Code 'CARDOUT' (siehe write read). Das gilt auch, wenn der Ausweis entnommen wurde, um eine Fehlermeldung zurückzusetzen. Damit wird das Anwendungsprogramm von jedem Bedienungsfehler informiert.

2.3.2 Feld verlassen

Sie verlassen das aktuelle Eingabefeld, wenn Sie

- das letzte Zeichen des Feldes eingegeben haben. FORMANT zeigt das nächste Eingabefeld mit Feldleerzeichen an, wenn es ein leeres Feld ist, sonst die Vorbelegung. Die Schreibmarke steht auf dem Beginn dieses Feldes.
- eine Systemsteuertaste gedrückt haben, mit der das Feld verlassen wird (siehe weiter unten, Schreibmarke positionieren). Die Wirkung ist je nach Systemsteuertaste verschieden.
- eine Funktionstaste gedrückt haben, deren Wirkung Sie entsprechend programmiert haben.

2.3.3 Schreibmarke positionieren

- innerhalb eines Datenfeldes:

  [→] ein Zeichen weiter.

  Die Taste wirkt nur, wenn schon Daten im Feld vorhanden sind. Das heißt, man kann keine Leerstellen überspringen.

  [←] ein Zeichen zurück.

Standardmäßig bleiben alle geschriebenen Zeichen stehen. Sie können aber auch festlegen, daß die Zeichen beim Rückpositionieren gelöscht werden. Aufruf:

modify environment parameters, Parameter LOESCHLINKS

⌫ ein Zeichen löschen.

Die Schreibmarke wird um ein Zeichen nach links gesetzt. Gelöscht wird nur, wenn die Schreibmarke am aktuellen Feldende steht. Das heißt, man kann mit dieser Taste nichts ausfügen.

F15 LOEF: Feld löschen.

Das Feld wird mit Feldleerzeichen gelöscht. Die Schreibmarke steht am Anfang des Feldes.

- auf andere Datenfelder:

Sie können ein Datenfeld verlassen, indem Sie vorwärts positionieren oder indem Sie rückwärts positionieren. Vorwärts und rückwärts beziehen sich auf die Reihenfolge der Bearbeitung. Diese legen Sie fest, wenn Sie die Maske erstellen. Sie muß nicht mit der Reihenfolge der Felder am Bildschirm übereinstimmen.

Das nächste Feld, das bearbeitet werden soll, wird jeweils mit Feldleerzeichen angezeigt. Die Schreibmarke steht am Beginn des Feldes.

Feld vorwärts verlassen:

Wenn Sie ein Feld vorwärts verlassen, bereitet es FORMANT immer auf. Wenn Sie beim Vorwärtspositionieren Felder überspringen, bereitet FORMANT diese ebenfalls auf. Aufbereiten heißt, die Felder werden ausgerichtet und mit Feldfüllzeichen gefüllt. Mußfelder kann man nicht überspringen.

Diese Reaktion ist Standard. Sie können sie ändern, so daß FORMANT die übersprungenen Felder nicht mit Feldfüllzeichen füllt. Aufruf:

modify environment parameters, Parameter VOR_AUFB

↵ FE+: positives Feldende

beendet die Eingabe des Datenfeldes. Der Rest des Feldes ab Schreibmarkenposition wird gelöscht und mit Feldfüllzeichen aufgefüllt. Die Schreibmarke steht am Beginn des nächsten Datenfeldes, das zu bearbeiten ist.

Vorzeichenfelder erhalten ein positives Vorzeichen.

[CE] FE-: negatives Feldende

beendet die Eingabe des Datenfeldes. Der Rest des Feldes ab Schreibmarkenposition wird gelöscht und mit Feldfüllzeichen aufgefüllt. Die Schreibmarke steht am Beginn des nächsten Datenfeldes, das zu bearbeiten ist.

Diese Taste ist nur bei Vorzeichenfeldern zulässig. Sie erhalten ein negatives Vorzeichen.

[⇥] FORWARDS: Schreibmarke auf den Beginn des nächsten Feldes, das zu bearbeiten ist.

[↓] NEXT: Schreibmarke auf den Beginn des ersten Feldes der nächsten Zeile. Dabei werden die restlichen Felder der Zeile übersprungen.

[F14] SEND: beendet die Eingabe des Datensatzes.

Damit überspringen Sie alle Felder, die noch nicht ausgefüllt sind, außer Mußfelder. Das Anwendungsprogramm erhält die Steuerung zurück.

Wenn Sie das letze Feld vorwärts verlassen, ist der Datensatz fertig eingegeben und das Anwendungsprogramm erhält die Steuerung zurück.

Feld rückwärts verlassen:

Wenn Sie ein Feld rückwärts verlassen, dann bereitet es FORMANT auf, d.h. es wird mit Feldfüllzeichen gefüllt. Diese Reaktion ist Standard. Sie können aber auch festsetzen, daß der Inhalt des Feldes mit Bildschirmfüllzeichen (Leerzeichen) gelöscht wird, wenn Sie es rückwärts verlassen. Aufruf:

modify environment parameters, Parameter RUECK_AUFB

Wenn Sie beim Rückwärtspositionieren Felder überspringen, bleibt der Inhalt der übersprungenen Felder unverändert.

[↩] BACKWARDS: Schreibmarke auf den Beginn des vorhergehenden Feldes.

[↖] HOME: Schreibmarke auf erstes Datenfeld der Maske.

[↑] LAST: Schreibmarke auf den Beginn des ersten Datenfeldes der vorhergehenden Zeile.

[F16] LVD: Löschen variabler Datenfelder.

Alle Datenfelder werden mit Bildschirmfüllzeichen (Leerzeichen) gelöscht und gelten als leer. Die Schreibmarke steht auf dem Beginn des ersten Datenfeldes.

Rückwärtspositionieren führt nie dazu, daß FORMANT die Eingabe beendet.

2.3.4 Funktionstasten

FORMANT unterstützt folgende Funktionstasten:

[F1] bis [F12] (F1 bis F12) und [F18] (F27),

[SHIFT] [F1] bis [SHIFT] [F14] (F13 bis F26),

[HELP], [START], [END], [MENU]

Wenn Sie eine dieser Tasten drücken, erhält das Anwendungsprogramm die Steuerung zurück. Alle noch nicht ausgefüllten Felder bleiben unverändert. Sie können im Programm abfragen, welche Taste gedrückt wurde. Die Information liefert der Leseaufruf in einem eigenen Feld.

Beim nächsten Leseaufruf setzt FORMANT die Eingabe mit dem nächsten Eingabefeld fort.

Diese Reaktion ist Standard. Sie können jedoch auch festlegen, daß beim Drücken einer Funktionstaste eine eigene Benutzerroutine die Steuerung erhält (siehe Kapitel 7).

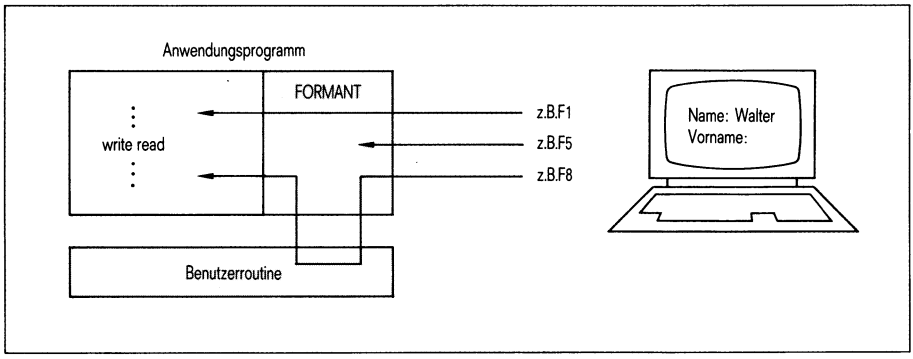


Bild 2-2: Wirkung der Funktionstasten

Beachten Sie bitte:

- Sie sollten alle Funktionstasten sperren, die nicht benutzt werden. Sie vermeiden damit unvorhergesehene Wirkungen.
- Die Bezeichnung der Tasten auf der Tastatur stimmt nicht immer mit dem Namen bei FORMANT überein.

2.3.5 Syntaxprüfung ausschalten

FORMANT prüft bei der Eingabe auf zulässige Eingabezeichen, z.B. Buchstaben, Sonderzeichen, numerisch mit oder ohne Vorzeichen und Nachkommastellen. Diese Prüfung können Sie für das aktuelle Feld ausschalten.

Drücken Sie [SHIFT] [F18] (VALID: Syntaxprüfung aus).

Sie können die Funktion dieser Taste im Programm sperren. Aufruf:

modify input reaction table, ignoriere Funktionstaste (IGN)

2.3.6 Gegentastprüfen

In der Maske können Sie mit FORMANTGEN Felder als Prüffelder definieren. Was Sie in ein Prüffeld eingeben, prüft FORMANT gegen eine Zeichenfolge, die Sie im Programm vorgeben. Das kann z.B. eine vorangegangene Eingabe sein. Die Vorgabe muß nicht unbedingt auf dem Bildschirm sichtbar sein, z.B. wenn Sie jeden Datensatz zweimal erfassen. Beispiele mit den Aufruffolgen im Programm finden Sie in Abschnitt 4.3.

FORMANT fordert im Prüfmodus nur die Prüffelder zur Eingabe an. Alle anderen Felder füllt FORMANT automatisch mit dem Inhalt des Vorgabesatzes. Stimmt die Eingabe mit der Vorgabe überein, können Sie das Feld wie üblich verlassen. Der Inhalt wird dem Programm übergeben.

Stimmt die Eingabe nicht mit der Vorgabe überein, können Sie korrigieren, vorausgesetzt das Prüffeld hat die Eigenschaft 'korrigierbar'.

[SHIFT] [F16] KOR: einzelne Zeichen korrigieren.
FORMANT zeigt das aktuelle Zeichen der Vorgabe und schaltet für ein Zeichen auf normale Eingabe. Das neue Zeichen wird nach der Eingabe gelöscht und ist nochmals einzugeben.

[SHIFT] [F15] FKOR: das Feld korrigieren.
FORMANT zeigt den ganzen Feldinhalt der Vorgabe. Sie können diesen nun überschreiben. Anschließend löscht FORMANT den Inhalt und verlangt die Eingabe nochmals.

[SHIFT] [F17] ANZ: Feld anzeigen.
Das nächste Zeichen der Vorgabe wird im Prüffeld angezeigt. Diese Anzeige ist nur eine Information, die den sonstigen Ablauf des Gegentastens nicht beeinflußt. Sie funktioniert auch bei nicht korrigierbaren Prüffeldern. Ein Systemsteuerzeichen wird als Schmierzeichen dargestellt.

Ist das Feld korrigiert, können Sie es normal verlassen.
Nicht korrigierbare Prüffelder können Sie nur verlassen,

- wenn Sie vollständig richtig gegengetastet sind oder
- mit einer Funktionstaste.

2.3.7 Duplizieren

Beim Duplizieren kopiert FORMANT einzelne Felder des vorher erfaßten Datensatzes in den aktuell zu erfassenden. Dazu müssen Sie im Anwendungsprogramm einen Dupliziersatz mit dem entsprechenden Inhalt versorgen (siehe Aufruf 'set dup record'. Nun gibt es zwei Möglichkeiten:

1. Automatisches Duplizieren

Felder, die dupliziert werden sollen, müssen die Eigenschaft Duplizierfeld haben (FORMANTGEN). Für diese Felder wird bei der Ausgabe automatisch der Inhalt des Dupliziersatzes angezeigt. Ist kein Dupliziersatz vorhanden, wird nichts dupliziert (Duplizierfunktion aus).

Standardmäßig wird nur bei solchen Feldern automatisch dupliziert, die leer sind. Vorbelegte Felder werden automatisch dupliziert, wenn Sie das im Programm festlegen. Aufruf:

modify environment parameters, Parameter AUTODUPVORF

2. Duplizieren mit der Dupliziertaste

Drücken Sie die Taste

F13 DUP: Duplizieren.

Für das Feld, in dem die Schreibmarke steht, übernimmt FORMANT die Daten aus dem Dupliziersatz. Steht die Schreibmarke nicht am Feldanfang, dupliziert FORMANT erst ab der Schreibmarkenposition. Ist kein Dupliziersatz vorhanden, meldet FORMANT einen Fehler (E12 Kein Dupliziersatz vorhanden).

2.3.8 Kurzübersicht über alle Systemsteuertasten

Die folgende Tabelle zeigt die Bedeutung aller Systemsteuertasten und ihre Standard-Zuordnung zur Tastatur.

In der letzten Spalte bedeutet:

ä Die Taste ist änderbar, d.h man kann der Tastenfunktion eine eigene Bedeutung geben oder sie sperren.

s Die Taste ist nur sperrbar. Man kann ihr aber keine andere Bedeutung geben.

Name der Taste	Bedeutung	Zuordnung zur Tastatur	änderbar sperrbar
FE+	Positives Feldende.	↵ ENTER	ä
FE-	Negatives Feldende.	CE	ä
HOME	Sprung auf erstes Feld.	↖	ä
NEXT	Sprung auf nächste Zeile.	↓	ä
LAST	Sprung auf vorige Zeile.	↑	ä
BACKWARDS	Sprung auf voriges Feld.	⇤	ä
FORWARDS	Sprung auf nächstes Feld.	⇥	ä
LVD	Lösche Datenfelder.	F16	ä
SEND	Satzende.	F14	ä
F1 - F12	Funktionstasten.	F1 bis F12	ä
F13- F26	Funktionstasten.	SHIFT F1 bis SHIFT F14	ä
F27	Funktionstaste.	F18	ä
MENU	Funktionstaste.	MENU	ä
HELP	Funktionstaste.	HELP	ä
START	Funktionstaste.	START	ä
END	Funktionstaste.	END	ä
PRINT	aktuelle Maske drucken.	PRINT	s
BACKSPACE	Ein Zeichen löschen.	⌫	s
FKOR	Feld korrigieren.	SHIFT F15	s
KOR	Zeichen korrigieren.	SHIFT F16	s
ANZ	Feld anzeigen	SHIFT F17	s
VALID	Syntaxprüfung ausschalten.	SHIFT F18	s
DUP	Duplizieren.	F14	s

2.4 Fehlermeldungen von FORMANT

Fehlermeldungen gibt FORMANT in der Systemzeile aus. Das ist die Zeile 25. Dabei sperrt FORMANT die Tastatur. Damit Sie weiter eingeben können, drücken Sie:

WORD (ERS: Fehler rücksetzen) oder
q oder Q

Dann können Sie weiter eingeben. FORMANT löscht die Fehlermeldung am Bildschirm.

Bei Syntaxfehlern in der Eingabe, die FORMANT erkennt, gibt FORMANT eigene Fehlermeldungen aus. Die Bedeutung dieser Fehlermeldungen finden Sie im folgenden Abschnitt.

In welcher Sprache FORMANT Fehlermeldungen ausgibt, legen Sie im Anwendungsprogramm fest. Standard ist Englisch. Sie können dies abändern mit dem Aufruf:

modify session parameters, Parameter FEHLERTEXTE

Ein Beispiel finden Sie bei der Beschreibung des Aufrufs.

Ausgeliefert werden Meldungstexte in englischer und deutscher Sprache. Meldungstexte in anderen Sprachen können Sie selbst erstellen und in die entsprechenden Dateien bringen (siehe Aufruf 'modify session parameters').

2.4.1 Bedeutung der Fehlermeldungen

Im folgenden finden Sie die Fehlermeldungen, die FORMANT ausgibt, sowie die möglichen Ursachen. Weitere Meldungen können im Anwendungsprogramm selbst definiert und ausgegeben werden. Diese sollten Sie dem Benutzer des Programms auch erklären (siehe nächster Abschnitt).

E0 Feld ohne Vorzeichen !

Sie haben die Taste CE (FE-) gedrückt. Das Feld ist jedoch nicht als Feld mit Vorzeichen definiert. Schließen Sie das Feld z.B ab mit CE (FE+) oder → (FORWARDS).

E1 Systemsteuerzeichen unzulaessig !

Sie haben eine Taste gedrückt, die im Anwendungsprogramm gesperrt wurde oder die nicht belegt ist (siehe Tastaturbelegung im Anhang).

E2 Feldende erreicht !

Mit der Taste [→] kann man nicht über das Feldende positionieren. Bei der Dateneingabe in leere Felder gilt bereits das Ende der eingegebenen Daten als Feldende.

E3 Feldanfang !

Mit der Taste [←] oder der Taste [⌫] wurde der Feldanfang erreicht.

E4 Zeichen unzulaessig !

Sie haben versucht ein Zeichen einzugeben, das für das Feld nicht erlaubt ist, z.B einen Buchstaben in ein numerisch definiertes Feld (auch bei Ausweislesefeldern).

E5 Zuwenig Nachkommastellen !

Für das Feld sind mehr Nachkommastellen definiert, als Sie noch eingeben können (numerisches Feld mit Nachkommastellen).

E6 Komma schon gesetzt !

In einem numerischen Feld mit Nachkommastellen ist nur ein Komma erlaubt.

E7 Zuviele Nachkommastellen !

Für das Feld sind weniger Nachkommastellen definiert, als Sie eingeben wollen (numerisches Feld mit Nachkommastellen). Sie müssen an dieser Stelle bereits den Punkt setzen.

E8 Nur Komma zulaessig !

Sie bearbeiten ein numerisches Feld mit Nachkommastellen. An dieser Stelle dürfen Sie nur Komma oder Punkt eingeben. Wenn Sie den Punkt falsch gesetzt haben, müssen Sie das Feld ganz löschen und neu eingeben ([F15]).

E9 Mussfeld !

Sie können dieses Feld nicht verlassen, ohne etwas eingegeben zu haben.

E10 Vollstaendigkeitsfeld !

Das Feld muß vollständig bis zum Feldende ausgefüllt sein, falls Sie wenigstens ein Zeichen eingegeben haben.

E11 Nur Vorzeichen erlaubt !

Sie bearbeiten ein numerisches Feld mit Vorzeichen. Das letzte Feld ist für das Vorzeichen reserviert. [CE] (FE-) setzt Minus, alle anderen Systemsteuerzeichen setzen Plus.

E12 Kein Dupliziersatz vorhanden !

Sie haben [F14] (DUP) gedrückt. Duplizieren ist nur möglich, wenn im Anwendungsprogramm der Dupliziersatz versorgt wurde (Aufruf 'set dup record').

E13 Prueffehler !

Die Eingabe im Prüfmodus stimmt nicht mit der Vorgabe überein, oder Sie haben im Prüfmodus fertig eingegeben und müssen das Feld mit einer Systemsteuertaste verlassen (Gegentastprüfen).

E14 Lesegeschwindigkeit falsch !

Sie haben die Ausweiskarte zu schnell oder zu langsam in den Ausweisleser eingezogen.

E15 Ausweisleser oder Karte defekt !

Diese Meldung erscheint auch, wenn Sie die Ausweiskarte falsch in den Leser eingezogen haben.

E16 Timeout !

Sie haben die Ausweiskarte nicht vollstindig in den Leser eingezogen.

2.4.2 Eigene Fehlermeldungen

Sie können im Anwendungsprogramm eigene Fehlermeldungen erzeugen. Diese gibt FORMANT ebenfalls in der Systemzeile aus und sperrt ebenfalls die Tastatur. Der Aufruf hierfür ist:

change error message

Ein Beispiel finden Sie in Kapitel 4.

2.5 Statusmeldungen

Statusmeldungen geben Sie im Anwendungsprogramm aus. Sie erscheinen in der Zeile 25 ab Spalte 41 und dürfen maximal 39 Zeichen lang sein. FORMANT sperrt die Tastatur nicht. Der Aufruf ist:

change status message

Ein Beispiel finden Sie in Kapitel 4.

Mit Statusmeldungen informieren Sie z.B. den Benutzer über den aktuellen Verarbeitungsstand. Statusmeldungen können nicht von Fehlermeldungen überschrieben werden.

FORMANT gibt selbst nur Statusmeldungen aus, wenn ein interner Fehler erkannt wird: 'Interner Fehler in FORMANT (#)!'. Benachrichtigen Sie den Kundendienst.

3 Masken erstellen mit FORMANTGEN

FORMANTGEN ist das Werkzeug zum Erstellen von Masken. Mit FORMANTGEN können Sie:

- Editieren: Sie bauen im Dialog das Bild der Maske auf und können dabei z.B. zwischen verschiedenen Zeichensätzen und Darstellungseigenschaften wählen, wie halbhell, invers usw.
- Feldeigenschaften definieren, die das Verhalten bei der Eingabe in die Maske bestimmen, z.B. numerische Felder, Großchreibung usw.
- Verwaltungsfunktionen ausführen, z.B. Auflisten, Ausdrucken usw.

3.1 Format und Maske, wie arbeitet FORMANTGEN?

FORMANTGEN unterscheidet zwischen Format und Maske. Ein Format ist ein logisches Abbild der später zu erzeugenden Maske. Das Format ist unabhängig von Hardwareeigenschaften, wie dem verwendeten Prozessor. Man kann es daher ohne Änderung auf andere Anlagen übertragen. Das ist mit einer Maske nicht möglich.

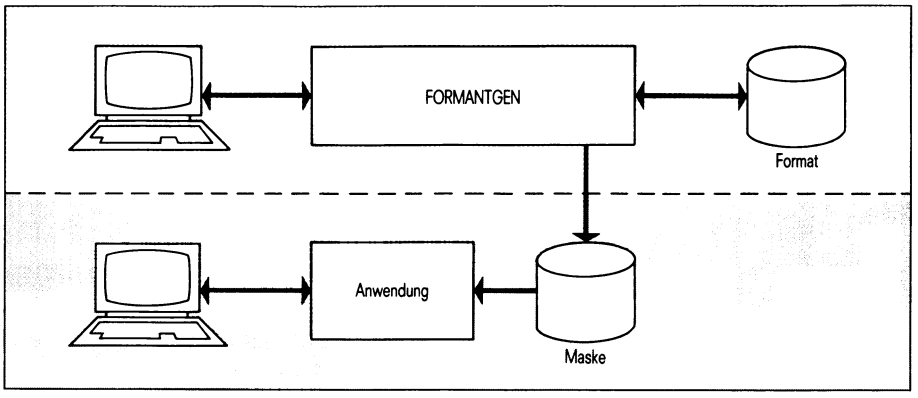


Bild 3-1: FORMANTGEN arbeitet mit Formaten, die Anwendung arbeitet mit Masken

FORMANTGEN arbeitet immer auf einem Bereich im Arbeitsspeicher. Darin liegt das aktuelle Format. Das aktuelle Format ist das, welches Sie gerade bearbeiten. Sind Sie fertig, können Sie:

- das Format als Datei in einem Dateiverzeichnis sichern,
- aus dem Format eine Maske generieren. Die generierte Maske speichert FORMANTGEN als Datei ab. Auf diese Datei greift Ihr FORMANT-Anwendungsprogramm zu um die Maske zu verwenden.

Für Formatdateien und Maskendateien kann man eigene Datzeiverzeichnisse anlegen, die Formatbibliothek und die Maskenbibliothek. Sie können auch mehrere Format- und Maskenbibliotheken verwenden.

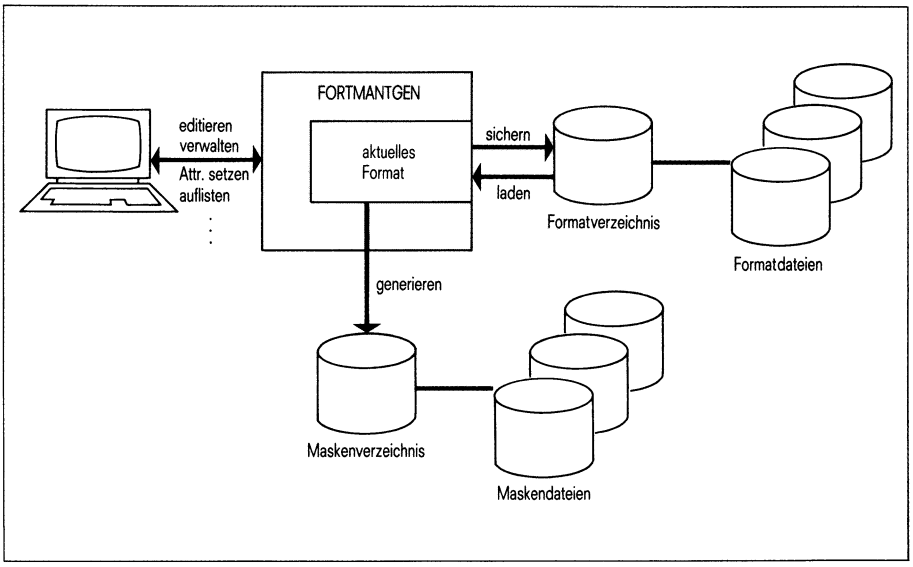


Bild 3-2: So arbeitet FORMANTGEN

Aus der Formatdatei kann FORMANTGEN ein Format laden. Sie können es dann ändern und ergänzen und anschließend erneut eine Maske generieren. Das aktuelle Format können Sie umbenennen, um es z.B. unter dem neuen Namen zu sichern ohne das Ausgangsformat zu zerstören.

Beim Generieren der Maske können Sie jedesmal den Dateinamen beliebig wählen.

Sie können auch absichtlich vorhandene Masken oder Formate überschreiben.

3.2 Ein Beispiel zum Erstellen einer Maske

FORMANTGEN lernen Sie am schnellsten kennen, wenn Sie sich an den Rechner setzen und es selbst versuchen. Das folgende Beispiel zeigt Ihnen alle wichtigen Arbeitsschritte in ihrer logischen Reihenfolge. Nebenher können Sie natürlich im Abschnitt 3.3 alle Funktionen nachschlagen. Die erzeugte Maske können Sie im Beispielprogramm in Kapitel 4 einsetzen.

1. Sie müssen in der Shell-Ebene arbeiten. Im Menüsystem ist FORMANTGEN nicht eingebaut.
2. Richten Sie je ein Dateiverzeichnis für Ihre Formate und Masken ein:

   mkdir formlib masklib

   Sie könnten auch ein gemeinsames Verzeichnis verwenden oder Formate und Masken im aktuellen Dateiverzeichnis ablegen.
3. Rufen Sie FORMANTGEN auf mit

   formantgen

   FORMANTGEN meldet sich mit dem Hauptmenü. Von nun an arbeiten Sie im Dialog mit FORMANTGEN.

Fehlermeldungen

Fehlermeldungen gibt FORMANTGEN zusammen mit einem Signalton aus. Die Tastatur ist dann gesperrt. Drücken Sie in diesem Fall [q]. Dann können Sie wieder eingeben. Die Fehlermeldungen sind selbsterklärend.

3.2.1 Format anlegen bzw. laden

```
                    >>>> FORMANTGEN V1.0 <<<<
                          Hauptmenue
-------------------------------------------------------------------------
l: Listen einer anzugebenden Formatbibliothek (= Directory in SINIX)
f: Format mit dem eingetragenen Namen im Arbeitsspeicher neu anlegen oder
   von der Platte in den Arbeitsspeicher laden (wenn es bereits existiert)
v: Verwalten bietet folgende Optionen:
        a: Anzeigen des im Arbeitsspeicher befindlichen Formates
        d: Drucken des im Arbeitsspeicher befindlichen Formates
        s: Sichern (= auf der Platte abspeichern) des Formates
        l: Loeschen (= auf der Platte loeschen) des Formates
        u: Umbenennen des im Arbeitsspeicher befindlichen Formates
e: Editieren des im Arbeitsspeicher befindlichen Formates
a: Attribute fuer die Bearbeitung der aus dem Format generierten Maske setzen
r: Reihenfolge der variablen Felder setzen
g: Generieren einer mit FORMANT V1.0 bearbeitungsfaehigen Maske

             Bitte vor dem Beenden das aktuelle Format sichern!

-------------------------------------------------------------------------
l: Listen der Formate      f: Formate anlegen/laden   v: Verwalten der Formate
e: Editieren               a: Attribute setzen        r: Reihenfolge festlegen
g: Generieren der Maske    d: Druckermaske            <: Beenden
-------------------------------------------------------------------------
Bitte auswaehlen (aefglrv<):
```

Drücken Sie die Taste [f]. Damit wählen Sie die Funktion Format anlegen/ laden. Erinnern Sie sich: Ein Format ist das logische Abbild der Maske, aus dem Sie dann die Maske generieren.

Im folgenden Bild tragen Sie den Namen Ihres Dateiverzeichnisses für Formate (Formatbibliothek) und den gewünschten Namen des Formates ein. Tippen Sie:

formlib [↵] formbsp1 [MENU]

```
>>>> FORMANTGEN V1.0 <<<<
Eingabe des Namens des Formates

Name der Formatbibliothek:    formlib_____________________

Name des Formates:            formbsp1_____

Bitte eingeben und danach MENU druecken!
```

Hier sehen Sie bereits, wie man bei einer FORMANT-Anwendung Daten eingibt. FORMANTGEN verhält sich wie eine FORMANT-Anwendung. Die Schreibmarke positionieren Sie wie in Abschnitt 2.3.3 beschrieben (siehe auch Tastaturbelegung im Anhang).

Mit [MENU] haben Sie die Nameneingabe abgeschlossen. Nun drücken Sie nochmals [f], um das aktuelle Format im Arbeitsspeicher anzulegen.

FORMANTGEN meldet: '13 Menueauswahl wird ausgefuehrt. Bitte warten.' Die Menüauswahl (f) wird invers angezeigt.

Erscheint die Meldung: 'Bibliothek existiert nicht oder anderer Eigentuemer', dann haben Sie das Verzeichnis nicht eingerichtet. Sie können dann 'n' wählen und die Angabe bei 'Name der Formatbibliothek' löschen. Dazu drücken Sie einfach die Taste [↵], wenn die Schreibmarke am Feldanfang steht. Anschließend drücken Sie wieder [MENU] und [f]. FORMANTGEN nimmt dann das aktuelle Dateiverzeichnis.

Drücken Sie [<]. Damit kehren Sie zum Hauptmenü zurück.

3.2.2 Format editieren

Drücken Sie [e]. FORMANTGEN zeigt folgendes Bild:

```
                        >>>> FORMANTGEN V1.0 <<<<
Format: formbsp1        Zeile: 1      Spalte: 1     Zeichensatz: Standard
```

```
Bitte editieren und danach MENU druecken!
```

Editieren heißt: Sie erstellen das Bild der späteren Maske, indem Sie alle gewünschten Felder am Bildschirm eintragen. Dazu gibt es zwei Bearbeitungsmodi:

1. Positioniermodus. In diesem Modus können Sie die Schreibmarke frei über den ganzen Bildschirm bewegen. Versuchen Sie es. Verwenden Sie die Tasten [←], [→], [↓] und [↑]. Die Taste [↖] positioniert die Schreibmarke nach links oben. [⇥] und [⇤] sind Tabulatortasten. Tabulatorpositionen sind 1,9,17 usw.
   [↵] positioniert auf den Beginn der nächsten Zeile.

2. Schreibmodus. In diesem Modus können Sie die Felder einsetzen. Dazu müssen sie den Feldanfang definieren:

   Setzen Sie die Schreibmarke auf den gewünschten Feldanfang.
   Drücken Sie [F1]. Damit öffnen Sie ein neues Feld.

   Jetzt geben Sie den Text Ihres Feldes ein. Am Feldende drücken Sie [F2], um das Feld abzuschließen.

   Im Schreibmodus können Sie die Schreibmarke nur innerhalb des Feldes bewegen.

Wie unterscheiden Sie Textfelder und variable Felder?

Die Eigenschaft Textfeld oder variables Feld vergeben Sie erst später beim Setzen der Attribute. Sie sollten sich aber für variable Felder ein bestimmtes Zeichen wählen, z.B. Unterstrich.

Definieren Sie 10 Felder wie im folgenden Bild. Vergessen Sie nicht, jedes Feld mit F2 abzuschließen.

```
                        >>>> FORMANTGEN V1.0 <<<<
 Format: formbsp1       Zeile: 14     Spalte: 35    Zeichensatz: Standard
-------------------------------------------------------------------------

      Name:          ---------------

      Vorname:       ---------------


      Strasse:       --------------------

      Ort:           --------------------

      Telefon:       ----------

-------------------------------------------------------------------------


-------------------------------------------------------------------------
Bitte editieren und danach MENU druecken!
```

Wie können Sie eine Eingabe korrigieren?

Im Positioniermodus können Sie Felder verschieben und Zeilen ein- oder ausfügen:

CHAR — Das folgende Feld wird nach rechts verschoben bei INS, nach links bei DEL (mit SHIFT). Die Schreibmarke muß außerhalb eines Feldes stehen.

LINE — Zeile einfügen bei INS, Zeile ausfügen bei DEL (wie beim Editor ced). Beim Einfügen einer Zeile geht die 24. Zeile verloren!

SHIFT WORD — Das Feld ausfügen, auf dem die Schreibmarke steht.

Weitere Möglichkeiten finden Sie in der vollständigen Übersicht in Abschnitt 3.3.4.

Im Schreibmodus können Sie Felder verändern. Wenn Sie ein Feld, das Sie ändern wollen, bereits abgeschlossen haben, müssen Sie es öffnen: positionieren Sie die Schreibmarke darauf und drücken Sie [F3].

[CHAR] Bei INS wird an der Schreibmarkenposition ein Leerzeichen eingefügt, bei DEL ein Zeichen ausgefügt. Das folgende Feld wird nicht verschoben. Beim Einfügen wird das Feld automatisch verlängert.

[⌫] Löscht ein Zeichen ab Schreibmarkenposition.

Probieren Sie die Wirkung dieser Tasten aus.

Wenn Sie ein Feld mit [F3] geöffnet haben, müssen Sie dieses nach dem Ändern abschließen: drücken Sie [F4].

Andere Zeichensätze

Bis jetzt haben Sie mit dem Zeichensatz 'Standard' gearbeitet. Ziehen Sie nun eine Linie:

1. Öffnen Sie an der gewünschten Position ein Feld mit [F1].
2. Drücken Sie [MENU] und dann [z]. Nun können Sie den Zeichensatz auswählen. Linienelemente befinden sich im Zeichensatz 'Klammern'. Drücken Sie also [k]. Nun können Sie in den Schreibmodus zurückkehren: [<] [e]. Alle jetzt eingegebenen Zeichen werden im gewählten Zeichensatz dargestellt. Die Zuordnung entnehmen Sie den Tabellen im Anhang. Einen geraden Strich für eine zusammenhängende Linie erhalten Sie z.B., wenn Sie [A] drücken.

```
                        >>>> FORMANTGEN V1.0 <<<<
Format: formbsp1        Zeile: 8      Spalte: 43    Zeichensatz: Klammern
-------------------------------------------------------------------------

      Name:          ---------------
      Vorname:       ---------------
      ____________________________

      Strasse:       --------------------
      Ort:           --------------------
      Telefon:       ----------

-------------------------------------------------------------------------


-------------------------------------------------------------------------
Bitte editieren und danach MENU druecken!
```

Ziehen Sie die Linie und schließen Sie das Feld ab.

3. Weitere neue Felder stellt FORMANTGEN automatisch im neu gewählten Zeichensatz dar bis Sie wieder auf einen anderen Zeichensatz umschalten.

Weitere Möglichkeiten

Wenn Sie [MENU] drücken, sehen Sie die weitere Auswahl.

f Farbe (ist noch nicht implementiert).

d Darstellungsattribute. Die Auswahl der Attribute geht wie die Auswahl der Zeichensätze. Sie können nacheinander mehrere Attribute auswählen, z.B halbhell und unterstrichen. [r] setzt alle Attribute zurück. Um d wählen zu können, müssen Sie ein Feld geöffnet haben ([F1] oder [F3]).

\+ Weiterblättern. Zunächst sehen Sie nur die Zeilen 1 bis 16. Nun werden die Zeilen 9 bis 24 gezeigt.

\- Zurückblättern auf die Zeilen 1 bis 16. FORMANTGEN blättert automatisch vor oder zurück, wenn Sie an die Bereichsgrenze stoßen.

Blättern können Sie nur im Positioniermodus (kein Feld geöffnet).

Editieren beenden

Gehen Sie in den Positioniermodus (Feld schließen mit F2 bzw. F4) und drücken Sie MENU, dann <. FORMANTGEN zeigt wieder das Hauptmenü.

3.2.3 Attribute setzen

Der nächste Schritt ist das Festlegen der Feldeigenschaften (Attribute). Mit den Attributen bestimmen Sie zum Beispiel den Feldtyp und wie FORMANT beim Eingeben Daten prüft. Der Feldtyp kann sein:

- fix: das ist ein Textfeld,
- variabel: das ist ein Feld, in das man Daten eingeben kann (variables Feld bzw. Datenfeld).

Alle Attribute sind erklärt in Abschnitt 3.4.

Drücken Sie a. FORMANTGEN zeigt folgendes Bild:

```
                    >>>> FORMANTGEN V1.0 <<<<
                      Auswaehlen des Feldes
-------------------------------------------------------------------------------

      Name:          ---------------
      Vorname:       ---------------
      ------------------------------

      Strasse:       --------------------
      Ort:           --------------------
      Telefon:       ----------

-------------------------------------------------------------------------------


-------------------------------------------------------------------------------
Bitte auswaehlen und danach MENU druecken!
```

Sie könnten nun die Schreibmarke auf das Feld setzen, das Sie als erstes bearbeiten wollen. Das ist nicht unbedingt nötig. FORMANTGEN beginnt dann beim ersten Feld. Drücken Sie MENU .

Die Auswahl ist ähnlich wie beim Editieren. Mit [+] und [-] können Sie blättern, mit [f] das Feld auswählen, mit dem begonnen wird und mit [<] zum Hauptmenü zurückkehren.

Drücken Sie [a] um die Attribute zu setzen. Sie erhalten folgendes Bild:

```
                    >>>> FORMANTGEN V1.0 <<<<
                    Setzen der Bearbeitungsattribute

Name:  _________ Vorname:  _________ ———————— Strasse:  _________ Ort:

F0_____ F1_____ F2_____ F3_____ F10____ F4_____ F5_____ F6_____
f       f       f       f       f       f       f       f
v       v       v       v       v       v       v       v
-----   -----   -----   -----   -----   -----   -----   -----
*       *       *       *       *       *       *       *
1       1       1       1       1       1       1       1
0_      0_      0_      0_      0_      0_      0_      0_
-       -       -       -       -       -       -       -
---     ---     ---     ---     ---     ---     ---     ---
-       -       -       -       -       -       -       -
-       -       -       -       -       -       -       -

Bitte Feldnamen eingeben!
```

In der Meldungszeile zeigt FORMANTGEN, welches Attribut gerade einzugeben ist, im Menübereich sehen Sie die verschiedenen Eingabemöglichkeiten.

Die Schreibmarke bewegen Sie mit den Tasten:

[→] und [←]	innerhalb eines Feldes,
[⇥] und [⇤]	von Feld zu Feld,
[↓] und [↑]	nach unten und oben,
[↖]	zum ersten Feld,
[↵]	zum nächsten Feld; der Inhalt ab Schreibmarkenposition wird gelöscht.

Für dieses Beispiel brauchen wir nur den Feldtyp festzulegen. Gehen Sie also in die zweite Zeile ([↓]) und übertippen Sie bei den Feldern, die variabel sein sollen, das 'f' mit einem 'v'. Variable Felder sind die, in die Daten eingegeben werden sollen. Diese Felder versehen Sie auch noch mit dem Attribut 'leer' (nächste Zeile). Würden Sie 'vorbelegt' wählen, dann würden die Felder von FORMANT mit den Zeichen ausgegeben, mit denen Sie sie definiert haben. Ihr Bildschirm sieht nun so aus:

```
                    >>>> FORMANTGEN V1.0 <<<<
                    Setzen der Bearbeitungsattribute

Name:  _________ Vorname:  _________ ———————— Strasse:  _________ Ort:

F0_____ F1_____ F2_____ F3_____ F10____ F4_____ F5_____ F6_____
f       v       f       v       f       f       v       f
v       l       v       l       v       v       l       v

_____   _____   _____   _____   _____   _____   _____   _____
*       *       *       *       *       *       *       *
l       l       l       l       l       l       l       l
0_      0_      0_      0_      0_      0_      0_      0_
-       -       -       -       -       -       -       -
---     ---     ---     ---     ---     ---     ---     ---
-       -       -       -       -       -       -       -
-       -       -       -       -       -       -       -


l: Leer                 v: Vorbelegt

Bitte auswaehlen (a-+<):
```

Sie werden bemerken, daß noch nicht alle Felder des Formates abgebildet sind. Drücken Sie [MENU]. Mit [+] können Sie nun noch die folgenden Felder anzeigen lassen. Um für diese Felder die Attribute zu setzen, drücken Sie wieder [a]. Verfahren Sie dabei wie vorher. Definieren Sie noch das Feld für die Telefonnummer numerisch (nur Ziffern).

Nun sind Sie fertig mit den Attributen. Kehren Sie zum Hauptmenü zurück: [MENU], dann [<] führt Sie ins letzte Teilmenü, nochmals [<] ins Hauptmenü.

3.2.4 Reihenfolge festlegen

Mit dieser Funktion legen Sie fest,

- in welcher Reihenfolge man in die variablen Felder Daten eingeben kann (Reihenfolge der Bearbeitung) und
- in welcher Reihenfolge die variablen Felder im Datensatz stehen sollen.

Wenn Sie die Reihenfolge nicht ändern wollen, lassen Sie die Funktion weg.

Drücken Sie [r] und anschließend [b] um die Reihenfolge der Bearbeitung festzulegen. Sie erhalten folgendes Bild:

```
Name:        ---------------
Vorname:     ---------------
             ---------------------------------

Strasse:     --------------------
Ort:         --------------------
Telefon:     ----------
```

```
Bitte positionieren, mit '>' markieren; dann MENU druecken!
```

Positionieren Sie die Schreibmarke auf das Feld, das das erste in der Reihenfolge sein soll. Dazu verwenden Sie die Tasten

[→|] nächstes Feld und
[|←] vorhergehendes Feld,

wobei Sie nicht über das letzte bzw. erste Feld hinauskommen.

[↖] erreicht immer das erste Feld der Reihenfolge.

Drücken Sie nun [>] um das gewünschte Feld zu markieren. Das markierte Feld verschwindet vom Bildschirm, so daß Sie gleich sehen, welche Felder noch nicht eingeordnet sind. Markieren Sie alle variablen Felder in der gewünschten Reihenfolge.

Haben Sie das letzte Feld markiert, fragt FORMANTGEN, ob die Reihenfolge wirksam werden soll. Antworten Sie mit 'j'.

Wenn Sie erneut [b] drücken, können Sie die gewählte Reihenfolge überprüfen und evtl. neu markieren.

Brechen Sie das Markieren vor dem letzten Feld ab ([MENU]), hängt FORMANTGEN die noch nicht markierten Felder als letzte an.

Beachten Sie: Sie können mit dieser Funktion nur variable Felder bearbeiten. Die Anordnung der Felder auf dem Bildschirm ändert sich damit nicht.

Genauso wie die Reihenfolge der Bearbeitung legen Sie die Reihenfolge im Datensatz fest.

Mit der Menüauswahl 'B' bzw. 'D' können Sie die geometrischesche Reihenfolge wieder herstellen (von links oben nach rechts unten).

Wenn Sie fertig sind, gehen Sie zurück zum Hauptmenü.

3.2.5 Format sichern und andere Verwaltungsfunktionen

Das Format ist die Arbeitsgrundlage für FORMANTGEN. Sie brauchen es, wenn Sie eine fertige Maske abändern oder ergänzen wollen. Das aktuelle Format, das Sie bearbeitet haben, befindet sich bis jetzt nur im Arbeitsspeicher. Damit Sie es später wieder verwenden können, müssen Sie es in einer Datei abspeichern.

Wählen Sie v: Verwaltungsfunktionen. Sie erhalten folgendes Bild:

```
                    >>>> FORMANTGEN V1.0 <<<<
                Teilmenue: Verwalten des aktuellen Formates
-------------------------------------------------------------------------------
l: Listen einer anzugebenden Formatbibliothek (= Directory in SINIX)
f: Format mit dem eingetragenen Namen im Arbeitsspeicher neu anlegen oder
   von der Platte in den Arbeitsspeicher laden (wenn es bereits existiert)
v: Verwalten bietet folgende Optionen:
        a: Anzeigen des im Arbeitsspeicher befindlichen Formates
        d: Drucken des im Arbeitsspeicher befindlichen Formates
        s: Sichern (= auf der Platte abspeichern) des Formates
        l: Loeschen (= auf der Platte loeschen) des Formates
        u: Umbenennen des im Arbeitsspeicher befindlichen Formates
e: Editieren des im Arbeitsspeicher befindlichen Formates
a: Attribute fuer die Bearbeitung der aus dem Format generierten Maske setzen
r: Reihenfolge der variablen Felder setzen
g: Generieren einer mit FORMANT V1.0 bearbeitungsfaehigen Maske

             Bitte vor dem Beenden das aktuelle Format sichern!

-------------------------------------------------------------------------------
a: Anzeigen des Formates   d: Drucken des Formates
s  Sichern des Formates    l: Loeschen des Formates   u: Umbenennen des Formates
                                                      <: Hauptmenue
-------------------------------------------------------------------------------
Bitte auswaehlen (adlsu<):
```

Wenn Sie nun [s] drücken, sichert FORMANTGEN das aktuelle Format als Datei. Der Dateiname ist gleich dem Formatnamen. Diesen und den Namen des Dateiverzeichnisses haben Sie beim Anlegen des Formates angegeben (siehe Abschnitt 3.2.1). Mit denselben Angaben können Sie später das Format wieder laden.

Falls schon eine Datei dieses Namens existiert, fragt FORMANTGEN, ob sie überschrieben werden soll. Wenn Sie mit 'n' antworten, sichert FORMANTGEN das Format nicht.

Probieren Sie noch andere Verwaltungsfunktionen aus:

[a] zeigt das aktuelle Format ganz am Bildschirm (Zeile 1 bis Zeile 24).

[d] druckt das aktuelle Format am Drucker aus (allerdings nur mit dem Standardzeichensatz).

[u] Umbenennen des aktuellen Formates. Sie geben wieder die Namen der Datei und des und Dateiverzeichnisses an, wie beim Anlegen des Formates. Damit hat nur das aktuelle Format im Arbeitsspeicher einen anderen Namen, nicht die Datei, in der Sie das Format gesichert haben. Erst wenn Sie erneut sichern, wird das aktuelle Format mit den neuen Angaben abgespeichert.

[l] Löschen des Formates. Diese Funktion probieren Sie jetzt lieber nicht aus. Sie löscht die Datei mit dem gesicherten Format.

3.2.6 Maske generieren

Ein Anwendungsprogramm kann mit dem Format, das Sie nun erstellt haben, noch nichts anfangen. Sie müssen daraus erst eine Maske generieren. Gehen Sie zurück zum Hauptmenü und drücken Sie [g]. Im folgenden Bild geben Sie den Namen des Dateiverzeichnisses an, in dem FORMANTGEN die Maskendatei eintragen soll und den Namen der Maske. Das Dateiverzeichnis haben Sie vor dem Aufruf von FORMANTGEN eingerichtet. Wenn Sie keines eingerichtet haben, füllen Sie das Feld für den Namen der Maskenbibliothek nicht aus. FORMANTGEN verwendet dann das aktuelle Dateiverzeichnis.

```
>>>> FORMANTGEN V1.0 <<<<
Eingabe des Namens der Maske

Name der Maskenbibliothek:    masklib_____________________

Name der Maske:               maskep_______

Bitte eingeben und danach MENU druecken!
```

Drücken Sie [MENU] und anschließend nochmals [g] um die Maske zu generieren.

Ist bereits eine Datei gleichen Namens vorhanden, fragt FORMANTGEN, ob sie überschrieben werden darf. Wenn Sie mit 'n' antworten, generiert FORMANTGEN die Maske nicht.

Damit sind Sie fertig und können FORMANTGEN beenden.

Wenn Sie ein weiteres Format erstellen wollen ...

dann benennen Sie das aktuelle Format um und gehen über das Hauptmenü in die Funktion Editieren. Nun können Sie

- auf der Grundlage des umbenannten Formates weitermachen oder
- das aktuelle Format im Arbeitsspeicher löschen mit [F16]. Sie können dann völlig neu mit dem Editieren beginnen.

Sie können auch im Hauptmenü 'f' wählen, um ein neues Format anzulegen. Dabei wird das aktuelle Format im Arbeitsspeicher gelöscht.

3.2.7 FORMANTGEN beenden und die Maske ausprobieren

Gehen Sie zum Hauptmenü und drücken Sie [<]. Damit Sie nicht versehentlich FORMANTGEN beenden, fragt FORMANTGEN nochmals nach. Bei 'j' beendet sich das Programm.

Achtung! Wenn Sie Ihr aktuelles Format nicht gesichert haben, ist es verloren! Nur Generieren der Maske genügt nicht, weil FORMANTGEN die Maskendatei nicht wieder lesen kann.

Sie können jetzt noch Ihre erstellte Maske ausprobieren. Dazu gibt es das Programm 'formant'. Es ist beschrieben in Abschnitt 8.1.3.

3.3 Alle Funktionen von FORMANTGEN kurz erklärt

Dieser Abschnitt enthält zu jeder Funktion eine kurze, aber vollständige Beschreibung. Während Sie im vorigen Abschnitt den logischen Ablauf und die Bedienung an einem Beispiel kennengelernt haben, können Sie hier jede Funktion nachschlagen.

Aufruf: formantgen

Das Hauptmenü ist der Ausgangspunkt für alle Funktionen. Sie sind in der Reihenfolge beschrieben, in der sie auf dem Menübildschirm stehen.

Die Funktion wählen sie durch Drücken der entsprechenden Taste.

Bildschirmaufbau

Der Bildschirm ist bei FORMANTGEN immer gleich aufgebaut:

```
                        >>>> FORMANTGEN V1.0 <<<<       Kopf
                              Hauptmenue
-------------------------------------------------------------------------
l: Listen einer anzugebenden Formatbibliothek (= Directory in SINIX)
f: Format mit dem eingetragenen Namen im Arbeitsspeicher neu anlegen oder
   von der Platte in den Arbeitsspeicher laden (wenn es bereits existiert)
v: Verwalten bietet folgende Optionen:
        a: Anzeigen des im Arbeitsspeicher befindlichen Formates
        d: Drucken des im Arbeitsspeicher befindlichen Formates
        s: Sichern (= auf der Platte abspeichern) des
        l: Loeschen (= auf der Platte loeschen) des F     Arbeitsbereich
        u: Umbenennen des im Arbeitsspeicher befindli
e: Editieren des im Arbeitsspeicher befindlichen Formates
a: Attribute fuer die Bearbeitung der aus dem Format generierten Maske setzen
r: Reihenfolge der variablen Felder setzen
g: Generieren einer mit FORMANT V1.0 bearbeitungsfaehigen Maske

              Bitte vor dem Beenden das aktuelle Format sichern!

-------------------------------------------------------------------------
l: Listen der Formate      f: Formate anlegen/laden
e: Editieren               a: Attribute setzen            Menübereich
g: Generieren der Maske    d: Druckermaske
-------------------------------------------------------------------------
Bitte auswaehlen (aefglrv<):                              Meldungszeile
```

Im Kopf finden Sie Informationen zur laufenden Funktion.

Im Arbeitbereich bearbeiten Sie Ihr Format oder es sind darin Informationen zur Bedienung von FORMANTGEN.

Im Menübereich finden Sie die aktuellen Auswahlmöglichkeiten.

Die Meldungszeile enthält Hinweise und Fehlermeldungen.

Dateneingabe

Wenn Daten einzugeben sind (z.B Formatname oder beim Setzen der Attribute), benutzt FORMANTGEN dieselben Tasten wie eine normale FORMANT-Anwendung. Das heißt es gilt die Beschreibung in Abschnitt 2.3.3 und die Tastaturbelegung im Anhang. Die wichtigsten Tasten:

→ ←	zum Positionieren der Schreibmarke im Feld,
⌫	ein Zeichen löschen (rückwärts),
R L	nächstes bzw. vorhergehendes Feld,
↵	nächstes Feld, löschen ab Schreibmarkenposition,
↖	erstes Feld,
SHIFT WORD	Feld löschen,
F16	alle Felder löschen.

Zusätzlich geht:

CHAR	bei INS ein Leerzeichen einfügen, bei DEL ein Zeichen ausfügen.

Fehlermeldungen

Fehlermeldungen sperren die Tastatur.

q oder Q entsperren die Tastatur wieder.

FORMANTGEN beenden

FORMANTGEN können Sie nur aus dem Hauptmenü heraus beenden.

< : Zur Sicherheit fragt FORMANTGEN, ob Sie wirklich beenden wollen. Wenn Sie mit 'n' antworten, beendet sich FORMANTGEN nicht.

Ein aktuelles Format, das Sie nicht gesichert haben, ist verloren!!!

3.3.1 Listen einer Formatbibliothek

Mit dieser Funktion können Sie sich alle Dateien eines Dateiverzeichnisses auflisten lassen, die Formate enthalten. In der Liste können Sie einen Namen markieren, um den Namen des aktuellen Formates festzulegen.

Menüauswahl

```
n: Namen der Bibliothek    l: Listen der Formate      e: Format eintragen
\: Zeile nach oben         /: Zeile nach unten
-: Scrollen nach oben      +: Scrollen nach unten     <: Hauptmenue

Bitte auswaehlen (ln\/-+<):
```

[n] Name der Bibliothek eingeben. Das ist der Name des Dateiverzeichnisses, das nach Formaten durchsucht wird. Sie können auch Pfadnamen angeben.

Standard (nichts angegeben): FORMANTGEN durchsucht das aktuelle Dateiverzeichnis.

[l] Die Dateien werden aufgelistet, die Formate enthalten. Der erste Name ist markiert.

[e] Der markierte Name wird als aktueller Formatname eingetragen. Bei der Funktion 'Format anlegen/laden' ist dieser Name Voreinstellung. Wenn Sie schon ein Format geladen haben, ändern Sie damit den Namen dieses Formates auf den markierten Namen. Das ist dieselbe Wirkung wie bei der Funktion 'Umbenennen'. Geben Sie acht beim Sichern, da Sie auf den Namen eines vorhandenen Formates umbenennen.

[\] Markierung um eine Zeile nach oben setzen.

[/] Markierung um eine Zeile nach unten setzen.

[-] Vorhergehende Zeilen der Liste zeigen
(nur wenn di Liste länger als eine Bildschirmseite ist).

[+] Nachfolgende Zeilen der Liste zeigen
(nur wenn di Liste länger als eine Bildschirmseite ist).

[<] Zurück zum Hauptmenü.

3.3.2 Format anlegen oder laden

Mit dieser Funktion wird der Name des aktuellen Formates bestimmt und die Formatbibliothek (Dateiverzeichnis) festgelegt, in die das Format später zu sichern ist. Ist ein Format dieses Namens in der angegeben Formatbibliothek vorhanden, wird es geladen.

Das aktuelle Format im Arbeitsspeicher wird gelöscht.

Namen eingeben

Name der Formatbibliothek

Anzugeben ist der Name eines Dateiverzeichnisses, in das die Formate beim Sichern einzutragen sind. Das Dateiverzeichnis muß bereits eingerichtet sein. Sie können auch einen Pfadnamen angeben.

Standard (keine Angabe): das aktuelle Dateiverzeichnis.

Name des Formates

Unter diesem Namen wird das Format beim Sichern abgespeichert. Ein Name muß angegeben werden.

Namenskonventionen: Zulässig sind

- Groß- und Kleinbuchstaben,
- Ziffern,
- das Zeichen '/'.

Menüauswahl

```
n: Namen des Formates      f: Format anlegen/laden

                                          <: Hauptmenue

Bitte auswaehlen (fn<):
```

[n] Namen ändern, um z.B. ein anderes Format zu laden.

[f] Die Funktion wird ausgeführt, wenn gültige Namen angegeben wurden. Dazu erscheint die Meldung 'Menueauswahl wird ausgefuehrt. Bitte warten!'

[<] Zurück zum Hauptmenü.

3.3.3 Verwalten des aktuellen Formates

Die Funktion Verwalten hat mehrere Unterfunktionen.

Menüauswahl

```
a: Anzeigen des Formates   d: Drucken des Formates
s  Sichern des Formates    l: Loeschen des Formates   u: Umbenennen des Formates
                                                      <: Hauptmenue

Bitte auswaehlen (adlsu<):
```

[a] Anzeigen des aktuellen Formates. Auf dem Bildschirm erscheint das aktuelle Format in voller Größe und Schönheit (Zeile 1 bis 24).

[d] Drucken des aktuellen Formates. Das Format wird über das SINIX-Kommando lpr ausgedruckt. Es wird nur der Standardzeichensatz verwendet. FORMANTGEN erzeugt eine temporäre Datei mit Namen druck_format.

[s] Sichern des aktuellen Formates. Das aktuelle Format wird als Datei abgespeichert. Der Dateiname ist gleich dem Formatnamen. Die Datei wird in das Dateiverzeichnis eingetragen, das Sie als Formatbibliothek angegeben haben (beim Laden oder beim Umbenennen des Formates). FORMANTGEN zeigt Ihnen den aktuellen Dateinamen.

Wollen Sie einen anderen Dateinamen wählen, müssen Sie vor dem Sichern das aktuelle Format umbenennen (siehe unten).

Ist eine Datei dieses Namens in der Formatbibliothek vorhanden, fragt FORMANTGEN, ob sie überschrieben werden soll. Wenn Sie mit 'n' antworten, wird das Format nicht gesichert. Dateien, die kein Format enthalten kann FORMANTGEN nicht überschreiben.

[l] Löschen des gesicherten Formates. FORMANTGEN löscht die Datei mit dem Namen des aktuellen Formats aus der Formatbibliothek, die Sie beim Laden oder Umbenennen angegeben haben. FORMANTGEN zeigt Ihnen den aktuellen Dateinamen. Das aktuelle Format im Arbeitsspeicher bleibt erhalten.

Vor dem Löschen fragt FORMANTGEN zur Sicherheit nochmals ab, ob Sie wirklich löschen wollen. Dateien, die kein Format enthalten, kann FORMANTGEN nicht löschen.

[u] Umbenennen des aktuellen Formates. Damit ändern Sie

- den Namen des aktuellen Formates im Arbeitsspeicher (nicht aber den Dateinamen eines bereits gesicherten Formats). Wenn Sie anschließend sichern, wird das Format unter dem neuen Namen abgespeichert.
- den Namen der Formatbibliothek (Dateiverzeichnis), in der das Format beim Sichern einzutragen ist.

Eingabe und Namenskonventionen wie beim Anlegen/Laden.

3.3.4 Editieren

Beim Editieren definieren Sie die Felder des Formates, sowie Zeichensatz und Darstellung am Bildschirm.

Positioniermodus

Der Positioniermodus ist zu Beginn eingeschaltet. Im Positioniermodus können Sie die Schreibmarke frei über den Bildschirm bewegen (siehe Tasten). Stoßen Sie an die untere bzw. obere Bereichsgrenze, blättert FORMANTGEN automatisch vor bzw. zurück.

Sie können in diesem Modus nicht schreiben. Die Position der Schreibmarke bestimmt

- den Beginn eines neuen Feldes, wenn Sie nicht innerhalb eines vorhandenen steht oder
- das Feld, das Sie zum Ändern öffnen wollen, wenn sie innerhalb eines vorhandenen Feldes steht.

Tasten im Positioniermodus:

F1	Neues Feld öffnen und in den Schreibmodus gehen. Die Schreibmarke definiert den Beginn des Feldes. Das Feldende wird beim Schreiben nach rechts geschoben.
CTRL a	Wie F1 (auf).
F3	Feld zum Ändern öffnen und in den Schreibmodus gehen. Die Schreibmarke muß innerhalb eines vorhandenen Feldes stehen.
CTRL u	Wie F3 (update).
→	Schreibmarke nach rechts.
←	Schreibmarke nach links.
↹	Tabulator rechts.
⇤↹	Tabulator links (Spalten 1,9,17, usw.).
↵	Anfang der nächsten Zeile.
CHAR	Nächstes Feld nach rechts verschieben bei INS. Nächstes Feld nach links verschieben bei DEL.
LINE	Zeile einfügen bei INS (24. Zeile geht verloren). Zeile ausfügen bei DEL.
F16	Aktuelles Format im Arbeitspeicher löschen.

F5	Feld teilen in zwei Felder. Die Schreibmarke ist dorthin zu positionieren, wo das zweite Feld beginnen soll.
CTRL c	Wie F5 (cut).
F6	Zwei nebeneinanderliegende Felder verbinden. Die Schreibmarke ist in das zweite Feld zu setzen. Die Felder müssen direkt nebeneiander liegen und alle Attribute gleich haben.
CTRL l	wie F6 (link).
WORD	Bei DEL: Feld löschen. Anschließen brauchen Sie nicht mehr F4 zu drücken.
F16	Aktuelles Format im Arbeitspeicher löschen.

Schreibmodus

Im Schreibmodus definieren Sie Lage und Inhalt der Felder. Sie gelangen in den Schreibmodus, wenn Sie ein Feld öffnen (siehe Tasten). Mit dem Schließen eines Feldes sind Sie wieder im Positioniermodus. Im Schreibmodus kann man die Schreibmarke nur innerhalb des geöffneten Feldes bewegen.

Textfelder definieren Sie, indem Sie den Feldinhalt hinschreiben.

Variable Felder definieren Sie ebenfalls durch eine beliebige Zeichenfolge. Beim Setzen der Attribute vergeben Sie erst die Eigenschaft 'variabel'. Dabei bestimmen Sie auch, ob FORMANT das Feld mit der Zeichenfolge vorbelegt oder leer ausgibt. Im ersten Fall ist die Zeichenfolge ein vorgegebener Feldinhalt. Im zweiten Fall ist sie nur eine Markierung, wo sich das Feld befindet und wie lang es ist.

Tasten im Schreibmodus:

F2	Neues Feld schließen und in den Positioniermodus gehen. Drückt man F2 unmittelbar nach F1, entsteht kein Feld.
CTRL z	Wie F2 (zu).
F4	Feld nach dem Ändern schließen und in den Positioniermodus gehen. Ein mit F3 geöffnetes Feld können Sie nur mit F4 schließen.
CTRL e	Wie F4 (exit).

[→] Schreibmarke nach rechts.
[←] Schreibmarke nach links.
[⇥] Schreibmarke an das aktuelle Feldende.
[⇤] Schreibmarke an den aktuellen Feldanfang.
[⌫] Ein Zeichen löschen (rückwärts).

[CHAR] Bei INS: ein Leerzeichen einfügen. Das Feldende verschiebt sich nach rechts.

Bei DEL: ein Zeichen ausfügen. Das Feldende verschiebt sich nach links.

[WORD] Bei DEL: Feld löschen. Anschließen brauchen Sie nicht mehr [F4] zu drücken.

[F16] Aktuelles Format im Arbeitspeicher löschen.

Ein Feld kann nicht über die Zeile hinweggehen, also höchstens 80 Zeichen lang sein. Ein Format kann höchstens 400 Felder haben.

Menüauswahl

Zur Menüauswahl gelangen Sie mit der Taste [MENU] .

```
e: Editieren
z: Zeichensatz          f: Farbe                d: Darstellungsattribute
-: Rueckwaerts          +: Vorwaerts            <: Hauptmenue

Bitte auswaehlen (defz-+<):
```

[e] Zurück zum Editieren.

[z] Zeichensatz auswählen. Die Funktion geht nur, wenn Sie ein Feld geöffnet haben. Die Wirkung ist sofort sichtbar, vorausgesetzt, das Feld hat schon einen Inhalt.

Untermenü:

[s] Standard (International).
[n] National (Deutsch).
[i] IBM-Zeichensatz (IBM).
[a] Alpha-Mosaik (MOSAICS).
[k] Klammern.
[m] Mathematisch (MATH).
[<] Zurück zum Teilmenü Editieren.

Die Zeichensätze sind im Anhang abgebildet. Dort finden Sie auch die Zuordnung zu den Tasten.

Wenn Sie ein neues Feld öffnen, erhält es den Zeichensatz, der für das letzte neue Feld eingestellt war. Zu Beginn oder wenn das Format im Arbeitsspeicher gelöscht wurde ist der Zeichensatz 'Standard' eingestellt. Dieser Zeichensatz wird auch im Kopf angezeigt.

Wenn Sie ein Feld zum Ändern öffnen, stellt FORMANTGEN den Zeichensatz dieses Feldes ein. Beim Schließen stellt FORMANTGEN den Zeichensatz wieder zurück.

[f] Farbe auswählen. Diese Funktion ist noch nicht implementiert.

[d] Darstellung auswählen. Die Funktion geht nur, wenn Sie ein neues Feld geöffnet haben oder im Positioniermodus die Schreibmarke auf einem vorhandenen Feld steht. Die Wirkung ist sofort sichtbar, vorausgesetzt, das Feld hat schon einen Inhalt.

Sie können mehrere Eigenschaften auswählen, soweit sie sich nicht von selbst ausschließen.

Untermenü:

[n] Normal. Das Feld wird hell dargestellt.
[h] Halbhell.

[d] Dunkel. Das Feld wird nicht dargestellt. Andere Darstellungseigenschaften sind dadurch unwirksam (außer invers).
[i] Invers.

[b] Blinkend.
[r] Rücksetzen. Das Feld wird standardmäßig hell dargestellt. i, u und b sind ausgeschaltet.
[<] Zurück zum Teilmenü Editieren.

Wenn Sie ein neues Feld öffnen, erhält es die Darstellung, die für das letzte neue Feld eingestellt war. Zu Beginn oder wenn das Format im Arbeitsspeicher gelöscht wurde, ist 'normal' eingestellt.

Wenn Sie ein Feld zum Ändern öffnen, stellt FORMANTGEN die Darstellung dieses Feldes ein. Beim Schließen stellt FORMANTGEN die Darstellung wieder zurück.

[-] Zeile 1 bis 16 des Formates anzeigen. Zum Blättern darf kein Feld geöffnet sein.

[+] Zeile 9 bis 24 des Formates anzeigen. Zum Blättern darf kein Feld geöffnet sein.

[<] Zurück zum Hauptmenü. Es darf kein Feld geöffnet sein.

3.3.5 Attribute setzen

Mit dieser Funktion definieren sie die Feldeigenschaften (Attribute). Alle Attribute sind beschrieben im Abschnitt 3.4. Dort finden Sie auch Verweise auf die FORMANT-Aufrufe, bei denen Attribute vorkommen.

Feld auswählen

FORMANTGEN kann für 8 aufeinanderfolgende Felder die Attributdaten zeigen. Mit der Schreibmarke können Sie das erste dieser Felder auswählen. Steht die Schreibmarke nicht auf einem Feld, beginnt FORMANTGEN beim ersten Feld.

Menüauswahl

```
f: Feld auswaehlen          a: Attribute setzen

-: Rueckwaerts              +: Vorwaerts              <: Hauptmenue

Bitte auswaehlen (af-+<):
```

[f] Zurück zum Feld auswählen (wie oben beschrieben).

[a] Attribute setzen. FORMANTGEN zeigt die Attributdaten für 8 Felder. Die gewünschten Attribute geben Sie nacheinander ein. Zusätzlich zu den unter Dateneingabe (am Beginn des Abschnitts 3.3) beschriebenen Tasten können Sie nach oben und unten positionieren:

[↑] Darüberliegendes Feld.
[↓] Darunterliegendes Feld.

Je nach Position der Schreibmarke zeigt FORMANTGEN

- in der Meldungszeile, welches Attribut einzugeben ist.
- im Menübereich, welche Werte zulässig sind.

Die Erklärungen der Attribute finden Sie in Abschnitt 3.4. FORMANTGEN prüft die Eingaben nur syntaktisch, jedoch nicht ob sie sinnvoll sind.

Wenn Sie [MENU] drücken erhalten Sie ein Untermenü:

[a] Weiter Attribute setzen.
[−] Vorhergehende Felder anzeigen.
[+] Nachfolgende Felder anzeigen.
[<] Zurück zum Teilmenü Attribute setzen.

[−] Zeile 1 bis 16 des Formates anzeigen.

[+] Zeile 9 bis 24 des Formates anzeigen.

[<] Zurück zum Hauptmenü.

3.3.6 Reihenfolge der variablen Felder festlegen

Mit dieser Funktion legen Sie fest,

- in welcher Reihenfolge man in die variablen Felder Daten eingeben kann (Reihenfolge der Bearbeitung) und
- in welcher Reihenfolge die variablen Felder im Datensatz stehen sollen.

Die Reihenfolge der Bearbeitung wirkt sich bei einem Leseaufruf aus (read bzw. write read). Die Reihenfolge im Datensatz wirkt sich beim Aufruf 'get record' aus.

Die Funktion bearbeitet nur variable Felder. Sind keine variablen Felder im Format vorhanden, kann man sie gar nicht aufrufen.

Menüauswahl

```
b: Bearbeitung              d: Datensatz
B: BRF ruecksetzen          D: DRF ruecksetzen
                                              <: Hauptmenue

Bitte auswaehlen (bdBD<):
```

[b] Reihenfolge der Bearbeitung festlegen. FORMANTGEN zeigt das ganze Format an (Zeile 1 bis Zeile 24). Die Felder sind in der gewünschten Reihenfolge zu markieren. Markierte Felder werden gelöscht.

[d] Reihenfolge im Datensatz festlegen. FORMANTGEN zeigt das ganze Format an (Zeile 1 bis Zeile 24). Die Felder sind in der gewünschten Reihenfolge zu markieren. Markierte Felder werden gelöscht.

[B] Reihenfolge in der Bearbeitung rücksetzen auf die geometrische Reihen-folge (von links oben nach rechts unten).

[D] Reihenfolge im Datensatz rücksetzen auf die geometrische Reihenfolge.

[<] Zurück zum Hauptmenü.

Tasten beim Festlegen der Reihenfolge:

[↹]	nächstes Feld in der aktuell eingestellten Reihenfolge (nur bis zum letzten Feld).
[⇤↹]	Vorhergehendes Feld in der aktuell eingestellten Reihenfolge (nur bis zum ersten Feld).
[↖]	Erstes Feld in der aktuell eingestellten Reihenfolge.
[>]	Markieren des Feldes. Das markierte Feld wird am Bildschirm gelöscht.
[MENU]	Markieren beenden. FORMANTGEN fragt, ob die gewählte Reihenfolge wirksam werden soll. Wenn Sie mit 'n' antworten, bleibt die vorher vorhandene Reihenfolge erhalten.

Wenn Sie das Markieren vor dem letzten Feld beenden, hängt FORMANTGEN die restlichen Felder in geometrischer Reihenfolge hinten an.

Nachprüfen können Sie die Reihenfolge, indem Sie nochmals b bzw. d wählen. Mit [↹] durchlaufen Sie alle Felder in der neuen Reihenfolge.

3.3.7 Generieren der Maske

Mit dieser Funktion erzeugen Sie eine Maske, auf die das FORMANT-Anwendungsprogramm zugreifen kann. Die Maske wird in einer Datei abgespeichert. Für die Maskendateien können Sie ein eigenes Dateiverzeichnis festlegen, die Maskenbibliothek.

Namen eingeben

Name der Maskenbibliothek

Anzugeben ist der Name eines Dateiverzeichnisses, in das die Maske einzutragen ist. Das Dateiverzeichnis muß bereits eingerichtet sein. Sie können auch einen Pfadnamen angeben.

Standard (keine Angabe): das aktuelle Dateiverzeichnis.

Name der Maske

Unter diesem Namen wird die Maske abgespeichert. Ein Name muß angegeben werden. Für COBOL-Anwendungen darf der Maskenname höchstens 8 Zeichen lang sein.

Namenskonventionen: Zulässig sind

- Groß- und Kleinbuchstaben,
- Ziffern,
- das Zeichen '/'.

Menüauswahl

```
n: Namen eingeben          g: Generieren

                                          <: Hauptmenue

Bitte auswaehlen (gn<):
```

[n] Namen ändern um z.B. eine weitere Maske zu generieren.

[g] Die Funktion wird ausgeführt, wenn gültige Namen angegeben wurden. Dazu erscheint die Meldung 'Menueauswahl wird ausgefuehrt. Bitte warten!'

[<] Zurück zum Hauptmenü.

3.3.8 Druckermaske generieren

Mit dieser Funktion können Sie Masken generieren, die eine dem Drucker angemessene Anzahl von Zeilen und Spalten haben. Möglich sind maximal:

72 Zeilen
136 Spalten

Das entspricht einer ganz bedruckten Seite DIN A4. Um eine Druckermaske auszugeben, müssen Sie im Anwendungsprogramm den Sitzungsparameter GERAETEKLASSE auf den Wert DRUCKER setzen (Aufruf 'modify session parameters').

Eine Druckermaske setzen Sie aus bis zu 6 Formaten zusammen. Diese Teilformate erstellen Sie einzeln. Beim Generieren mit dieser Funktion setzt FORMANTGEN die angegebenen Formate wie Kacheln zu einer Maske zusammen.

Namen eingeben

Bibliothek
: Name des Dateiverzeichnisses, in dem das jeweilige Format liegt (Formatbibliothek).

  Standard (keine Angabe): das aktuelle Dateiverzeichnis.

Format
: Name des Teilformates. Die Teilformate sind nach den unten beschriebenen Regeln zu erstellen. Jedes Teilformat muß als Datei vorhanden sein, das heißt Sie müssen es gesichert haben.

Name der Maskenbibliothek
: Name des Dateiverzeichnisses, in dem die generierte Druckermaske liegen soll. Das Dateiverzeichnis muß existieren.

  Standard (keine Angabe): das aktuelle Dateiverzeichnis.

Name der Maske
: Unter diesem Namen wird die Maske als Datei abgespeichert. Sie müssen einen Namen angeben. Für COBOL-Anwendungen darf der Maskenname höchstens 8 Zeichen lang sein.

```
                    >>>> FORMANTGEN V1.0 <<<<
                Teilmenue: Generieren einer Druckermaske
-------------------------------------------------------------------------

Bibl.: ------------------------------  Bibl.: ------------------------------
Format: --------------                 Format: --------------

Bibl.: ------------------------------  Bibl.: ------------------------------
Format: --------------                 Format: --------------

Bibl.: ------------------------------  Bibl.: ------------------------------
Format: --------------                 Format: --------------

        Name der Maskenbibliothek:     ------------------------------

        Name der Maske:                --------------

-------------------------------------------------------------------------
n: Namen eingeben          g: Generieren

                                                    <: Hauptmenue
-------------------------------------------------------------------------
Bitte auswaehlen (gn<):
```

Menüauswahl

[n] Namen ändern, um z.B. eine weitere Maske zu generieren.

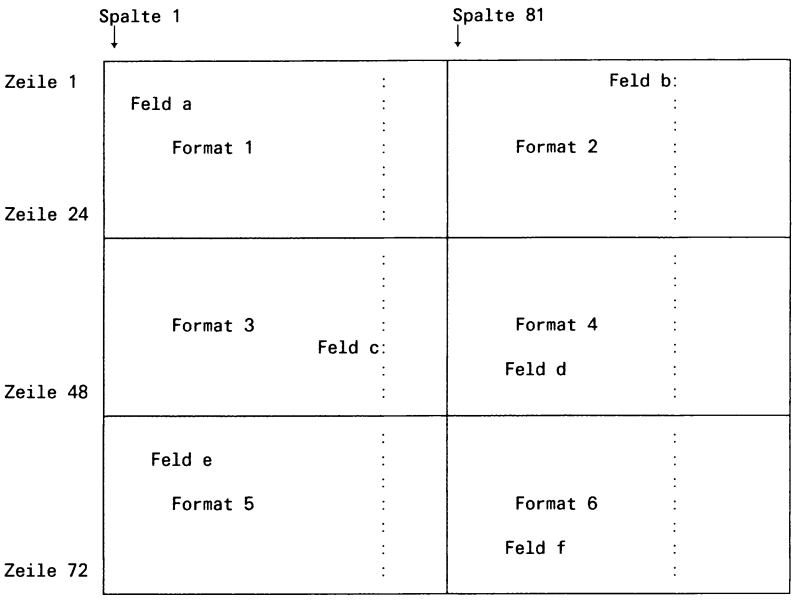
[g] Die Funktion wird ausgeführt, wenn gültige Namen angegeben wurden. Dazu erscheint die Meldung 'Menueauswahl wird ausgefuehrt. Bitte warten!'

[<] Zurück zum Hauptmenü.

Wie erstellen Sie die Teilformate?

Jedes Teilformat erstellen Sie einzeln wie jedes andere Format: Editieren, Attribute setzen, sichern.

Beim Editieren müssen Sie beachten, wie die Formate später zusammengesetzt werden:



FORMANTGEN verwendet nur die grün unterlegten Bereiche. Die Grenzen nach oben und unten sind fest. Die rechten Grenzen sind bestimmt durch das Feld, das am weitesten rechts endet. Im Beispiel oben sind das Feld c für Format 1, 3 und 5 und Feld b für Format 2, 4 und 6.

Die gesamte Breite für eine Druckermaske ist maximal 132 Spalten.

Ein Feld kann nicht über eine Teilformatgrenze hinausgehen. Das ist durch das Verfahren bedingt. Um die Breite eines Bereichs festzulegen, können Sie Dummyfelder definieren. Das sind z.B. feste Felder, die mit Leerzeichen vorbelegt werden.

Die Funktion 'Reihenfolge festlegen' hat für Druckermasken keine Wirkung. Es gilt die geometrische Reihenfolge (von links oben nach rechts unten).

Wichtig: Alle Feldnamen der verwendeten Formate müssen verschieden sein! Wenn Sie keine eigenen Namen definiert haben, müssen Sie die Standardnamen des zweiten und der weiteren Formate abändern.

Unterstützte Zeichensätze

FORMANT V1.0 unterstützt nur die Zeichensätze Standard bzw. National entsprechend dem am Drucker (oder mit lpr) eingestellten Zeichensatz.

3.4 Die Attribute - wichtig für den Programmierer

Hier finden Sie alle Attribute, die Sie mit FORMANTGEN definieren können. Sie stehen in derselben Reihenfolge wie bei der Funktion Attribute setzen.

Bitte Feldnamen eingeben !

Name des Feldes. Mit diesem Namen wird das Feld bei FORMANT-Aufrufen angesprochen. Der Name muß mindestens 2 Zeichen lang sein. Sie dürfen keinen Namen doppelt vergeben.
FORMANTGEN vergibt Standardnamen (F0, F1, ...) in der Reihenfolge, wie die Felder definiert wurden.

Bitte Feldtyp eingeben !

Variabel	Das Feld ist ein Datenfeld. Es wird zur Dateneingabe benutzt, sofern es nicht geschützt ist. Der Inhalt aller variablen Felder bildet den Datensatz.
Fix	Das Feld ist ein Textfeld. Ein Textfeld kann am Bildschirm nicht verändert werden. Es ist nicht im Datensatz vorhanden.

Bitte Feldinhalt beim Generieren eingeben !

Leer	Das Feld wird als leeres Feld ausgegeben, d.h. es wird mit Leerzeichen dargestellt (Bildschirmfüllzeichen.
Vorbelegt	Das Feld wird mit dem Text ausgegeben, mit dem Sie es definiert haben (Standard für Textfelder). Bei variablen Feldern können Sie die Vorbelegung benutzen, um Standardwerte vorzugeben.

Bitte eingeben !

Mussfeld (m)

In ein Mussfeld ist mindestens ein Zeichen einzugeben. Sonst kann man es nicht verlassen und FORMANT meldet einen Eingabefehler.

Ausweislesefeld (a)

In ein Ausweislesefeld kann man Daten nur über einen eingebauten Ausweisleser eingeben und nicht über die Tastatur. FORMANT übernimmt soviele Daten aus dem Ausweis, wie in das Feld passen. Ist kein Ausweisleser vorhanden, behandelt FORMANT das Feld wie ein normales Eingabefeld. Zur Dateneingabe mit dem Ausweisleser siehe Abschnitt 2.3.1.

Skipfeld (s)

Wird ein Feld mit diesem Attribut übersprungen, füllt FORMANT es mit Zeichen X'7F'. Dadurch kann das Anwendungsprogramm erkennen, ob das Feld bei der Eingabe übersprungen oder ob etwas eingegeben wurde.

Vollständigkeitsfeld (v)

Wenn in das Feld etwas eingegeben wurde, muß man es vollständig ausfüllen bis zum Feldende. Sonst meldet FORMANT einen Eingabefehler. Man kann es aber überspringen.

Duplizierfeld (d)

In ein Duplizierfeld übernimmt FORMANT automatisch den entsprechenden Inhalt aus dem Dupliziersatz. Dazu muß ein Dupliziersatz gesetzt sein (siehe Aufruf 'set dup record' und Abschnitt 2.3.5). Sonst wird das Attribut ignoriert.

Geschütztes Feld (g)

In das Feld kann man keine Daten eingeben. Das Feld muß ein variables Feld sein. Das Anwendungsprogramm kann es verändern und sein Inhalt ist Bestandteil des Datensatzes.

Triggerfeld (t)

Beim Verlassen des Feldes ruft FORMANT die dafür definierte Benutzerroutine auf. Ist für das Feld keine Benutzerroutine definiert, wird das Attribut ignoriert (siehe Aufruf 'define user exits' und Kapitel 7.

Sie können fünf dieser Eigenschaften kombinieren. Die folgende Tabelle zeigt, welche Eigenschaft mit welcher kombinierbar ist.

	m	a	s	v	d	g	t
m	-	j	n	j	j	n	j
a	j	-	j	j	n	n	j
s	n	j	-	j	j	n	j
v	j	j	j	-	j	n	j
d	j	n	j	j	-	j	j
g	n	n	n	n	j	-	n
t	j	j	j	j	j	n	-

j Die Kombination ist sinnvoll.

n Die Kombination ist nicht sinnvoll.

Tabelle 3-1: Kombination von Bearbeitungsattributen

Bitte den zugelassenen Zeichenvorrat eingeben !

Alle Zeichen	Alle abdruckbaren Zeichen.
Buchstaben	A-Z, a-z, ' ' (Leerzeichen).
Alphanumerisch	A-Z, a-z, 0-9, ' ' (Leerzeichen)
Numerisch	0-9

In das Feld kann man nur die jeweils zugelassene Menge von Zeichen eingeben. Sonst meldet FORMANT einen Eingabefehler.

Bitte die Feldausrichtung eingeben !

Linksbündig	Das Feld wird links ausgerichtet.
Rechtsbündig	Das Feld wird rechts ausgerichtet.

Die Ausrichtung gilt für Ausgabe und Eingabe. Das Feld wird sowohl im Datensatz entsprechend ausgerichtet als auch beim Aufbereiten am Bildschirm.

Bitte die Anzahl der Nachkommastellen (0-15) eingeben !

0-15	Definiert die Anzahl der Nachkommastellen. Bei 0 werden keine Nachkommastellen vorgesehen.

Diese Feldeigenschaft wirkt nur bei numerischen Feldern. Bei der Eingabe muß man einen Dezimalpunkt setzen. Das Feld muß lang genug sein, um alle Stellen und den Dezimalpunkt aufzunehmen. Das Feld wird im Datensatz mit Dezimalpunkt abgespeichert. Beispiele finden Sie in Abschnitt 2.3.1.
Anstelle des Punktes kann man auch Komma eingeben. Es wird aber immer ein Punkt abgebildet.

Bitte die entsprechende Feldeigenschaft eingeben !

Vorzeichen — Das Feld muß ein numerisches Feld sein. Schließt man es mit der Taste [CE] ab (FE-) ab, dann setzt FORMANT an die rechteste Stelle das Vorzeichen '-'. Sonst steht an dieser Stelle ein Leerzeichen. Das gilt für den Bildschirm. Im Datensatz steht das Vorzeichen links.

Großschreibung — Gibt man Kleinbuchstaben ein, wandelt Sie FORMANT sofort in Großbuchstaben um.

Bitte eingeben !

Gegentastprüfung — Das Attribut definiert ein Prüffeld. Ist der Prüfmodus (Gegentastprüfen) eingeschaltet, fordert FORMANT nur Prüffelder zur Eingabe an und prüft den Inhalt gegen einen im Anwendungsgrogramm vorgegebenen Inhalt. Siehe dazu den Aufruf 'modify operation mode' und Abschnitt 2.3.6.

Korrektursperre — Das Attribut sperrt die Korrekturmöglichkeit für Prüffelder. Die Tasten [SHIFT] [F16] (KOR) und [SHIFT] [F15] (FKOR) sind dann wirkungslos (siehe Abschnitt 2.3.6). Das Feld muß das Attribut Gegentastprüfung haben.

Userfeld — Das Attribut ist nicht implementiert. Die Angabe ist wirkungslos. Vorgesehen ist, daß man bei Userfeldern eigene Prüfroutinen zum Prüfen während der Eingabe vorsehen kann.

Bitte das Feldfüllzeichen eingeben !

alle abdr. Zeichen

Mit dem angegebenen Zeichen füllt FORMANT den Rest des Feldes am Bildschirm, wenn es nur teilweise oder gar nicht ausgefüllt wurde. Beim Aufbereiten richtet FORMANT das Feld erst wie gewünscht aus und füllt dann mit Füllzeichen. Das gilt auch für übersprungene Skipfelder.

Bitte das Datenfüllzeichen eingeben !

alle abdr. Zeichen

Mit dem angegebenen Zeichen füllt FORMANT den Rest des Feldes im Datensatz, wenn es nur teilweise oder gar nicht ausgefüllt wurde. Das Datenfüllzeichen steht also an den Stellen im Datensatz, wo am Bildschirm das Feldfüllzeichen steht.
Übersprungene Skipfelder enthalten das Zeichen X'7F'.

Den Zusammenhang mit den Angaben bei den Aufrufen

insert field	(N-INF bzw. n_inf)
get field attribute	(N-GFA bzw. n_gfa)
change field attribute	(N-CFA bzw. n_cfa)

finden Sie in folgenden Abschnitten.

3.4.1 Attribute bei FORMANTGEN und FORMANT-COBOL-Aufrufen

Attribut	Parameter	Wert
Feldname	N-NAME	2-8 Zeichen
Feldtyp		
Variabel	N-F-TYP	VAR
Fix	N-F-TYP	FIX
Feldinhalt		
Leer	N-F-VORB	NEIN
Vorbelegt	N-F-VORB	JA
Mussfeld	MUSS	JA
Ausweislesefeld	AUSWEIS	JA
Skipfeld	SKIP	JA
Vollständigkeitsfeld	VOLLSTAENDIG	JA

Duplizierfeld	DUPLIZIER	JA
Geschütztes Feld	GESCHUETZT	JA
Triggerfeld	TRIGGER	JA
Zeichenvorrat		
Alle Zeichen	F-ATTR	SONDERZ
Buchstaben	F-ATTR	ALPHA
Alphanumerisch	F-ATTR	ALPHANUM
Numerisch	F-ATTR	NUM
Feldausrichtung		
Linksbündig	LINKS	JA
Rechtsbündig	RECHTS	JA
Nachkommastellen		
0-15	DEZIMALST	0-15
Vorzeichen	VORZ	JA
Großschreibung	GROSS	JA
Gegentastprüfung	GEGENTAST	JA
Korrektursperre	KORRSPERRE	JA
Userfeld	nicht impl.	
Feldfüllzeichen		
alle abdr. Zeichen	AUSG-FZ	alle abdr. Zeichen
Datenfüllzeichen		
alle abdr. Zeichen	EING-FZ	alle abdr. Zeichen

3.4.2 Attribute bei FORMANTGEN und FORMANT-C-Aufrufen

Attribut	Parameter	Wert	Kategorie
Feldname	fieldname	2-8 Zeichen	
Feldtyp			
Variabel	type	VAR	FELDTYP
Fix	type	FIX	FELDTYP
Feldinhalt			
Leer	type	VAR+LEER	FELDTYP
Vorbelegt	type	VAR+VORB	FELDTYP
Mussfeld	at3	MUSS	BEARBEITUNG
Ausweislesefeld	at3	AUSWEISLESE	BEARBEITUNG
Skipfeld	at3	SKIP	BEARBEITUNG
Vollständigkeitsfeld	at3	VOLLSTAENDIG	BEARBEITUNG
Duplizierfeld	at3	DUPLIZIER	BEARBEITUNG
Geschütztes Feld	at3	GESCHUETZT	BEARBEITUNG
Triggerfeld	at3	TRIGGER	BEARBEITUNG
Zeichenvorrat			
Alle Zeichen	at4	SONDERZ	FELDATT
Buchstaben	at4	ALPHA	FELDATT
Alphanumerisch	at4	ALPHANUM	FELDATT

Numerisch	at4	NUM	FELDATT
Feldausrichtung			
Linksbündig	at4	LINKS	FELDATT
Rechtsbündig	at4	RECHTS	FELDATT
Nachkommastellen			
0–15	at5	0–15	NACHKOMMA
Vorzeichen	at4	VORZ	FELDATT
Großschreibung	at3	GROSS	BEARBEITUNG
Gegentastprüfung	at5	KGEGENTAST	NACHKOMMA
Korrektursperre	at5	KORRSPERRE	NACHKOMMA
Userfeld	nicht impl.		
Feldfüllzeichen			
alle abdr. Zeichen	at6	alle abdr. Z.	FUELLFELD
Datenfüllzeichen			
alle abdr. Zeichen	at7	alle abdr. Z.	FUELLDATEN

4 FORMANT in Beispielen

Dieses Kapitel soll Ihnen zeigen, wie man Anwendungsprogramme mit FORMANT schreibt. Dabei gehen wir vom 'Kleinstmöglichen' aus und werden dann weitere FORMANT-Funktionen kennenlernen.

Das Kapitel ist so aufgebaut, daß Sie einzelne Abschnitte auch überschlagen können, wenn Sie die Funktionen für Ihre Anwendung nicht benötigen. Sie sollen hier ein Gerüst vorfinden, das Ihnen wichtige Funktionen von FORMANT vorstellt und Sie mit FORMANT vertraut macht. Es wird Ihnen dann nicht schwer fallen, alle weiteren Funktionen anhand der Beschreibung in Kapitel 5 und 6 zu anzuwenden. In diesen Kapiteln sind alle Aufrufe nochmals mit einem kleinen Beispiel versehen.

Alle Beispiele sind erst vom Ablauf her erklärt und dann in COBOL und in C ausgeführt. Die meist verwendete Maske wurde als Beispiel in Kapitel 3 erstellt.

4.1 Das Kleinstmögliche - Daten erfassen

Was soll erreicht werden?

- Eine mit FORMANTGEN erstellte Maske ist am Bildschirm auszugeben.
- Wenn der Benutzer die Daten fertig eingegeben hat, soll eine Meldung erscheinen.
- Wird die Systemsteuertaste F14 (SEND) gedrückt, soll der Datensatz geschrieben werden. Die variablen Felder der Maske sind für eine neue Eingabe zu löschen.
- Wird die Taste END gedrückt, soll das Programm beendet werden.

Das ist nicht viel, liefert aber den Rahmen für jede FORMANT-Anwendung.

4.1.1 Struktur eines FORMANT-Anwendungsprogramms

Die einfachste Struktur eines FORMANT-Anwendungsprogramms ist:

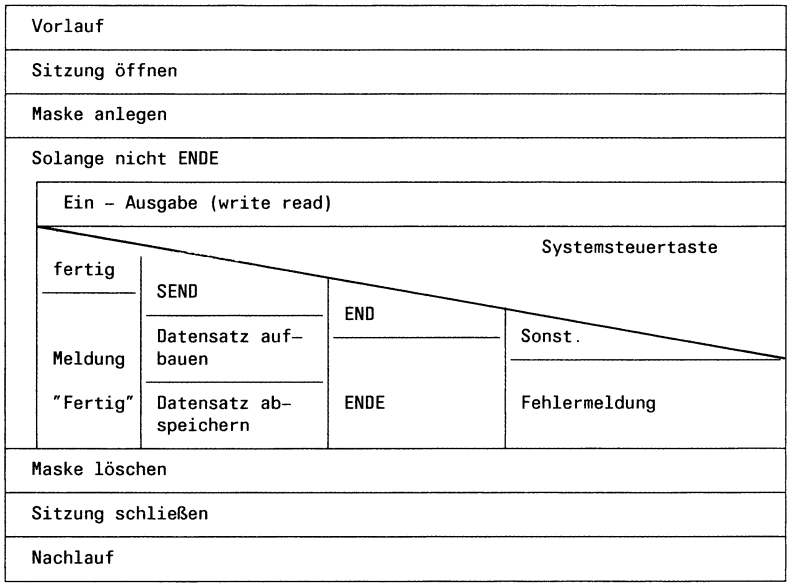


Bild 4-1: Struktogramm für Beispiel 1.

Die Aufrufe bewirken:

open session (N-OPS bzw. n_ops)
: liefert den Sitzungs-Deskriptor, eine Nummer, die FORMANT der Sitzung zuordnet und die bei jedem weiteren Aufruf anzugeben ist.

create presentation image (N-CPI bzw. n_cpi)
: holt die gewünschte Maske aus der Datei, die FORMANTGEN beim Generieren erzeugt hat. Dies ist nun die aktuelle Maske.

write read (N-WIR bzw. n_wir)
: gibt die aktuelle Maske am Bildschirm aus und verlangt die Eingabe von Daten in die Maske. Die Dateneingabe ist in Kapitel 2 beschrieben. Ist die Dateneingabe beendet, gibt der Aufruf die Steuerung an das Programm zurück.

change status message (N-CSM bzw. n_csm)
: gibt eine Statusmeldung in der letzen Bildschirmzeile aus.

change error message (N-CEM bzw. n_csm)
: gibt eine Fehlermeldung aus, in diesem Beispiel, wenn eine nicht zugelassene Systemsteuertaste gedrückt wurde.

set cursor (N-SEC bzw. n_sec)
: positioniert die Schreibmarke. In diesem Beispiel wird sie auf das aktuelle Feld gesetzt, damit die Schreibmarke nicht in das nächste Feld geht, wenn eine nicht zugelassene Taste gedrückt wurde.

get record (N-GER bzw. n_ger)
: baut einen Datensatz aus den variablen Feldern der Maske auf. Die Reihenfolge der Felder haben Sie beim Erstellen der Maske bestimmt.

clear record (N-CLR bzw. n_clr)
: löscht alle variablen Felder der Maske.

destroy presentation image (N-DPI bzw. n_dpi)
: schließt die aktuelle Maske. FORMANT löscht den Bildschirm und gibt den Speicherplatz für die aktuelle Maske frei.

close session (N-CLS bzw. n_cls)
: schließt die FORMANT-Sitzung. Danach ist kein FORMANT-Aufruf mehr möglich. Von FORMANT belegter Speicherplatz wird freigegeben. Dies muß immer der letzte FORMANT-Aufruf sein.

Das Beispiel enthält auch noch das Umschalten des Bildschirms in den 25-Zeilen-Modus und zurück, sowie eine Meldung zum Programmanfang und zum Programmende.

Im Struktogramm nicht berücksichtigt sind die Fehlerabfragen nach den FORMANT-Aufrufen. Solche Abfragen sollten Sie einbauen, in welcher Form auch immer. Sie erleichtern sich damit die Fehlersuche wesentlich.

Noch ein kleiner Hinweis: Falls in der Maske kein Eingabefeld ist, dann hält der Aufruf 'write read' nicht an, um Daten entgegenzunehmen. Die Steuerung geht gleich wieder an die Anwendung und ihr Programm läuft in einer endlosen Schleife. Es muß also mindestens ein ungeschütztes variables Feld vorhanden sein.

4.1.2 COBOL-Beispiel - Daten erfassen

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.              beispiel1.
*
*
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

*****************************************************************
*  Datei zum Abspeichern der Daten                              *
*****************************************************************
*
 select adresse assign "p-datei"
             organization is line sequential.

*****************************************************************
*  Datei fuer Fehlerprotokoll                                   *
*****************************************************************
*
 select fehl assign "f-datei"
             organization is line sequential.

*
 DATA DIVISION.
 FILE SECTION.

 FD adresse.
    01 d-daten              pic x(80).

*****************************************************************
*  Datensatz fuer Fehlerprotokoll                               *
*****************************************************************
*
 FD fehl.
 01 f-info.
    05 f-meld              pic x(19).
    05 f-aufruf            pic x(5).
    05 filler              pic x.
    05 f-code              pic -99999.
*
 WORKING-STORAGE SECTION.

*****************************************************************
*  Die COPY-Elemente enthalten die Datenbereiche fuer FORMANT:  *
*    FORGP  Globale Parameter                                   *
*    FORSK  FORMANT-Steuertasten                                *
*    FORER  FORMANT-Rueckmeldungen                              *
*    FORVA  FORMANT-Parameterwerte                              *
*****************************************************************

  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORSK.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".
```

```
  77 fehlerart                    pic x(8).
  77 ende                         pic xx        value "N".
  77 offset                       pic 99        value zero.

  01 init-par.
     05 init-text                 pic x(27)
        value "Programm beispiel1 geladen.".
     05 25-zeilen-modus           pic xxxx   value x"1B5B3075".

  01 end-par.
     05 loeschen                  pic xxxx   value x"1B5B324A".
     05 24-zeilen-modus           pic xxxx   value x"1B5B3175".
     05 end-text                  pic x(27)
        value "Programm beispiel1 beendet.".
*
 LINKAGE SECTION.
*
 PROCEDURE DIVISION.

*****************************************************************
*  Steuerteil                                                   *
*  Alle noetigen FORMANT Funktionen laufen nacheinander ab      *
*****************************************************************
*
 ablauf section.

    display init-par upon console.
    open extend adresse.
    perform sitzung-oeffnen.
    perform maske-laden.
    perform einaus-maske until ende = "J".
    perform maske-loeschen.
    perform sitzung-schliessen.
    close adresse.
    display end-par upon console.

    STOP RUN.

*****************************************************************
*  Sitzung öffnen                                               *
*  Den Sitzungs-Deskriptor liefert N-OPS im Feld                *
*  N-SESSION-ID in N-MAIN-PAR.                                  *
*  Er ist nun fuer alle anderen Aufrufe verfuegbar.             *
*****************************************************************
*
 sitzung-oeffnen section.

     move "N-OPS" to f-aufruf.
     call f-aufruf using n-main-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.

     if n-ret-code not = n-noerr
                  perform fehler.

     cancel f-aufruf.
```

```
*****************************************************************
*  Maske anlegen.                                               *
*  Der Aufruf laedt die in der Datei maskep vorhandene Maske.   *
*****************************************************************
*
 maske-laden section.

     move "maskep" to n-pi-name.
     move vorhanden to n-pi-mode.
     move "N-CPI" to f-aufruf.
     call f-aufruf using n-main-par, n-pi-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

*****************************************************************
*  Maske ausgeben und Daten einlesen.                           *
*  N-WIR fordert die Dateneingabe an.                           *
*****************************************************************
*
 einaus-maske section.

     move "N-WIR" to f-aufruf.
     call f-aufruf using n-main-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

*    *** Statusmeldung löschen ***
*
     move space to n-status-message.
     perform melde.

*    *** Abfrage der Systemsteuertaste. ***
*
     if  n-ret-sys = n-ausg
                 or n-next
                 or n-forwards
                 or n-feminus
                 or n-feplus

*    *** Status-Meldung, wenn fertig ausgefüllt ***
*
                 move "Fertig. Abspeichern mit F14"
                      to n-status-message

                 perform melde
*    *** Satz schreiben, wenn F14 (SEND) gedrueckt ***
*
     else if n-ret-sys = n-send
                 perform satz-schreiben
```

```
*    *** Programmende, wenn END gedrueckt ***
*
     else if n-ret-sys = n-end
                  move "J" to ende

*    *** Fehlermeldung, wenn falsche Taste gedrueckt wurde. ***
*
     else  move "Taste nicht definiert" to n-error-message
                  perform f-melde,

*    *** Schreibmarke positionieren auf aktuelles Feld. ***
*
                  move n-akt-feld to n-name,
                  perform schreibmarke-pos.


*****************************************************************
*  Maske schließen.                                             *
*  N-DPI schließt die aktuelle Maske und gibt den               *
*  Speicherplatz dafür frei.                                    *
*****************************************************************
*
 maske-loeschen section.

     move "N-DPI" to f-aufruf.
     call f-aufruf using n-main-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

*****************************************************************
*  Sitzung schließen.                                           *
*  Dies ist der letzte FORMANT-Aufruf.                          *
*****************************************************************
*
 sitzung-schliessen section.
     move "N-CLS" to f-aufruf.
     call f-aufruf using n-main-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.
*****************************************************************
*  Datensatz aufbauen und schreiben.                            *
*  Der Aufruf get record baut in d-daten einen Datensatz auf,   *
*  wie er in der Maske definiert ist.                           *
*  clear record loescht die variablen Felder der Maske fuer     *
*  eine neue Eingabe.                                           *
*****************************************************************
*
 satz-schreiben section.

*    *** Datensatz aufbauen. ***
*
```

```
     move 80 to n-ndsl.
     move spaces to d-daten.
     move "N-GER" to f-aufruf.
     call f-aufruf using n-main-par, d-daten
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

*    *** Datensatz schreiben. ***
*
     write d-daten.
*    *** Variable Felder der Maske loeschen. ***
*
     move "N-CLR" to f-aufruf.

     call f-aufruf using n-main-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.

     cancel f-aufruf.


*********************************************************************
*  Statusmeldung ausgeben.                                          *
*  Der Aufruf change status message gibt in der letzten Bild-       *
*  schirmzeile eine Meldung aus (Datenfeld n-status-message).       *
*********************************************************************
*
 melde section.

 move "N-CSM" to f-aufruf.
 call f-aufruf using n-main-par, n-pi-par
              on overflow
              move "OVERFLOW" to fehlerart
              perform fehler.
 if n-ret-code not = n-noerr
              perform fehler.
 cancel f-aufruf.


*********************************************************************
*  Schreibmarke positionieren.                                      *
*  Der Feldname steht in n-name, offset ist der Abstand zum         *
*  Feldanfang.                                                      *
*********************************************************************
*
 schreibmarke-pos section.

 move "N-SEC" to f-aufruf.
 call f-aufruf using n-main-par, n-field-par, offset
              on overflow
              move "OVERFLOW" to fehlerart
              perform fehler.
 if n-ret-code not = n-noerr
```

```
                perform fehler.
 cancel f-aufruf.

*****************************************************************
*  Fehlermeldung ausgeben.                                      *
*  Der Aufruf change error message gibt in der letzten Bild-    *
*  schirmzeile eine Meldung aus (Datenfeld n-error-message).    *
*****************************************************************
*
 f-melde section.

 move "N-CEM" to f-aufruf.
 call f-aufruf using n-main-par, n-pi-par
                on overflow
                move "OVERFLOW" to fehlerart
                perform fehler.
 if n-ret-code not = n-noerr
                perform fehler.
 cancel f-aufruf.


*****************************************************************
*  Fehler-Routine.                                              *
*  Die Routine wird aufgerufen, wenn ein FORMANT-Aufruf wegen   *
*  Speichermangel nicht ausgefuehrt werden kann (overflow)      *
*  oder mit einem Fehlercode ungleich N-NOERR zurueckkehrt.     *
*  Aufruf und Fehlercode werden in die Datei f-datei geschrieben.*
*****************************************************************
*
 fehler section.

     move n-ret-code to f-code.

     if fehlerart = "OVERFLOW"
                move "OVFLOW bei Aufruf: " to f-meld
     else
                move "Fehler bei Aufruf: " to f-meld.

*    *** Information auf die Fehlerdatei f-datei schreiben. ***
*
     open extend fehl.
     write f-info.
     close fehl.
*    *** Programmabbruch bei Fehlercode < 0 oder overflow. ***
*
     if     n-ret-code < 0
     or     fehlerart = "OVERFLOW"

            stop run.

*****************************************************************
*    Ende gut, alles gut.                                       *
*****************************************************************
```

4.1.3 C-Beispiel - Daten erfassen

```
/*************************************************************************
   Das Programm 'beispiel1' realisiert das oben gezeigte Struktogramm.
   Fehler werden auf stderr geschrieben (aufrufen z.B mit
   beispiel1 2> errdatei)
*************************************************************************/

#include <formant.h>     /* Definitionen für FORMANT */
#include <stdio.h>
#define SATZLAENGE 80    /* max. Satzlänge = Länge var. Daten in der Maske */
int sd;                  /* Sitzungs-Deskriptor */

main (argc, argv)
int argc;
char *argv[];   {

int version = 100;       /* FORMANT-Version 1.00 */
int ret;                 /* Rueckgabewert */
int t_code;              /* logischer Code der Systemsteuertaste */
char akfeld[9];          /* aktuelles Feld */
char satz[SATZLAENGE+1];/* Datensatz */
int s_laenge;            /* Laenge der uebergebenen Daten */
FILE *a_datei;           /* Ausgabedatei */

                                        /* Ausgabedatei oeffnen */
if ( (a_datei = fopen("p-datei","a")) == NULL ) {
        fprintf (stderr,"%s","Datei p-datei kann nicht geoeffnet werden");
        exit (300);
        }

                                        /* Umschalten auf 25-Zeilen-Modus */
printf ("Programm %s geladen\33[0u",argv[0]);

ret = n_ops (version,&sd);              /* Sitzung öffnen */
teste ("n_ops",ret);
                                        /* Maske anlegen */
ret = n_cpi (sd,"maskep",VORHANDEN);
teste("n_cpi",ret);

do      {                               /* Leseschleife */

        ret = n_wir (sd,&t_code,akfeld);/* Dateneingabe */
        teste ("n_wir",ret);
        ret = n_csm (sd," ");           /* Statusmeldung löschen */
        teste ("n_csm",ret);

        switch (t_code) {
            case FORWARDS:
            case NEXT    :
            case FEPLUS  :
            case FEMINUS :
            case AUSG    :      {       /* wenn fertig ausgefuellt, */
                                        /* Statusmeldung  */
                ret = n_csm (sd,"Fertig. Abspeichern mit F14.");
                teste ("n_csm",ret);
                break;
                }

            case SEND    :      {

                                        /* Datensatz aufbauen */
                ret = n_ger (sd,satz,&s_laenge);
                teste ("n_ger",ret);
```

```
                                        /* Datensatz schreiben */
                fprintf(a_datei,"%s\n",satz);

                ret = n_clr (sd);       /* Datenfelder löschen */
                teste ("n_clr",ret);
                break;
                }

            case END     : break;

            default      :      {       /* Undefinierte Taste: */
                                        /* Fehlermeldung */

                ret = n_cem (sd,"Taste nicht definiert");
                teste ("n_cem",ret);

                                        /* Schreibmarke positionieren */
                                        /* auf aktuelles Feld */
                ret = n_sec (sd,akfeld,0);
                teste ("n_sec",ret);

                break;
                }
            }

        }  while (t_code != END);

ret = n_dpi (sd);                       /* Maske schließen */
teste ("n_dpi",ret);

ret = n_cls (sd);                       /* Sitzung schließen */
teste ("n_cls",ret);

                                        /* Umschalten auf 24-Zeilen-Modus und
                                           Bildschirm löschen. */
printf("\33[2J\33[1uProgramm %s beendet\n",argv[0]);

exit (0);

}
/*********************************************************************
   Die Funktion 'teste' prüft den Rueckgabewert auf NOERR.
   Ist der Rueckgabewert =0(NOERR), wird das Programm fortgesetzt.

   Sonst gibt die Funktion eine Meldung aus und reagiert wie folgt:
   Ist der Rueckgabewert <0, bricht teste das Programm ab.
   Ist der Rueckgabewert >0, wird das Programm fortgesetzt.
*********************************************************************/
teste (aufruf,r_wert)
char *aufruf;
int r_wert;     {

if (r_wert == NOERR)
        return;

else    {
        fprintf(stderr,"Fehler bei Aufruf %s. Fehlercode %d\n",aufruf,r_wert);

        if (r_wert > NOERR)
                return;
        else    {
                n_dpi (sd);
                n_cls (sd);
```

```
                                        /* Umschalten auf 24-Zeilen-Modus und
                                           Bildschirm löschen. */
                printf("\33[2J\33[1u\n");
                exit(r_wert);
                }
        }
}
```

4.2 Masken wechseln - Menütechnik

Nicht immer hat man Masken, in die man Daten eingeben will. In einer Auswahlmaske soll z.B. nur ein Buchstabe eingegeben werden oder eine Funktionstaste gedrückt werden. Abhängig davon soll dann eine andere Maske erscheinen (wie z.B. bei FORMANTGEN).

Dabei ist folgendes zu beachten:

1. Die Auswahlmaske muß mindestens ein variables Feld haben, damit der Leseaufruf anhält und eine Eingabe anfordert. Ein Feld mit 1 Zeichen Länge genügt. Darauf positioniert FORMANT die Schreibmarke. Sie können das Feld dunkel setzen; es darf aber nicht geschützt sein.

   Für eine Auswahl unter mehreren Möglichkeiten könnten Sie auch die Position der Schreibmarke verwenden. Dann müssen Sie entsprechend viele Felder definieren und können mit 'get cursor' den Feldnamen abfragen.

2. Nach Durchlaufen des Leseaufrufs fragen Sie die Eingabe ab. Wenn Sie Funktionstasten verwenden, dann kann eine besondere Situation auftreten:

   FORMANT übergibt standardmäßig die Steuerung an das Anwendungsprogramm, wenn eine Funktionstaste gedrückt wurde. Beim nächsten Leseaufruf kehrt FORMANT wieder zur Eingabe zurück. Ist nun in der Maske nur ein einziges Feld vorhanden, kehrt FORMANT ebenfalls nochmals zur Eingabe zurück und stellt fest, daß dies das letzte Feld war und die Eingabe beendet ist. Das kann in einigen Fällen Ihre Programmlogik durcheinanderbringen, weil der Leseaufruf dabei einmal durchlaufen wird, ohne daß man etwas eingeben kann.

   Vermeiden können Sie diesen Effekt dadurch, daß Sie als nächstes Eingabefeld nochmals das aktuelle Feld vereinbaren. Das geht mit dem Aufruf 'set cursor'.

3. Zum Maskenwechsel müssen Sie

   — erst die aktuelle Maske schließen,
   — dann die neue Maske anlegen.

   Es ist gleichgültig, in welchem Programmteil (SECTION bzw. Funktion) Sie die Maske schließen. Es ist aber günstig, immer denselben Ablauf zu wählen: z.B. immer der aufrufende Programmteil schließt seine Maske und legt sie dann gegebenenfalls wieder an. Oder der aufrufende Teil wechselt immer die Maske.

   Wenn Sie die Reihenfolge 'anlegen - schließen' nicht einhalten, bringt FORMANT als Ergebniswert 'Falsche Reihenfolge'. Das gilt auch, wenn Sie FORMANT-Funktionen aufrufen, die eine aktuelle Maske voraussetzen, ohne daß eine Maske angelegt ist. Z.B. funktioniert der Aufruf 'write read' nicht, wenn Sie noch keinen 'create p.i.' aufgerufen haben.

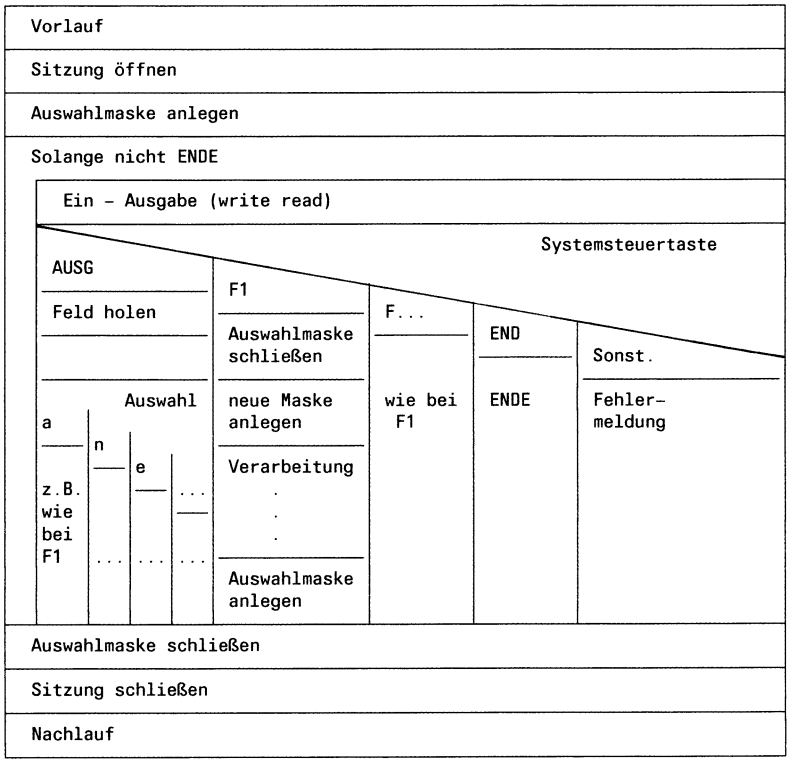


Bild 4-2: Struktogramm Menüauswahl

4.2.1 COBOL-Beispiel - Menüauswahl

Die im Beispiel ausgelassenen Programmteile (z.B. sitzung-oeffnen section, fehler section usw.) sind identisch mit denen im Beispiel in Abschnitt 4.1.2. Die Auswahlmaske 'maskew' muß ein einstelliges variables Feld enthalten.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.                bsp-menue.
*
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
         CONSOLE IS CRT.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
    .
    .
    .
```

```
*
 DATA DIVISION.
 FILE SECTION.
    .
    .
    .
*
 WORKING-STORAGE SECTION.

*****************************************************************
*  Die COPY-Elemente enthalten die Datenbereiche fuer FORMANT:  *
*    FORGP  Globale Parameter                                   *
*    FORFP  FORMANT-Feldparameter                               *
*    FORSK  FORMANT-Steuertasten                                *
*    FORER  FORMANT-Rueckmeldungen                              *
*    FORVA  FORMANT-Parameterwerte                              *
*****************************************************************

  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORSK.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".

  77 fehlerart                  pic x(8).
  77 ende                       pic xx       value "N".
  77 wahl                       pic x.
  77 offset                     pic 99       value zero.
    .
    .
    .
*
 LINKAGE SECTION.
*
 PROCEDURE DIVISION.

*****************************************************************
*  Steuerteil. Das Programm kehrt immer zur Auswahlmaske        *
*  zurück, Ende bei Taste END in der Auswahlmaske.              *
*****************************************************************
*
 ablauf section.
    .
    .
    perform sitzung-oeffnen.

    move "maskew" to n-pi-name.
    perform maske-laden.

    perform auswahl until ende = "J".

    perform maske-loeschen.
    perform sitzung-schliessen.
    .
    .

    STOP RUN.

    .
    .
```

```
*****************************************************************
*  Maske anlegen.                                               *
*  Der Aufruf laedt eine vorhandene Maske.                      *
*****************************************************************
*
 maske-laden section.

     move vorhanden to n-pi-mode.
     move "N-CPI" to f-aufruf.
     call f-aufruf using n-main-par, n-pi-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

*****************************************************************
*  Funktionsauswahl mit Menue.                                  *
*  Abhängig von einer Buchstabentaste (a, l oder n) oder        *
*  einer Funktionstaste (F1 - F5) wird eine Unterfunktion       *
*  des Programms mit einer anderen Maske ausgeführt.           *
*****************************************************************
*
 auswahl section.

*    *** Ein-Ausgabe der Auswahlmaske  ***
*
     move "N-WIR" to f-aufruf.
     call f-aufruf using n-main-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

*    *** Statusmeldung löschen ***
*
     move space to n-status-message.
     perform melde.
*    *** Abfrage der Systemsteuertaste, ***
*        bei n-ausg wurde eine Buchstabentaste gedrueckt.
*
     if  n-ret-sys = n-ausg
         perform buchstaben-wahl,

*        bei F1 Maske wechseln und
*        Verarbeitung-1 aufrufen.
*
     else if  n-ret-sys = N-F1
         perform maske-loeschen,
         move "maske1" to n-pi-name,
         perform maske-laden,

         perform verarbeitung-1 until ende = "J",
         move "N" to ende,
```

```
          perform maske-loeschen,
          move "maskew" to n-pi-name,
          perform maske-laden

*         bei F2 Maske wechseln und
*         Verarbeitung-2 aufrufen.
*
     else if  n-ret-sys = N-F2
          perform maske-loeschen,
          move "maske2" to n-pi-name,
*              .
*              .
*              .

*         bei END Programmende.
*
     else if  n-ret-sys = n-end

          move "J" to ende

*         Bei nicht verwendeten Tasten Meldung ausgeben.
*         Die Schreibmarke wird ins aktuelle Feld gesetzt.
*         Dadurch bricht die Eingabe nicht ab.

     else
          move "Taste nicht zugelassen" to n-status-message,
          perform melde,
          move n-akt-feld to n-name,
          perform schreibmarke-pos.

*****************************************************************
*  Funktionsauswahl mit Buchstaben a, n, l.                     *
*****************************************************************
*
 buchstaben-wahl section.
     move n-akt-feld to n-name.
     move 1 to n-laenge.
     move "N-GEF" to f-aufruf.
     call f-aufruf using n-main-par, n-field-par, wahl
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

     if        wahl = "a"
                    .
                    .   wie oben
                    .
     else if   wahl = "n"
                    .
                    .   wie oben
                    .
     else if   wahl = "l"
                    .
                    .   wie oben
                    .
     else          move "Auswahl nicht zugelassen." to
                   n-status-message,    perform melde.
```

```
*****************************************************************
*  Verarbeitung-1.                                              *
*  Maske ausgeben und Daten einlesen.                           *
*  N-WIR fordert die Dateneingabe an.                           *
*****************************************************************
*
 verarbeitung-1 section.

     move "N-WIR" to f-aufruf.
     call f-aufruf using n-main-par, n-pi-par
   .
   .
   .
```

4.2.2 C-Beispiel - Menüauswahl

```
/*****************************************************************
   Das Programm 'bsp_menue.c' realisiert eine Menueauswahl.

   Die Verarbeitungsfunktionen wurden hier nicht ausgeführt.
   In der Auswahlmaske muß ein einstelliges variables Feld sein.

   Die Funktion teste ist wie im Beispiel in Abschnitt 4.1.
*****************************************************************/

#include <formant.h>    /* Definitionen für FORMANT */
#include <stdio.h>

int version = 100;      /* FORMANT-Version 1.00 */
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Rueckgabewert */
int t_code;             /* logischer Code der Systemsteuertaste */
char akfeld[9];         /* aktuelles Feld */
char wahl[2];           /* Auswahlfeld */
main (argc, argv)
int argc;
char *argv[];   {

                                    /* Umschalten auf 25-Zeilen-Modus */
printf ("Programm %s geladen\33[0u",argv[0]);


ret = n_ops (version,&sd);          /* Sitzung oeffnen */
teste ("n_ops",ret);

                                    /* Auswahl-Maske laden */
ret = n_cpi (sd,"maskew",VORHANDEN);
teste("n_cpi",ret);

while ( auswahl () != END )
      ;
ret = n_dpi (sd);                   /* Maske schliessen */
teste ("n_dpi",ret);

ret = n_cls (sd);                   /* Sitzung schliessen */
teste ("n_cls",ret);
```

```
                                        /* Umschalten auf 24-Zeilen-Modus und
                                           Bildschirm löschen. */
printf("\33[2J\33[1uProgramm %s beendet\n",argv[0]);

exit (0);

}

auswahl ()  {
ret = n_wir (sd,&t_code,akfeld);        /* Ein- Ausgabe */
teste ("n_wir",ret);
ret = n_csm(sd," ");
teste ("n_csm",ret);

switch (t_code) {

        case AUSG:                              /* Buchstabentaste ? */

                                                /* Auswahlfeld holen */
                ret = n_gef(sd,akfeld,wahl);
                teste ("n_gef",ret);

                switch (wahl)  {

                        case 'a':  ...  ; break;    /* ...  Aufruf einer  */
                                                    /* Unterfunktion      */
                        case 'l':  ...  ; break;    /* ebenso wie bei F1  */
                                                    /* weiter unten       */
                        case 'n':  ...  ; break;    /* gezeigt.           */

                        default :  ret=n_cem(sd,"Auswahl nicht zugelassen.");
                                   teste ("n_cem",ret);
                                   break;
                        }
                break;

        case F1:                        /* Bei F1 Verarb. 1         */
                ret = n_dpi (sd);       /* Maske schliessen */
                teste ("n_dpi",ret);
                v_maske1 ();
                ret = n_cpi (sd,"maskew",VORHANDEN);
                teste("n_cpi",ret);
                break;

        case F2: ...
                .
                .
                .       weitere Unterfunktionen
                .

        case F5:
                ret = n_csm (sd,"Funktion nicht implementiert.");
                teste ("n_csm",ret);
                ret = n_sec (sd,akfeld,0);
                teste ("n_sec",ret);
                break;

        case END: return (END);

        default:                        /* Undefinierte Systemsteuertaste */

                n_cem (sd,"Taste nicht zugelassen.");
                teste ("n_cem",ret);
```

```
                ret = n_sec (sd,akfeld,0);
                teste ("n_sec",ret);

    }

}


v_maske1 ()   {
                                        /* Maske laden */
ret = n_cpi (sd,"maske1",VORHANDEN);
teste("n_cpi",ret);
    .
    .
    .
    .    beliebige  Verarbeitung
    .
    .
ret = n_dpi (sd);                       /* Maske schliessen */
teste ("n_dpi",ret);

}

teste (aufruf,r_wert)
char *aufruf;
int r_wert;     {

    .
    .    wie im Beispiel in Abschnitt 4.1
    .
}
```

4.3 Gegentastprüfen

Beim Gegentastprüfen müssen Daten, die bereits eingegeben wurden, ein zweites Mal eingetastet werden. Dabei gibt es zwei Möglichkeiten:

1. Die Daten werden sofort nach der Ersteingabe ein zweites Mal eingegeben. Dabei kann man feldweise oder satzweise prüfen, je nach eingestellter Eingabereaktion.

2. Bereits erfaßte Datensätze werden gelesen. Die zu prüfenden Felder jedes Satzes müssen ein zweites Mal eingegeben werden.

Mit FORMANT lassen sich beide Prüfverfahren realisieren. Der Ablauf ist im Prinzip gleich. Er geht nach folgendem Schema:

modify operation mode	Prüfmodus einschalten
change record	Vorgabesatz übergeben
read oder write read	Dateneingabe

Im Prüfmodus fordert FORMANT nur Prüffelder zur Eingabe an. Prüffelder definieren Sie mit FORMANTGEN (Funktion Attribute setzen). Der Ablauf der Eingabe ist in Abschnitt 2.3.6 beschrieben. FORMANT prüft die Eingabe gegen den Inhalt des Datensatzes, den Sie mit 'change record' an FORMANT übergeben haben. Felder, die keine Prüffelder sind, füllt FORMANT bei der Eingabe mit dem Inhalt des Vorgabesatzes.

Den Prüfmodus können Sie jederzeit ein- oder ausschalten. Beim ersten Verfahren (sofortiges Prüfen) schalten Sie jedesmal den Prüfmodus aus, um die nächste Eingabe anzufordern (Feld oder Satz). Um diese Eingabe zu prüfen, schalten Sie den Prüfmodus ein, übergeben den zu prüfenden Satz und machen erneut einen Leseaufruf. Die Aufruffolge ist dann:

modify operation mode	Prüfmodus ausschalten
read oder write read	Dateneingabe
get record	Satz übertragen
clear record	Satz beim Prüfen nicht anzeigen
modify operation mode	Prüfmodus einschalten
change record	Vorgabesatz übergeben (das ist der, den Sie mit 'get record' geholt haben)
read oder write read	Dateneingabe

Ein in C programmiertes Beispiel hierfür finden Sie beim Aufruf n_mom.

Anschließend finden Sie ein Beispiel, wie man bereits erfaßte Datensätze prüfen kann.

Hinweise:

- Normalerweise zeigt FORMANT den Datensatz an, den Sie mit 'change record' übergeben. Wollen Sie das nicht, müssen Sie 'clear record' aufrufen, bevor Sie den Prüfmodus einschalten.
- Auch bei feldweiser Eingabereaktion müssen Sie die Vorgabe mit 'change record' übergeben. Auch wenn nur ein Feld zu prüfen ist, genügt 'change field' nicht.
- Ein nicht korrigierbares Prüffeld kann man bei der Eingabe nur verlassen, wenn der originale Inhalt der Vorgabe eingegeben wurde oder aber mit einer Funktionstaste. Sie sollten in diesem Fall eine Funktionstaste für eine gezielte Reaktion verwenden und Funktiontasten sperren, die ein 'unerlaubtes Ausweichen' ermöglichen würden.

4.3.1 COBOL-Beispiel - Gegentastprüfen

Das folgende Programm zeigt, wie man Datensätze prüft, die bereits in einer Datei abgespeichert wurden.

Es wird dieselbe Maske verwendet wie beim Erfassen der Sätze. Welches Felder zu prüfen sind, legt man mit FORMANTGEN in der Maske fest.

Einige Programmteile sind identisch mit den entsprechenden in Abschnitt 4.1.2. Sie sind hier weggelassen.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.              bsp-gtast.
*
*
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
          CONSOLE IS CRT.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

*****************************************************************
*  Eingabedatei                                                 *
*****************************************************************
*
 select pruefdaten assign "p-datei"
             organization is line sequential.

*****************************************************************
*  Datei zum Abspeichern der Daten                              *
*****************************************************************
*
 select adresse assign "a-datei"
             organization is line sequential.

*****************************************************************
*  Datei fuer Fehlerprotokoll                                   *
*****************************************************************
*
 select fehl assign "f-datei"
             organization is line sequential.

*
 DATA DIVISION.
 FILE SECTION.

 FD pruefdaten.
    01 p-daten              pic x(80).

 FD adresse.
    01 a-daten              pic x(80).
```

```
*****************************************************************
*  Datensatz fuer Fehlerprotokoll                               *
*****************************************************************
*
 FD fehl.
 01 f-info.
    05 f-meld              pic x(19).
    05 f-aufruf            pic x(5).
    05 filler              pic x.
    05 f-code              pic -99999.
*
 WORKING-STORAGE SECTION.

*****************************************************************
*  Die COPY-Elemente enthalten die Datenbereiche fuer FORMANT:   *
*    FORGP  Globale Parameter                                    *
*    FORSK  FORMANT-Steuertasten                                 *
*    FORCP  FORMANT-Steuerparameter                              *
*    FORFP  FORMANT-Feldparameter                                *
*    FORER  FORMANT-Rueckmeldungen                               *
*    FORVA  FORMANT-Parameterwerte                               *
*****************************************************************

  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORSK.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORCP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".

  77 fehlerart                  pic x(8).
  77 ende                       pic xx        value "N".
  77 offset                     pic 99        value zero.

  01 init-par.
     05 init-text               pic x(27)
        value "Programm bsp-gtast geladen.".
     05 25-zeilen-modus         pic xxxx   value x"1B5B3075".

  01 end-par.
     05 loeschen                pic xxxx   value x"1B5B324A".
     05 24-zeilen-modus         pic xxxx   value x"1B5B3175".
     05 end-text                pic x(27)
        value "Programm bsp-gtast beendet.".

*
 LINKAGE SECTION.
*
 PROCEDURE DIVISION.

*****************************************************************
*  Steuerteil                                                   *
*  Alle noetigen FORMANT Funktionen laufen nacheinander ab      *
*****************************************************************
*
 ablauf section.

    display space.
    display init-par at 0505.
```

```
    open input pruefdaten.
    open extend adresse.
    perform sitzung-oeffnen.
    perform maske-laden.
    perform gegentast-ein.
    perform satz-lesen.
    perform einaus-maske until ende = "J".
    perform maske-loeschen.
    perform sitzung-schliessen.
    close adresse pruefdaten.
    display end-par at 0505.

    STOP RUN.

 sitzung-oeffnen section.
      .
      .
      .

 maske-laden section.
      .
      .
      .
*****************************************************************
*  Gegentastpruefen einschalten.                                *
*****************************************************************
*
 gegentast-ein section.

     move JA to n-gegentastp.
     move "N-MOM" to f-aufruf.
     call f-aufruf using n-main-par, n-op-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.


*****************************************************************
*  Datensatz mit Pruefdaten einlesen.                           *
*  Die Pruefdaten werden mit change record an FORMANT uebergeben.*
*****************************************************************
*
 satz-lesen section.

     read pruefdaten at end move "J" to ende.

     move 80 to n-ndsl.
     move "N-CHR" to f-aufruf.
     call f-aufruf using n-main-par, p-daten
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.
```

```
*****************************************************************
*  Maske ausgeben und Daten einlesen.                           *
*  N-WIR fordert die Dateneingabe an.                           *
*****************************************************************
*
 einaus-maske section.

     move "N-WIR" to f-aufruf.
     call f-aufruf using n-main-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

*    *** Statusmeldung löschen ***
*
     move space to n-status-message.
     perform melde.

*    *** Eingabe richtig abgeschlossen  ***
*
     if  n-ret-sys not > n-send

                 perform satz-schreiben,
                 perform satz-lesen

*    *** Verarbeitung bei F5                      ***
*        für nicht korriegierbare Prueffelder.
*
     else if n-ret-sys = n-f5
                 move "F5" to n-status-message,
                 perform melde
*    *** Fehlermeldung, wenn falsche Taste gedrueckt wurde. ***
*
     else if n-ret-sys not = n-end
                 move "Taste nicht erlaubt" to n-error-message,
                 perform f-melde

*    *** Schreibmarke positionieren auf aktuelles Feld. ***
*
                 move n-akt-feld to n-name,
                 perform schreibmarke-pos

*    *** Programmende bei END ***
*
     else        move "J" to ende.


 maske-loeschen section.
      .
      .

 sitzung-schliessen section.
      .
      .
```

```
*****************************************************************
*  Datensatz aufbauen und schreiben.                            *
*  Der Aufruf get record baut in a-daten einen Datensatz auf,   *
*  wie er in der Maske definiert ist.                           *
*  clear record loescht die variablen Felder der Maske fuer     *
*  eine neue Eingabe.                                           *
*****************************************************************
*
 satz-schreiben section.
*    *** Datensatz aufbauen. ***
*
     move 80 to n-ndsl.
     move spaces to a-daten.
     move "N-GER" to f-aufruf.
     call f-aufruf using n-main-par, a-daten
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

*    *** Datensatz schreiben. ***
*
     write a-daten.

*    *** Variable Felder der Maske loeschen. ***
*
     move "N-CLR" to f-aufruf.

     call f-aufruf using n-main-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.

     cancel f-aufruf.


 schreibmarke-pos section.
      .
      .
 melde section.
      .
      .
 f-melde section.
      .
      .
 fehler section.
      .
      .
*****************************************************************
*    Ende gut, alles gut.                                       *
*****************************************************************
```

4.3.2 C-Beispiel - Gegentastprüfen

Das folgende Programm zeigt, wie man Datensätze prüft, die bereits in einer Datei abgespeichert wurden.

Es wird dieselbe Maske verwendet wie beim Erfassen der Sätze. Welches Felder zu prüfen sind, legt man mit FORMANTGEN in der Maske fest.

```
/*********************************************************************
   Programmbeispiel für Gegentastprüfen von Sätzen, die aus einer
   DAtei gelesen werden.
   Fehler werden auf stderr geschrieben (aufrufen z.B mit
   beispiel1 2> errdatei).
*********************************************************************/

#include <formant.h>     /* Definitionen für FORMANT */
#include <stdio.h>
#define SATZLAENGE 80    /* max. Satzlänge = Länge var. Daten in der Maske */

int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Rueckgabewert */
FILE *e_datei;           /* Eingabedatei */

main (argc, argv)
int argc;
char *argv[];   {
int version = 100;       /* FORMANT-Version 1.00 */
int t_code;              /* logischer Code der Systemsteuertaste */
char akfeld[9];          /* aktuelles Feld */
char satz[SATZLAENGE+2];  /* Datensatz */
int s_laenge;            /* Länge des übergebenen Datensatzes */
FILE *a_datei;           /* Ausgabedatei */


                                        /* Ausgabedatei oeffnen */
if ( (a_datei = fopen("a-datei","a")) == NULL ) {
        fprintf (stderr,"%s","Datei a-datei kann nicht geoeffnet werden");
        exit (300);
        }

                                        /* Eingabedatei oeffnen */
if ( (e_datei = fopen("p-datei","r")) == NULL ) {
        fprintf (stderr,"%s","Datei p-datei kann nicht geoeffnet werden");
        exit (301);
        }

                                        /* Umschalten auf 25-Zeilen-Modus */
printf ("Programm %s geladen\33[0u",argv[0]);

ret = n_ops (version,&sd);              /* Sitzung oeffnen */
teste ("n_ops",ret);

                                        /* Maske laden */
ret = n_cpi (sd,"maskep",VORHANDEN);
teste ("n_cpi",ret);

                                        /* Gegentastprüfen einschalten */
ret = n_mom (sd,GEGENTASTP,EIN);
teste ("n_mom",ret);

t_code = lesen (satz);                  /* ersten Satz lesen */
```

```
while ( t_code != END ) {                    /* Leseschleife */

        ret = n_wir (sd,&t_code,akfeld);/* Ein- Ausgabe */
        teste ("n_wir",ret);

        ret = n_csm (sd," ");                /* Statusmeldung löschen */
        teste ("n_csm",ret);

                                             /* Eingabe richtig abgeschlossen */
        if (t_code <= SEND)     {

                                             /* Datensatz aufbauen */
                ret = n_ger (sd,satz,&s_laenge);
                teste ("n_ger",ret);

                                             /* Datensatz schreiben */
                fprintf(a_datei,"%s\n",satz);
                                             /* Felder löschen */
                ret = n_clr (sd);
                teste ("n_clr",ret);

                                             /* neuen Satz lesen */
                t_code = lesen (satz);
                }

        else if (t_code == F5)       {
                                             /* Verarbeitung bei F5 */
                    .
                    .       hier kann man z.B. eine Reaktion vorsehen
                    .       für nicht korrigierbare Prüffelder.
                    .
                }

        else if (t_code != END)      {
                ret = n_csm (sd,"Taste nicht erlaubt");
                teste ("n_csm",ret);

                ret = n_sec (sd,akfeld,0);
                teste ("n_sec",ret);
                }
        }

ret = n_dpi (sd);                            /* Maske schliessen */
teste ("n_dpi",ret);

ret = n_cls (sd);                            /* Sitzung schliessen */
teste ("n_cls",ret);

                                             /* Umschalten auf 24-Zeilen-Modus und
                                                Bildschirm löschen. */
printf("\33[2J\33[1uProgramm %s beendet\n",argv[0]);
exit (0);

}
```

```
/*********************************************************************
Die Funktion 'lesen' holt einen Satz aus p-datei und übergibt ihn als
Vorgabesatz.
*********************************************************************/
lesen (satz)
char *satz;
{

if ( fgets (satz,SATZLAENGE+2,e_datei) == NULL)  {
        return  (END);
        }

satz [SATZLAENGE] = '\0';        /* \n entfernen */

                                 /* Vorgabesatz übergeben */
ret = n_chr (sd,satz,SATZLAENGE);
teste ("n_chr",ret);

return (0);
}
```

Die Funktion 'teste' ist definiert wie in Abschnitt 4.1.3.

5 Anwendungsprogramme in COBOL

COBOL-Programme benutzen FORMANT in Form von CALL-Aufrufen. Für jede FORMANT-Funktion steht eine Datei mit Zwischencode zur Verfügung. Die Dateien sind im Dateiverzeichnis /usr/lib/cobol/formant/int.

Zum Ablauf verwendet FORMANT ein erweitertes COBOL-Laufzeitsystem rts2, das Sie nach der Installation von FORMANT nach /usr/lib/cobol bringen müssen (siehe Abschnitt 8.3). Ferner benutzt FORMANT das Programm termst, das neben dem Laufzeitsystem rts2 als eigener Prozess läuft.

Bei einem FORMANT-Aufruf läuft folgendes ab:

Ein in rts2 enthaltener Protokollumsetzer setzt den Aufruf und die übergebenen Parameter in ein sprachunabhängiges Protokoll um. Dieses Protokoll übergibt rts2 an das sprachneutrale FORMANT-Laufzeitsystem termst. Dieses führt die Funktion aus und übergibt das Ergebnis auf demselben Weg.

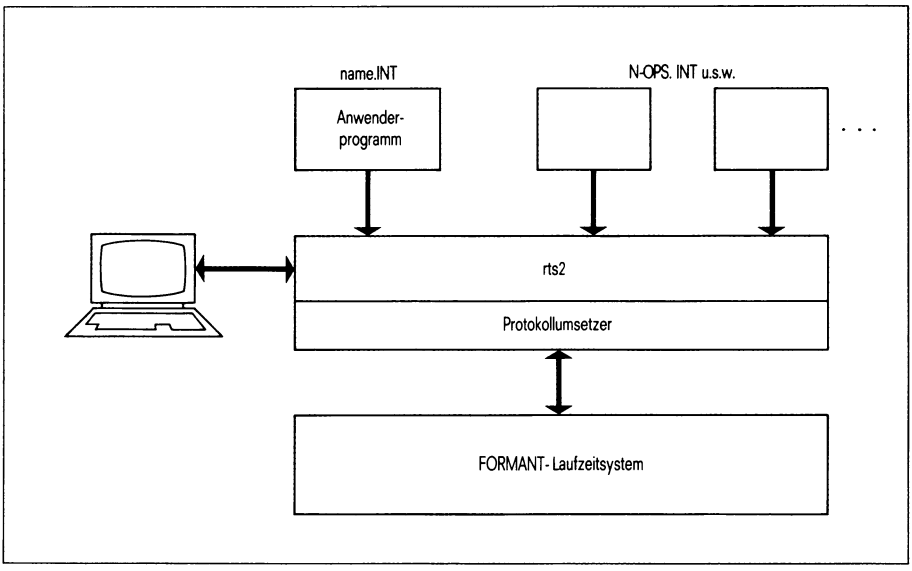


Bild 5-1: Ablauf einer COBOL-Anwendung

5.1 Der Programmrahmen

Für ein COBOL-Programm mit FORMANT-Aufrufen ist weiter nichts nötig, als die COPY-Elemente anzugeben, die die Aufrufe benötigen. Welche, das sehen Sie an den Beispielen zu den einzelnen Aufrufen.

Der erste FORMANT-Aufruf ist open session (N-OPS), der letzte ist close session (N-CLS).

Die normale Schnittstelle zu Datenein- und ausgabe bleibt erhalten. Sie können Sie verwenden um auf den 25-Zeilen-Modus umzuschalten.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
  .
  .
DATA DIVISION.
WORKING-STORAGE SECTION.

copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
copy "FORSK.LIB" in "/usr/lib/cobol/formant/copy".
copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .   und evtl. weitere
  .
01 init-par.
   05 init-text                 pic x(27)
      value "Programm beispiel1 geladen.".
   05 25-zeilen-modus           pic xxxx   value x"1B5B3075".

01 end-par.
   05 loeschen                  pic xxxx   value x"1B5B324A".
   05 24-zeilen-modus           pic xxxx   value x"1B5B3175".
   05 end-text                  pic x(27)
      value "Programm beispiel1 beendet.".
  .
  .
  .
PROCEDURE DIVISION.

display init-par upon console.
  .
  .
call "N-OPS" using n-main-par.
  .
  .   weitere beliebige FORMANT-Aufrufe
  .
call "N-CLS" using n-main-par.
  .
display end-par upon console.

STOP RUN.
```

5.2 Form der Aufrufe, Parameterübergabe und Ergebnis

Alle FORMANT-Aufrufe haben die Form

CALL funktion USING parameter, ...

Für funktion geben Sie an:

- nur den Namen der Funktion, wenn die entsprechenden Zwischencode-Dateien im selben Dateiverzeichnis liegen, in dem das Programm aufgerufen wird (z.B. als Verweise).

  Beispiel: call "N-CPI" using ... oder

```
77  f-aufruf          pic x(5).
 .
 .
move "N-CPI" to f-aufruf.
call f-aufruf using ...
```

- den vollen Pfadnamen, wenn dies nicht der Fall ist.

  Beispiel: 01 f-aufruf.

```
    05  f-verzeichnis pic x(27) value "/usr/lib/cobol/for-
        mant/int/".
    05  f-name        pic xxxx.
 .
 .
move "N-CPI" to f-name.
call f-aufruf using ...
```

Die Namenserweiterung .INT ist nur nötig, wenn Sie mit ANIMATOR testen wollen oder wenn Sie das Programm nicht mit cbrun aufrufen.

Parameterübergabe

Alle Parameterwerte werden als Zeichenvariable (PIC X) oder als numerische Variable (PIC 9) übergeben. FORMANT übergibt alle Werte als Adressen entsprechend der COBOL-Konventionen (CALL ... USING ...).

Für alle benötigten Parameterbereiche gibt es COPY-Elemente. Sie stehen im Dateiverzeichnis /usr/lib/cobol/formant/copy.

Die Parameter können Sie auf zwei Arten mit Werten versorgen:

- direkt mit dem Wert, z.B. move "V" to n-pi-mode.
- mit Datenfeldern, für die im COPY-Element FORVA.LIB Werte definiert sind, z.B. move vorhanden to n-pi-mode (N-VALUES).

Die zweite Möglichkeit ist vorzuziehen, weil sie besser dokumentiert und weil Sie damit unabhängiger von möglichen Änderungen sind.

Ergebnisabfrage

Jeder Aufruf liefert ein Ergebnis zurück. Das Ergebnis steht im Feld n-ret-code im Parameterbereich n-main-par. Die möglichen Werte finden Sie im COPY-Element FORER.LIB (N-ERRORS).

Bei jedem Aufruf sind die möglichen Werte mit Ihrer Bedeutung angegeben.

FORMANT unterscheidet:

n-ret-code = n-noerr (bzw. 0)
: Der Aufruf ist erfolgreich abgelaufen.

n-ret-code > n-noerr (bzw. 0)
: Leichter Fehler. Die Sitzung kann u.U. ohne weitere Maßnahmen fortgesetzt werden, weil der Fehler ohne Einfluß auf den Ablauf weiterer Aufrufe ist. Jedoch ist das Ergebnis dieses Aufrufs meist nicht das, was Sie wollten.

n-ret-code < n-noerr (bzw. 0)
: Schwerer Fehler. Weitere FORMANT-Aufrufe könnten nicht laufen, z.B wenn bei create p.i. ein falscher Maskenname angegeben wurde. Sie müssen den Fehler im Programm behandeln oder das Programm abbrechen.

5.3 Speicherbedarf, Hinweise zur Programmierung

Die einzelnen FORMANT-Aufrufe benötigen folgenden Speicherplatz:

Aufruf	Code/Daten	Aufruf	Code/Daten
N-CEM.INT	0512/1280	N-INF.INT	1536/1536
N-CFA.INT	1280/1536	N-MEP.INT	0768/1280
N-CFT.INT	0768/1280	N-MIR.INT	0768/1280
N-CHF.INT	0768/1280	N-MOM.INT	0512/1280
N-CHR.INT	0768/6144	N-MSP.INT	0768/1280
N-CLF.INT	0512/1280	N-OPS.INT	0512/1280
N-CLR.INT	0512/1280	N-PPI.INT	0512/1280
N-CLS.INT	0512/1280	N-RED.INT	0768/1280
N-CPI.INT	0768/1280	N-REP.INT	0768/1280
N-CSM.INT	0512/1280	N-ROM.INT	0512/1280
N-DEF.INT	0512/1280	N-RSP.INT	0512/1280
N-DPI.INT	0512/1280	N-SBE.INT	0512/1280
N-DUE.INT	0768/1280	N-SDR.INT	0768/3072
N-GEC.INT	0512/1280	N-SEC.INT	0768/1280
N-GEF.INT	0768/1280	N-SPI.INT	0512/1280
N-GER.INT	0512/6144	N-USR.INT	1024/3840
N-GFA.INT	1536/1536	N-WIR.INT	0768/1280
		N-WRT.INT	0512/1280

Da der Speicherplatz bei COBOL-Programmen eher knapp ist, kann es passieren, daß ein CALL-Aufruf wegen Speichermangel nicht ausgeführt wird. Dies erkennen Sie höchstens am fehlerhaften Verhalten des Programms, wenn Sie nicht die Klausel 'on overflow ... ' verwenden.

Um Speichermangel zu vermeiden können Sie die in den folgenden Abschnitten angedeuteten Möglichkeiten verwenden.

5.3.1 CALL - CANCEL

Aufrufe, die nur selten gebraucht werden, sollten Sie nicht im Speicher halten. Sie können Sie sofort nach dem Aufruf freigeben mit der COBOL-Anweisung CANCEL.

Allerdings löst dies noch nicht alle Probleme, weil beim Freigeben andere Programmteile im Speicher nicht nachrücken. Dadurch wird unter Umständen der Speicherplatz zerstückelt und nicht optimal genutzt.

5.3.2 Laden der FORMANT-Aufrufe bei Programmbeginn

Sie haben die Möglichkeit, FORMANT-Aufrufe nur in den Speicher zu laden, ohne die Funktion auszuführen. Dadurch können Sie gleich zu Programmbeginn alle FORMANT-Aufrufe laden, die Sie häufig benötigen. Sie vermeiden dadurch häufiges Nachladen und das Zerstückeln des Hauptspeichers, wie es beim CALL-CANCEL-Mechanismus auftritt.

Dazu dient das Lade-Byte n-init im Parameterbereich n-main-par. Ist dort "J" eingetragen, wird die Funktion nur geladen. Als Parameterangabe ist in diesem Fall immer nur n-main-par nötig.

Beispiel:

```
  .
  .
PROCEDURE DIVISION.
  .
  .
move "J" to n-init.
call "N-WRT" using n-main-par.
call "N-GER" using n-main-par.
call "N-SEC" using n-main-par.
  .
  .
move "N" to n-init.

call "N-OPS" ...
  .
  .
```

5.3.3 Segmentierung

Sie können COBOL-Programme mit FORMANT-Aufrufen wie üblich segmentieren. Beachten Sie aber, daß Sie nur Segmentnummern ab 50 (für Überlagerungssegmente) und 0 verwenden können. Innerhalb des festen Teils ist keine Segmentierung möglich.

5.3.4 Einschränkungen

- ACCEPT und DISPLAY sind sinnvoll nur außerhalb der FORMANT-Sitzung zu verwenden.
- Das NIL-Zeichen (X"00") ist als Zeichen nicht erlaubt, da es der Begrenzer für Zeichenfolgen ist. Es ist also kein abdruckbares Zeichen und Sie können es z.B. nicht als Füllzeichen für Datenfelder am Bildschirm verwenden.
- Namen für Benutzerrroutinen (ohne .INT), Masken und Maskenfelder dürfen 1-8 Zeichen lang sein entsprechend der COBOL-Klausel PIC X(8).
- Von FORMANT benutzte Namen siehe Auflistung der Copy-Elemente in Abschnitt 5.4.
- Felder mit Vorzeichen sind wie folgt zu definieren, z.B. 5 Zeichen lang mit Vorzeichen (für Eingabe und Ausgabe):

  01 V-FELD PIC S9(5) SIGN IS LEADING SEPARATE.

- Felder mit Nachkommastellen sind als Druckaufbereitungsfelder zu definieren, z.B. für ein Feld mit 2 Nachkommastellen (für Eingabe und Ausgabe):

  01 DP-FELD PIC ZZZ.ZZ.

  Felder mit Nachkommastellen und Vorzeichen sind ebenfalls als Drukkaufbereitungsfelder zu definieren.

5.4 COPY-Elemente für die Parameterbereiche

FORMANT stellt folgende COPY-Elemente bereit. Sie stehen alle im Dateiverzeichnis /usr/lib/cobol/formant/copy.

FORCP.LIB enthält:

N-EVENT-PAR	Ereignis-Parameter
N-OP-PAR	Eingabemodus

FOREP.LIB enthält:

N-ENVIRON-PAR	Umgebungs-Parameter

FORER.LIB enthält:

N-ERRORS	Fehlercodes

FORFP.LIB enthält:

N-FIELD-PAR	Feld-Parameter
N-ATTR-PAR	Feldeigenschaften

FORGP.LIB enthält:

N-MAIN-PAR	Globale FORMANT-Parameter
N-PI-PAR	Masken-Parameter

FORSK.LIB enthält:

N-SYS-KEYS	Log. Codes der Systemsteuertasten

FORSP.LIB enthält:

N-SESSION-PAR	Sitzungs-Paramter

FORUP.LIB enthält:

N-USER-PAR	Parameter für Benutzerroutinen

FORVA.LIB enthält:

N-VALUES	Parameterwerte

```
*****************************************************************
*****************************************************************
***                                                           ***
***      F O R M A N T  -  S T E U E R P A R A M E T E R      ***
***                                                           ***
***                      (VERSION V1.0)                       ***
***                                                           ***
*****************************************************************
*****************************************************************
***
***

*
*    ERREIGNISPARAMETER
*
 01  N-EVENT-PAR.
     05  N-EVENT-FELD       PIC X(8).
     05  N-EVT REDEFINES N-EVENT-FELD.
         10  N-EVENT-KEY        PIC 999.
         10  FILLER             PIC X(5).
     05  N-REACTION         PIC X(3)   VALUE  "STD".
     05  N-USER-REACT       PIC X(8).
*
*    OPERATION-PARAMETER
*
 01  N-OP-PAR.
    05  N-GEGENTASTP       PIC X       VALUE  "N".
```

```
*****************************************************************
*****************************************************************
***                                                           ***
***     F O R M A N T  -  E N V I R O N M E N T - P A R .     ***
***                                                           ***
***                     (VERSION V1.0)                        ***
***                                                           ***
*****************************************************************
*****************************************************************
***
***

*
*    UMGEBUNGSPARAMETER
*
 01  N-ENVIRON-PAR.
     05  N-FUELLZEICHEN      PIC X      VALUE  " ".
     05  N-HINTERGRUND       PIC X      VALUE  "N".
     05  N-CONTROL           PIC X      VALUE  "S".
     05  N-FELDLEERZ         PIC X      VALUE  " ".
     05  N-VOR-AUFB          PIC X      VALUE  "J".
     05  N-RUECK-AUFB        PIC X      VALUE  "J".
     05  N-LOESCHLINKS       PIC X      VALUE  "N".
     05  N-AUTODUPVORF       PIC X      VALUE  "N".
     05  N-STRTERM           PIC X      VALUE  "N".
```

```
*****************************************************************
*****************************************************************
***                                                           ***
***       F O R M A N T  -  R U E C K M E L D U N G E N       ***
***                                                           ***
***                      (VERSION V1.0)                       ***
***                                                           ***
*****************************************************************
*****************************************************************
***
***
 01 N-ERRORS.
*
*   NICHT BEHEBBARE FEHLER
*
*  FATAL ERROR
    05   N-ERFA                PIC  S99999    VALUE  -1.
*  NO SESSION-DESKRIPTOR
    05   N-ENSD                PIC  S99999    VALUE  -2.
*  NO SPACE AVAILABLE
    05   N-ENSA                PIC  S99999    VALUE  -4.
*  NO FORMANT MAP
    05   N-ENFO                PIC  S99999    VALUE  -8.
*  NO SYSTEM-CONTROL KEY
    05   N-ENSY                PIC  S99999    VALUE  -16.
*  NONIDENTIFYABLE ERROR
    05   N-ERR                 PIC  S99999    VALUE  -32.
*
*   BEHEBBARE FEHLER
*
*  NO CORRECT PARAMETER VALUE
    05   N-ENOP                PIC  S99999    VALUE  +1.
*  EMPTY PARAMETER (NULL-POINTER)
    05   N-ENUP                PIC  S99999    VALUE  +2.
*  NO FIELD NAME
    05   N-ENOF                PIC  S99999    VALUE  +4.
*  NO FIELD NAME FOR RECORD
    05   N-ENOF2               PIC  S99999    VALUE  +16.
*  NO FIELD NAME FOR PROCESSING
    05   N-ENOF1               PIC  S99999    VALUE  +32.
*  WRONG POSITION
    05   N-EPOS                PIC  S99999    VALUE  +64.
*  TEXT TOO LONG
    05   N-ETTL                PIC  S99999    VALUE  +128.
*  FIRST FIELD REACHED
    05   N-FFELD               PIC  S99999    VALUE  +256.
*  SAVING DENIED
    05   N-ESAV                PIC  S99999    VALUE  +512.
*  NO CORRECT VERSION
    05   N-ENVS                PIC  S99999    VALUE  +1024.
*  MORE DATA AVAILABLE
    05   N-EMDA                PIC  S99999    VALUE  +2048.
*  NO FORMANT FILE
    05   N-ENFL                PIC  S99999    VALUE  +4096.
```

```
*  SEQUENCE ERROR
     05   N-ESEQ                PIC  S99999    VALUE  +8192.
*
*     GUTFAELLE
*
*  NO ERROR
     05   N-NOERR               PIC  S99999    VALUE   0.
*  USER-PROC-REACT
     05   N-APL                 PIC  S99999    VALUE   8.
```

```
*****************************************************************
*****************************************************************
***                                                           ***
***       F O R M A N T  -  F E L D P A R A M E T E R         ***
***                                                           ***
***                    (VERSION V1.0)                         ***
***                                                           ***
*****************************************************************
*****************************************************************
***
***

*
*    FELDPARAMETER
*

 01  N-FIELD-PAR.
     05  N-ZEILE              PIC 99.
     05  N-SPALTE             PIC 99.
     05  N-LAENGE             PIC 9999.
     05  N-NAME               PIC X(8).
     05  N-VD-NAME            PIC X(8).
     05  N-VB-NAME            PIC X(8).
     05  N-F-TYP              PIC X.
     05  N-F-VORB             PIC X.

*
*    ATTRIBUTPARAMETER
*
 01  N-ATTR-PAR.
*
*    DARSTELLUNGSATTRIBUTE
*
     05  DARST               PIC XX    VALUE  "HE".
*
*    ADDITIVE DARSTELLUNG
*
     05  INVERS              PIC X     VALUE  "N".
     05  BLINKEND            PIC X     VALUE  "N".
     05  UNTERSTRICHEN       PIC X     VALUE  "N".
*
*    FARBEN
*
     05  FARBE               PIC XX    VALUE  " ".
*
*    ZEICHENSAETZE
*
     05  ZEICHENSATZ         PIC XXX    VALUE  " ".
```

```
*
*    BEARBEITUNGSATTRIBUTE
*
     05  MUSS              PIC X     VALUE  "N".
     05  GESCHUETZT        PIC X     VALUE  "N".
     05  AUSWEIS           PIC X     VALUE  "N".
     05  DUPLIZIER         PIC X     VALUE  "N".
     05  GROSS             PIC X     VALUE  "N".
     05  TRIGGER           PIC X     VALUE  "N".
     05  VOLLSTAENDIG      PIC X     VALUE  "N".
     05  SKIP              PIC X     VALUE  "N".
     05  GEGENTAST         PIC X     VALUE  "N".
     05  KORRSPERRE        PIC X     VALUE  "N".
*
*    FELDATTRIBUTE
*
     05  F-ATTR            PIC XX    VALUE  "AN".
*
     05  LINKS             PIC X     VALUE  "N".
     05  VORZ              PIC X     VALUE  "N".
     05  RECHTS            PIC X     VALUE  "N".
     05  DEZIMALST         PIC 99    VALUE   0.
*
*    FUELLZEICHEN
*
     05  AUSG-FZ           PIC X      VALUE  " ".
     05  EING-FZ           PIC X      VALUE  " ".
```

```
*****************************************************************
*****************************************************************
***                                                           ***
***   F O R M A N T  -  G L O B A L E   P A R A M E T E R     ***
***                                                           ***
***                    (VERSION V1.0)                         ***
***                                                           ***
*****************************************************************
*****************************************************************
***
***
 01  N-MAIN-PAR.
     05  N-SESSION-ID          PIC 99.
     05  N-VERSION             PIC 999        VALUE 100.
     05  N-NDSL                PIC 9999.
     05  N-RET-CODE            PIC S99999.
     05  N-RET-SYS             PIC 999.
     05  N-AKT-FELD            PIC X(8).
     05  N-INIT                PIC X           VALUE "N".

 01  N-PI-PAR.
     05  N-PI-MODE             PIC X.
     05  N-PI-NAME             PIC X(8).
     05  N-ERROR-MESSAGE       PIC X(40)      VALUE SPACES.
     05  N-STATUS-MESSAGE      PIC X(40)      VALUE SPACES.

*
*****************************************************************
```

```
*****************************************************************
*****************************************************************
***                                                           ***
***        F O R M A N T  -  S T E U E R T A S T E N          ***
***                                                           ***
***                    (VERSION V1.0)                         ***
***                                                           ***
*****************************************************************
*****************************************************************
***
***

 01  N-SYS-KEYS.
     05  N-AUSG              PIC 999      VALUE     0.
     05  N-FEPLUS            PIC 999      VALUE     1.
     05  N-FEMINUS           PIC 999      VALUE     2.
     05  N-HOME              PIC 999      VALUE     3.
     05  N-NEXT              PIC 999      VALUE     4.
     05  N-LAST              PIC 999      VALUE     5.
     05  N-BACKWARDS         PIC 999      VALUE     6.
     05  N-FORWARDS          PIC 999      VALUE     7.
     05  N-LVD               PIC 999      VALUE     8.
     05  N-SEND              PIC 999      VALUE     9.
     05  N-F1                PIC 999      VALUE     19.
     05  N-F2                PIC 999      VALUE     20.
     05  N-F3                PIC 999      VALUE     21.
     05  N-F4                PIC 999      VALUE     22.
     05  N-F5                PIC 999      VALUE     23.
     05  N-F6                PIC 999      VALUE     24.
     05  N-F7                PIC 999      VALUE     25.
     05  N-F8                PIC 999      VALUE     26.
     05  N-F9                PIC 999      VALUE     27.
     05  N-F10               PIC 999      VALUE     28.
     05  N-F11               PIC 999      VALUE     29.
     05  N-F12               PIC 999      VALUE     30.
     05  N-F13               PIC 999      VALUE     31.
     05  N-F14               PIC 999      VALUE     32.
     05  N-F15               PIC 999      VALUE     33.
     05  N-F16               PIC 999      VALUE     34.
     05  N-F17               PIC 999      VALUE     35.
     05  N-F18               PIC 999      VALUE     36.
     05  N-F19               PIC 999      VALUE     37.
     05  N-F20               PIC 999      VALUE     38.
     05  N-F21               PIC 999      VALUE     39.
     05  N-F22               PIC 999      VALUE     40.
     05  N-F23               PIC 999      VALUE     41.
     05  N-F24               PIC 999      VALUE     42.
     05  N-F25               PIC 999      VALUE     43.
     05  N-F26               PIC 999      VALUE     44.
     05  N-F27               PIC 999      VALUE     45.
     05  N-MENU              PIC 999      VALUE     46.
     05  N-PRINT             PIC 999      VALUE     47.
     05  N-HELP              PIC 999      VALUE     48.
```

```
05  N-START              PIC 999      VALUE     49.
05  N-END                PIC 999      VALUE     50.
05  N-CARDBAD            PIC 999      VALUE     52.
05  N-CARDOUT            PIC 999      VALUE     53.
```

```
*****************************************************************
*****************************************************************
***                                                           ***
***     F O R M A N T  -  S E S S I O N P A R A M E T E R     ***
***                                                           ***
***                     (VERSION V1.0)                        ***
***                                                           ***
*****************************************************************
*****************************************************************
***
***

*
*    PARAMETER
*
 01  N-SESSION-PAR.
     05  N-GERAETEKLASSE     PIC X     VALUE  "T".
     05  N-GERAETETYP        PIC X(20) VALUE  "fo97801".
     05  N-BERECHTIGUNG      PIC X     VALUE  "B".
     05  N-FEHLERTEXTE       PIC X     VALUE  "D".
```

```
*****************************************************************
*****************************************************************
***                                                           ***
***      F O R M A N T  -  S T E U E R P A R A M E T E R      ***
***                                                           ***
***                     (VERSION V1.0)                        ***
***                                                           ***
*****************************************************************
*****************************************************************
***
***

*
*    EINGANGSPARAMETER
*

 01  N-USER-PAR.
     05  NU-SESSION-ID       PIC 99.
     05  NU-VERSION          PIC 999.
     05  NU-RET-SYS          PIC 999.
     05  NU-AKT-FELD         PIC X(8).
     05  NU-PI-NAME          PIC X(8).
     05  NU-PI-MODE          PIC X.
*
*    Ausgangsparameter:
*

     05  NU-REACTION         PIC XXX.
```

```
*****************************************************************
*****************************************************************
***                                                           ***
***      F O R M A N T  -  P A R A M E T E R W E R T E        ***
***                                                           ***
***                    (VERSION V1.0)                         ***
***                                                           ***
*****************************************************************
*****************************************************************
***
***

 01  N-VALUES.

*
*    ALLGEMEINE PARAMETERWERTE
*
     05  JA              PIC X        VALUE  "J".
     05  NEIN            PIC X        VALUE  "N".

*
*    PI-PARAMETERWERTE
*
     05  VORHANDEN       PIC X        VALUE  "V".
     05  NEU             PIC X        VALUE  "N".
*
*    SESSION-PARAMETERWERTE
*
     05  TERM            PIC X        VALUE  "T".
     05  DRUCKER         PIC X        VALUE  "D".
     05  IN-PER          PIC X        VALUE  "I".
     05  LESEN           PIC X        VALUE  "L".
     05  SCHREIBEN       PIC X        VALUE  "S".
     05  BEIDES          PIC X        VALUE  "B".
     05  F-STD           PIC X        VALUE  "S".
     05  F-NAT           PIC X        VALUE  "N".
*
*    ENVIRONMENT-PARAMTERWERTE
*
     05  NORM            PIC X        VALUE  "N".
     05  INV             PIC X        VALUE  "I".
     05  SATZ            PIC X        VALUE  "S".
     05  FELD            PIC X        VALUE  "F".
*
*    FELD-PARAMETERWERTE
*
     05  FIX             PIC X        VALUE  "F".
     05  VAR             PIC X        VALUE  "V".
*
*    ATTRIBUT-PARAMETERWERTE
*
     05  HELL            PIC XX       VALUE  "HE".
     05  HALBHELL        PIC XX       VALUE  "HH".
```

```
           05  DUNKEL          PIC XX       VALUE  "DU".
           05  WEISS           PIC XX       VALUE  "WE".
           05  ROT             PIC XX       VALUE  "RT".
           05  GRUEN           PIC XX       VALUE  "GR".
           05  GELB            PIC XX       VALUE  "GE".
           05  BLAU            PIC XX       VALUE  "BL".
           05  STAN            PIC XXX      VALUE  "INT".
           05  NATIONAL        PIC XXX      VALUE  "DTH".
           05  MATH            PIC XXX      VALUE  "MAT".
           05  MOSAICS         PIC XXX      VALUE  "MOS".
           05  KLAMMERN        PIC XXX      VALUE  "KLA".
           05  IBM             PIC XXX      VALUE  "IBM".
           05  ALPHA           PIC XX       VALUE  "AL".
           05  ALPHANUM        PIC XX       VALUE  "AN".
           05  NUM             PIC XX       VALUE  "NU".
           05  SONDERZ         PIC XX       VALUE  "SZ".
       *
       *   STEUER-PARAMETERWERTE
       *
           05  E-STD           PIC XXX      VALUE  "STD".
           05  E-IGN           PIC XXX      VALUE  "IGN".
           05  E-APL           PIC XXX      VALUE  "APL".
           05  E-USR           PIC XXX      VALUE  "USR".
```

5.5 FORMANT-COBOL-Aufrufe

Im folgenden Teil sind alle FORMANT-COBOL-Aufrufe beschrieben. Die Aufrufe sind alphabetisch sortiert. Zu jedem Aufruf finden Sie ein Beispiel, aus dem Sie entnehmen können, welche Datendefinitionen nötig sind und wie Sie die Parameter versorgen.

Die Pfeile bei den Parametern geben an, ob Sie den Parameter versorgen müssen oder ob FORMANT den Wert des Parameters liefert:

→ Ein Eingangsparameter ist vom Programm zu versorgen

← Ein Ergebnisparameter wird vom FORMANT-Aufruf zurückgeliefert.

FORMANT-Aufrufe nach Funktionsgruppen

Sitzungskontrolle

N-OPS Sitzung eröffnen - open session
N-MSP Sitzungsparameter ändern - modify session parameters
N-RSP Sitzungsparameter abfragen - request session parameters
N-CLS Sitzung schließen - close session

Umgebungskontrolle

N-MEP Umgebungsparameter ändern - modify environment parameters
N-REP Umgebungsparameter abfragen - request environment parameters
N-MIR Reaktion auf Systemsteuertasten - modify input reaction table
N-DUE Benutzerroutinen für Triggerfelder - define user exits

Maskenbehandlung

Maske

N-CPI Maske anlegen - create presentation image
N-DPI Maske schließen - destroy presentation image
N-SPI Maske abspeichern - save presentation image

N-PPI Maske ausdrucken - print presentation image

Datenfelder

N-GEF Datenfeld übernehmen - get field
N-CHF Feldinhalt ändern - change field
N-CLF Feldinhalt löschen - clear field
N-INF Feld einfügen - insert field
N-DEF Feld ausfügen - delete field
N-GFA Feldeigenschaften abfragen - get field attribute
N-CFA Feldeigenschaften ändern - change field attribute
N-CFT Feldtyp und Reihenfolge ändern - change field type

Datensatz

N-GER Datensatz aufbauen - get record
N-CHR Datensatz in die Maske übertragen - change record
N-CLR Alle Felder der Maske löschen - clear record
N-SDR Dupliziersatz setzen - set dup record

Schreibmarke

N-SEC Schreibmarke positionieren - set cursor
N-GEC Schreibmarkenposition abfragen - get cursor

Meldungen

N-CEM Fehlermeldung ausgeben - change error message
N-CSM Statusmeldung ausgeben - change status message
N-SBE Signalton erzeugen - set beeper

Datentransport

N-WRT Maske ausgeben - write
N-WIR Maske ausgeben und Daten einlesen - write read
N-RED Daten einlesen - read

Eingabeparameter

N-MOM Eingabeparameter ändern - modify operating mode
N-ROM Eingabeparameter abfragen - request operating mode

Fehlermeldung ausgeben - change error message

Mit dem Aufruf können Sie eigene Fehlermeldungen ausgeben. Die Meldung erscheint in der Systemzeile von Spalte 1 bis Spalte 40. FORMANT sperrt die Tastatur und erzeugt einen Signalton. Nach einer Fehlermeldung kann erst wieder eingegeben werden, wenn die Taste [WORD] (ERS) oder [q] gedrückt wurde. Drücken einer dieser Tasten löscht gleichzeitig den Fehlertext.

CALL "N-CEM" USING N-MAIN-PAR, N-PI-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-PI-PAR.            enthalten im COPY-Element FORGP.
→    05  N-ERROR-MESSAGE    PIC X(40)        VALUE SPACES.
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Session Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ERROR-MESSAGE	Text der Fehlermeldung. Der Text darf bis zu 40 Zeichen lang sein.

Hinweis:

Der Text der Fehlermeldung wird überschrieben, wenn gleich danach ein zweiter Aufruf N-CEM folgt.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.
  .
    move "Satz nicht gefunden." to n-error-message.
    move "N-CEM" to f-aufruf.

    call f-aufruf using n-main-par, n-pi-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
  .
```

Der Text, der n-error-message zugewiesen wird, erscheint in der untersten Bildschirmzeile als Fehlermeldung.

Feldeigenschaften ändern - change field attribute

Der Aufruf ändert für ein Feld den Wert von Feldeigenschaften (Attributen). Pro Feld ist ein Aufruf nötig.

Alle Feldeigenschaften sind in Abschnitt 3.4 erklärt.

CALL "N-CFA" USING N-MAIN-PAR, N-ATTR-PAR, N-FIELD-PAR.

Parameter:

01	N-MAIN-PAR.		enthalten im COPY-Element FORGP.	
→	05	N-SESSION-ID	PIC 99.	
→	05	N-INIT	PIC X	VALUE "N".
←	05	N-RET-CODE	PIC S99999.	
01	N-ATTR-PAR.		enthalten im COPY-Element FORFP.	
→	05	DARST	PIC XX	VALUE "HE".
→	05	INVERS	PIC X	VALUE "N".
→	05	BLINKEND	PIC X	VALUE "N".
→	05	UNTERSTRICHEN	PIC X	VALUE "N".
→	05	FARBE	PIC XX	VALUE " ".
→	05	ZEICHENSATZ	PIC XXX	VALUE " ".
→	05	MUSS	PIC X	VALUE "N".
→	05	GESCHUETZT	PIC X	VALUE "N".
→	05	AUSWEIS	PIC X	VALUE "N".
→	05	DUPLIZIER	PIC X	VALUE "N".
→	05	GROSS	PIC X	VALUE "N".
→	05	TRIGGER	PIC X	VALUE "N".
→	05	VOLLSTAENDIG	PIC X	VALUE "N".
→	05	SKIP	PIC X	VALUE "N".
→	05	GEGENTAST	PIC X	VALUE "N".
→	05	KORRSPERRE	PIC X	VALUE "N".
→	05	F-ATTR	PIC XX	VALUE "AN".
→	05	LINKS	PIC X	VALUE "N".
→	05	VORZ	PIC X	VALUE "N".
→	05	RECHTS	PIC X	VALUE "N".
→	05	DEZIMALST	PIC 99	VALUE 0.
→	05	AUSG-FZ	PIC X	VALUE " ".
→	05	EING-FZ	PIC X	VALUE " ".

01 N-FIELD-PAR.	enthalten im COPY-Element FORFP.
→ 05 N-NAME	PIC X(8).

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ENOP 1	Falsche Parameterangabe .
N-ENOF 4	Feldname nicht bekannt.

Felddarstellung

DARST	Felddarstellung am Bildschirm.
HELL	Das Feld wird hell dargestellt (Standard).
HALBHELL	Das Feld wird halbhell dargestellt.
DUNKEL	Das Feld wird nicht sichtbar dargestellt.
INVERS	Das Feld wird invers dargestellt. Werte: JA, NEIN.
BLINKEND	Das Feld blinkt. Werte: JA, NEIN.
UNTERSTRICHEN	Das Feld wird unterstrichen. Werte: JA, NEIN.
FARBE	in Version 1.0 nicht unterstützt.

Zeichensatz

ZEICHENSATZ	Zeichensatz.
STAN NATIONAL MATH MOSAICS KLAMMERN IBM	Alle Zeichensätze finden Sie im Anhang abgebildet.

Bearbeitungseigenschaften

Die folgenden Bearbeitungseigenschaften können Sie kombinieren, soweit sinnvoll. Als Werte sind jeweils möglich:

JA	Eigenschaft wird gesetzt.
NEIN	Eigenschaft wird nicht gesetzt.

MUSS	Mußfeld.
GESCHUETZT	Geschütztes Feld.
AUSWEIS	Ausweisleserfeld.
DUPLIZIER	Duplizierfeld.
GROSS	Großschreibung.
TRIGGER	Triggerfeld.
VOLLSTAENDIG	Vollständigkeitsfeld.
SKIP	Skipfeld.
GEGENTAST	Prüffeld.
KORRSPERRE	Prüffeld, nicht korrigierbar.

Feldausrichtung und zulässige Eingabezeichen

F-ATTR	Zulässige Eingabezeichen. Mögliche Werte sind:
ALPHANUM	alphanumerisches Feld (A-Z,a-z,0-9,' ')
ALPHA	alphabetisches Feld (A-Z,a-z,' ')
NUM	numerisches Feld (0-9)
SONDERZ	Sonderzeichen (alle abdruckbaren Zeichen)
LINKS	Linksausrichtung. Werte: NEIN, JA.
VORZ	Vorzeichenfeld. Werte: NEIN, JA.
RECHTS	Rechtsausrichtung. Werte: NEIN,JA.
DEZIMALST	Nachkommastellen. Werte 0 - 15.

Füllzeichen

AUSG-FZ	Feldfüllzeichen. Zugelassen sind alle abdruckbaren Zeichen.
EING-FZ	Datenfüllzeichen. Zugelassen sind alle abdruckbaren Zeichen.
N-NAME	Name des Feldes, dessen Eigenschaften Sie ändern wollen.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .

PROCEDURE DIVISION.
  .
  move "m-name" to n-name.
  move "J" to blinkend.
  move "N-CFA" to f-aufruf.

  call f-aufruf using n-main-par, n-attr-par, n-field-par
               on overflow
               move "OVERFLOW" to fehlerart
               perform fehler.

  if n-ret-code not = n-noerr
               perform fehler.

  move "m-ort" to n-name.
  move "J" to gross.

  call f-aufruf using n-main-par, n-attr-par, n-field-par.

  if n-ret-code not = n-noerr
               perform fehler.

  cancel f-aufruf.
  .
  .
```

Das Feld m-name soll blinken, die Eingabe in das Feld m-ort soll in Grossbuchstaben umgesetzt werden.

Feldtyp ändern - change field type

Der Aufruf ändert für ein Feld den Feldtyp von FIX auf VAR oder umgekehrt. Damit ändert sich u.U. auch die Reihenfolge in der Bearbeitung bzw. im Datensatz.

CALL "N-CFT" USING N-MAIN-PAR, N-FIELD-PAR.

Parameter:

01	N-MAIN-PAR.		enthalten im COPY-Element FORGP.	
→	05	N-SESSION-ID	PIC 99.	
→	05	N-INIT	PIC X	VALUE "N".
←	05	N-RET-CODE	PIC S99999.	
01	N-FIELD-PAR.		enthalten im COPY-Element FORFP.	
→	05	N-NAME	PIC X(8).	
→	05	N-VD-NAME	PIC X(8).	
→	05	N-VB-NAME	PIC X(8).	
→	05	N-F-TYP	PIC X.	

Erklärung der Parameter

N-SESSION-ID — Sitzungs-Deskriptor (siehe N-OPS).

N-INIT — Lade-Byte: N Funktion ausführen.
J Funktion nur laden.

N-RET-CODE — Ergebnis des Aufrufs.

N-ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR	0	Aufruf erfolgreich.
N-ENOP	1	Typ nicht bekannt.
N-ENOF	4	N-NAME nicht bekannt.
N-ENOF2	16	N-VD-NAME nicht bekannt.
N-ENOF1	32	N-VB-NAME nicht bekannt.

N-NAME — Name des Feldes, das Sie ändern wollen.

N-VD-NAME — Beim Aufruf 'get record' werden die Felder in der hier festgelegten Reihenfolge in einem Satzbereich übergeben. Sie erhalten also einen strukturierten Datensatz.

	Für das erste Feld geben Sie SPACES an. Die Angabe gilt nur für variable Felder.
N-VB-NAME	Name des vorhergehenden Feldes in der Bearbeitung. Damit legen Sie fest, in welcher Reihenfolge die Felder am Bildschirm eingegeben werden. Für das erste Feld geben Sie SPACES an.
N-F-TYP	gewünschter Typ des Feldes.
FIX	Textfeld
VAR	variables Feld

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.
  .
  .
    move "m-strass" to n-name.
    move fix to n-f-typ.
    move "N-CFT" to f-aufruf.

    call f-aufruf using n-main-par, n-field-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
  .
```

Der Aufruf ändert den Typ des Feldes m-ort in FIX. Aus einem variablen Feld wird ein Textfeld.

Feldinhalt ändern - change field

Der Aufruf ändert den aktuellen Feldinhalt des angegebenen Feldes. 'change field' bearbeitet variable Felder und Textfelder.

CALL "N-CHF" USING N-MAIN-PAR, N-FIELD-PAR, text.

Parameter:

```
01  N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→   05  N-SESSION-ID          PIC 99.
→   05  N-INIT               PIC X              VALUE "N".
←   05  N-RET-CODE           PIC S99999.

01  N-FIELD-PAR.         enthalten im COPY-Element FORFP.
→   05  N-NAME             PIC X(8).
→   05  N-LAENGE           PIC 9999.
```

text
→ alphanumerisches Datenfeld (PIC X) oder Literal.

Erklärung der Parameter

N-SESSION-ID		Sitzungs-Deskriptor (siehe N-OPS).
N-INIT		Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE		Ergebnis des Aufrufs.
N-ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR	0	Aufruf erfolgreich.
N-ENOF	4	Feldname nicht bekannt.
N-ENOF1	32	Ausgabefehler.
N-ETTL	128	Feldinhalt länger als 80 Zeichen.
N-NAME		Name des Feldes, dessen Inhalt Sie ändern wollen.
N-LAENGE		Länge des Textes, der in das Feld einzutragen ist.

text — alphanumerisch definiertes Feld mit dem neuen Feldinhalt oder ein alphanumerisches Literal.
Ist das angegebene Feld kürzer als die Feldlänge, füllt FORMANT mit Feldfüllzeichen auf. Ist es länger, wird der Rest abgeschnitten.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".

  77 inhalt                    pic x(5)      value  "HALS ".
  .
  .
PROCEDURE DIVISION.
  .
  .
    move "m-schuh" to n-name.
    move 5 to n-laenge.
    move "N-CHF" to f-aufruf.


    call f-aufruf using n-main-par, n-field-par, inhalt
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                perform fehler.

    cancel f-aufruf.
  .
  .
```

Der Aufruf besetzt das Textfeld m-schuh mit dem neuen Inhalt HALS. m-schuh ist 5 Zeichen lang, daher wird inhalt auch in dieser Länge definiert.

Datensatz in die Maske übertragen - change record

Der Aufruf überträgt einen Datensatz in die aktuelle Maske. Der Satz muß so strukturiert sein, wie er mit 'get record' aufgebaut wurde, das heißt die Reihenfolge der Felder im Datensatz muß mit der definierten Reihenfolge übereinstimmen. Der Aufruf ändert nur die variablen Felder der Maske.

CALL "N-CHR" USING N-MAIN-PAR, record.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
→    05  N-NDSL                PIC 9999.
←    05  N-RET-CODE            PIC S99999.
```

record
→ Datenbereich mit Feldern vom Typ PIC X und PIC 9 oder Literal.

Erklärung der Parameter

N-SESSION-ID — Sitzungs-Deskriptor (siehe N-OPS).

N-INIT — Lade-Byte: N Funktion ausführen. J Funktion nur laden.

N-NDSL

Laenge des Datensatzes. FORMANT ersetzt Zeichen nur bis zur angegebenen Länge. Ist die angegebene Länge kleiner als die tatsächliche Länge der Daten in der Maske bleibt der Rest unverändert. Ist die angegebene Länge größer, ignoriert FORMANT die restlichen Zeichen.

N-RET-CODE — Ergebnis des Aufrufs.

N-ENSA	-4	Kein Speicher verfügbar im Prüfmodus.
N-ENSD	-2	Situngs-Deskriptor nicht bekannt.
N-NOERR	0	Aufruf erfolgreich.
N-ENOF1	32	Ein vorangegangener Fehler wurde erkannt.
N-ENOF	4	Keine variablen Felder in der Maske definiert.

record — Alphanumerisches Datenfeld oder Literal mit dem Inhalt des strukturierten Datensatzes. Das ist der Inhalt aller Felder der Maske, die auszugeben sind.

Beispiel:

```
WORKING-STORAGE-SECTION.
   .
   .
copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
   .
   .
01 kunden-satz.
   05 kd-nummer          pic 9(6).
   05 kunde              pic x(15).
      .
   .
   .

PROCEDURE DIVISION.
   .
   .
   move 75 to n-ndsl.
   move "N-CHR" to f-aufruf.

   call f-aufruf using n-main-par, kunden-satz
                on overflow
                move "OVERFLOW" to fehlerart
                perform fehler.

   if n-ret-code not = n-noerr
                perform fehler.
   cancel f-aufruf.
   .
   .
```

Der Datensatz kunden-satz wird in die Maske übertragen. Die Struktur entspricht der 'Reihenfolge im Datensatz', wie sie beim Erstellen der Maske definiert wird. Die gesamte Länge des Datensatzes ist 75 Zeichen.

Feldinhalt löschen - clear field

Der Aufruf löscht den Inhalt des angegebenen Feldes in der Maske mit Bildschirmfüllzeichen (Leerzeichen). Mit dem Aufruf können Sie auch den Inhalt von Textfeldern löschen.

CALL "N-CLF" USING N-MAIN-PAR, N-FIELD-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-FIELD-PAR.         enthalten im COPY-Element FORFP.
→    05  N-NAME              PIC X(8).
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ENOP 1	Zeiger auf NULL bei fieldname.
N-ENOF 4	Feldname nicht bekannt.
N-ENOF1 32	Ausgabefehler.
N-NAME	Name des Feldes, dessen Inhalt zu löschen ist.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .

PROCEDURE DIVISION.
  .

          move "m-schuh" to n-name.
          move "N-CLF" to f-aufruf.

          call f-aufruf using n-main-par, n-field-par
                       on overflow
                       move "OVERFLOW" to fehlerart
                       perform fehler.

          if n-ret-code not = n-noerr
                       perform fehler.

          move "m-groess" to n-name.

          call f-aufruf using n-main-par, n-field-par.

          if n-ret-code not = n-noerr
                       perform fehler.

          cancel f-aufruf.
  .
  .
```

Die beiden Aufrufe bewirken, daß die Felder m-schuh und m-groess mit Leerzeichen gelöscht werden.

Variable Felder der Maske löschen - clear record

Der Aufruf löscht den Inhalt aller variablen Felder der Maske mit Bildschirmfüllzeichen (Standard: Leerzeichen).

CALL "N-CLR" USING N-MAIN-PAR.

Parameter:

01	N-MAIN-PAR.	enthalten im COPY-Element FORGP.		
→	05 N-SESSION-ID	PIC 99.		
→	05 N-INIT	PIC X	VALUE "N".	
←	05 N-RET-CODE	PIC S99999.		

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).	
N-INIT	Lade-Byte:	N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.	
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.	
N-NOERR 0	Aufruf erfolgreich.	
N-ENOF 4	Keine variablen Felder in der Maske definiert.	
N-ENOF1 32	Ausgabefehler.	

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.
  .
  .
  move "N-CLR" to f-aufruf.

  call f-aufruf using n-main-par
```

```
                on overflow
                move "OVERFLOW" to fehlerart
                perform fehler.

if n-ret-code not = n-noerr
                perform fehler.

cancel f-aufruf.
.
.
```

Der Aufruf löscht alle variablen Felder der aktuellen Maske mit Leerzeichen.

Sitzung schließen - close session

N-CLS schließt die FORMANT-Sitzung. Von FORMANT belegter Speicherplatz wird freigegeben. N-CLS muß der letzte FORMANT-Aufruf im Programm sein.

CALL "N-CLS" USING N-MAIN-PAR.

Parameter:

01	N-MAIN-PAR.	enthalten im COPY-Element FORGP.		
→	05 N-SESSION-ID	PIC 99.		
→	05 N-INIT	PIC X	VALUE "N".	
←	05 N-RET-CODE	PIC S99999.		

Erklärung der Parameter

N-SESSION-ID		Sitzungs-Deskriptor (siehe N-OPS).
N-INIT		Lade-Byte: N Funktion ausführen.
N-RET-CODE		Ergebnis des Aufrufs.
N-ENSD	-2	Es gibt keine Sitzung mit diesem Deskriptor.
N-NOERR	0	Die Sitzung wurde geschlossen.
N-ESEQ	8192	Falsche Reihenfolge: kein N-OPS oder Aufruf in einer Benutzerroutine.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.
  .
  .
    move "N-CLS" to f-aufruf.

    call f-aufruf using n-main-par
```

```
                    on overflow
                    move "OVERFLOW" to fehlerart
                    perform fehler.

        if n-ret-code not = n-noerr
                    perform fehler.

        cancel f-aufruf.
      .
      .
```

Der Aufruf schließt die FORMANT-Sitzung.

Maske anlegen - create presentation image

N-CPI legt den Namen der aktuellen Maske fest. Gibt es bereits eine Maske dieses Namens, wird sie geladen. Soll die Maske neu erzeugt werden, reserviert N-CPI nur Speicherplatz. Bevor Sie N-CPI erneut aufrufen, müssen Sie mit N-DPI die aktuelle Maske schliessen. Es kann immer nur eine Maske angelegt sein.

CALL "N-CPI" USING N-MAIN-PAR, N-PI-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-PI-PAR.            enthalten im COPY-Element FORGP.
→    05  N-PI-MODE             PIC X.
→    05  N-PI-NAME             PIC X(8).
```

Erklärung der Parameter

N-SESSION-ID		Sitzungs-Deskriptor (siehe N-OPS).
N-INIT		Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE		Ergebnis des Aufrufs.
N-ENFO	-8	Name ist keine Maske.
N-ENSA	-4	Kein Speicherplatz verfügbar.
N-ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR	0	Aufruf erfolgreich.
N-ENOF	4	Kein gültiger Name oder keine gültigen Felder.
N-ENFL	4096	Maske inkonsistent.
N-ESEQ	8192	Falsche Reihenfolge: kein N-OPS, kein N-DPI oder Aufruf in einer Benutzerroutine.
N-PI-MODE		gibt an, ob die Maske vorhanden ist oder ob sie neu angelegt werden soll.

NEU	Die Maske soll neu angelegt werden. Die Felder der neuen Maske definieren Sie mit N-INF-Aufrufen.
VORHANDEN	Die Maske steht in einer Datei: der Dateiname ist gleich dem Maskennamen.
N-PI-NAME	Name der Maske (max. 8 Zeichen): Bei Angebe NEU: beliebige Zeichenfolge. Bei Angabe VORHANDEN: SINIX-Dateiname.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.
  .
  .
  move "maskep" to n-pi-name.
  move vorhanden to n-pi-mode.
  move "N-CPI" to f-aufruf.

  call f-aufruf using n-main-par, n-pi-par
               on overflow
               move "OVERFLOW" to fehlerart
               perform fehler.

  if n-ret-code not = n-noerr
               perform fehler.

  cancel f-aufruf.
  .
  .
```

Die Maske mit Namen maskep wird angelegt. Die Datei maskep muß im aktuellen Dateiverzeichnis vorhanden sein.

Statusmeldung ausgeben - change status message

Mit dem Aufruf können Sie eine Statusmeldung ausgeben. Die Statusmeldung erscheint in der Systemzeile von Spalte 41 bis Spalte 79. FORMANT sperrt die Tastatur nicht. Die Statusmeldung löschen können Sie, indem Sie Leerzeichen als Statusmeldung ausgeben.

CALL "N-CSM" USING N-MAIN-PAR, N-PI-PAR.

Parameter:

01	N-MAIN-PAR.	enthalten im COPY-Element FORGP.		
→	05 N-SESSION-ID	PIC 99.		
→	05 N-INIT	PIC X	VALUE "N".	
←	05 N-RET-CODE	PIC S99999.		
01	N-PI-PAR.	enthalten im COPY-Element FORGP.		
→	05 N-STATUS-MESSAGE	PIC X(40)	VALUE SPACES.	

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Session Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-STATUS-MESSAGE	Text der Statusmeldung. Der Text darf bis zu 39 Zeichen lang sein.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".


  01 st-info.
```

```
       02 info   pic x(33) value "Anzahl uebergebener Datensaetze".
       02 anz    pic 999   value  0.
    .
    .
PROCEDURE DIVISION.
    .
    .
      add 1 to anz.
      move st-info to n-status-message.
      move "N-CSM" to f-aufruf.
      call f-aufruf using n-main-par, n-pi-par
                   on overflow
                   move "OVERFLOW" to fehlerart
                   perform fehler.

      if n-ret-code not = n-noerr
                   perform fehler.

      cancel f-aufruf.
    .
    .
```

Die in st-info definierte Statusmeldung erscheint in der letzten Bildschirmzeile ab Spalte 40.

Feld ausfügen - delete field

N-DEF fügt ein Feld aus der aktuellen Maske aus. An den Positionen der restlichen Felder ändert sich nichts.

N-DEF kann variable Felder und Textfelder ausfügen. Beim Ausfügen variabler Felder ändert sich der Aufbau des Datensatzes.

CALL "N-DEF" USING N-MAIN-PAR, N-FIELD-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→  05  N-SESSION-ID            PIC 99.
→  05  N-INIT                  PIC X              VALUE "N".
←  05  N-RET-CODE              PIC S99999.

01   N-FIELD-PAR.         enthalten im COPY-Element FORFP.
→  05  N-NAME                 PIC X(8).
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ENOF 4	Feldname nicht bekannt.
N-NAME	Name des Feldes, das auszufügen ist.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
```

```
PROCEDURE DIVISION.
  .
  .

    move "m-groess" to n-name.
    move "N-DEF" to f-aufruf.

    call f-aufruf using n-main-par, n-field-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.
    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.

  .
  .
```

Der Aufruf löscht das Feld m-groess aus der aktuellen Maske. Die Maskendatei wird dadurch nicht verändert. Wenn die Maske erneut mit N-CPI geladen wird, ist das Feld vorhanden.

Maske schließen - destroy presentation image

N-DPI schließt dis aktuelle Maske und gibt deren Speicherplatz frei. Anschließend können Sie mit N-CPI eine neue Maske anlegen oder mit N-CLS die FORMANT-Sitzung abschließen.
N-DPI löscht den Bildschirm und ist unbedingt vor dem Schließen der FORMANT-Sitzung aufzurufen.

CALL "N-DPI" USING N-MAIN-PAR.

Parameter:

01	N-MAIN-PAR.		enthalten im COPY-Element FORGP.	
→	05	N-SESSION-ID	PIC 99.	
→	05	N-INIT	PIC X	VALUE "N".
←	05	N-RET-CODE	PIC S99999.	

Erklärung der Parameter

N-SESSION-ID		Sitzungs-Deskriptor (siehe N-OPS).
N-INIT		Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE		Ergebnis des Aufrufs.
N-ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR	0	Aufruf erfolgreich.
N-ESEQ	8192	Falsche Reihenfolge: kein N-OPS oder kein N-CPI oder Aufruf in Benutzerroutine.

Beispiel:

```
WORKING-STORAGE SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formamt/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formamt/copy".
  .
  .
PROCEDURE DIVISION.
  .
  .
```

```
move "N-DPI" to f-aufruf.

call f-aufruf using n-main-par
              on overflow
              move "OVERFLOW" to fehlerart
              perform fehler.

if n-ret-code not = n-noerr
              perform fehler.

cancel f-aufruf.
.
.
```

Die aktuelle Maske wird geschlossen. FORMANT löscht den Bildschirm.

Benutzerroutinen für Triggerfelder - define user exits

Mit N-DUE geben Sie den Namen eines Unterprogramms an, das aufzurufen ist, wenn ein Triggerfeld ausgefüllt wurde. Die Eigenschaft Triggerfeld wird beim Erstellen der Maske vergeben. Pro Triggerfeld ist ein Aufruf N-DUE nötig. Sie können natürlich für verschiedene Triggerfelder dieselbe Funktion vereinbaren.

'define user exits' ist nach 'create p.i.' aufzurufen.

```
CALL "N-DUE" USING N-MAIN-PAR, N-EVENT-PAR.
```

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→  05  N-SESSION-ID            PIC 99.
→  05  N-INIT                  PIC X              VALUE "N".
←  05  N-RET-CODE              PIC S99999.

01   N-EVENT-PAR.         enthalten im COPY-Element FORCP.
→  05  N-EVENT-FIELD         PIC X(8).
→  05  N-USER-REACT          PIC X(8).
```

Erklärung der Parameter

N-SESSION-ID		Sitzungs-Deskriptor (siehe N-OPS).
N-INIT		Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE		Ergebnis des Aufrufs.
N-ENSA	-4	Keine Benutzerroutine mehr möglich (Speichermangel).
N-ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR	0	Erfolgreicher Aufruf.
N-ENOP	1	N-USR-REACT nicht versorgt.
N-ENOF	4	Feldname nicht bekannt.
N-ESEQ	8192	Falsche Reihenfolge: kein N-OPS oder kein N-CPI.
N-EVENT-FIELD		Name eines Triggerfeldes. Wird dieses Feld bei der Eingabe fertig ausgefüllt, ruft FORMANT

die in N-USER-REACT angegebene Benutzerroutine auf. Hat das Feld nicht die Eigenschaft Triggerfeld, passiert nichts.

N-USER-REACT — Name einer Benutzerroutine. Das ist ein Unterprogramm das FORMANT aufruft, wenn das angegebene Triggerfeld ausgefüllt wurde.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORCP.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .

PROCEDURE DIVISION.
  .
  .
  move "m-drei" to n-event-feld.
  move "brout1" to n-user-react.
  move "N-DUE" to f-aufruf.

  call f-aufruf using n-main-par, n-event-par,
                on overflow
                move "OVERFLOW" to fehlerart
                perform fehler.

  if n-ret-code not = n-noerr
                perform fehler.

  cancel f-aufruf.
  .
  .
```

Der Aufruf definiert das COBOL-Unterprogramm brout1.INT als Benutzerroutine für ein Triggerfeld. FORMANT durchläuft das Unterprogramm, wenn das Feld m-drei der aktuellen Maske bei der Eingabe verlassen wird. Das Feld m-drei muß die Feldeigenschaft 'Triggerfeld' haben. Siehe dazu auch das Beispiel in Abschnitt 7.1.

Schreibmarkenposition abfragen - get cursor

Der Aufruf liefert die aktuelle Position der Schreibmarke.

CALL "N-GEC" USING N-MAIN-PAR, N-FIELD-PAR, offset.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→  05  N-SESSION-ID            PIC 99.
→  05  N-INIT                  PIC X              VALUE "N".
←  05  N-RET-CODE              PIC S99999.

01   N-FIELD-PAR.         enthalten im COPY-Element FORFP.
←  05  N-NAME                 PIC X(8).

offset
←  Datenfeld PIC 99.
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-NAME	Name des Feldes, in dem die Schreibmarke steht.
offset	Datenfeld (PIC 99) für die Position der Schreibmarke innerhalb des Feldes. Feldanfang ist 0.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".

  77 o-set                        pic 99.
  .
  .

PROCEDURE DIVISION.
  .
  .
    move "N-GEC" to f-aufruf.

    call f-aufruf using n-main-par, n-field-par, o-set
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
  .
```

Nach dem Aufruf steht in n-name (enthalten in n-field-par) der Name des Feldes, in dem die Schreibmarke zuletzt stand. o-set enthält den Abstand zum Feldbeginn.

Datenfeld übernehmen - get field

Der Aufruf übernimmt den aktuellen Inhalt eines Datenfeldes in einen Datenbereich des Anwendungsprogramms. Das Feld wird als Zeichenfolge übergeben. Der aktuelle Inhalt hängt davon ab ,was beim vorhergehenden Leseaufruf (n_red oder n_wir) eingegeben wurde. Ist kein Leseaufruf vorangegangen oder ist es ein Textfeld (FIX), ist der aktuelle Inhalt die Vorbelegung. Leere Felder füllt FORMANT mit Datenfüllzeichen, ebenso wie nicht ausgefüllte Feldreste.

CALL "N-GEF" USING N-MAIN-PAR, N-FIELD-PAR, inhalt.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→  05  N-SESSION-ID            PIC 99.
→  05  N-INIT                  PIC X              VALUE "N".
←  05  N-RET-CODE              PIC S99999.

01   N-FIELD-PAR.         enthalten im COPY-Element FORFP.
→  05  N-NAME                 PIC X(8).
→  05  N-LAENGE               PIC 9999.
```

inhalt
← Datenfeld vom Typ PIC X oder PIC 9.

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ENOF 4	Feldname nicht bekannt.
N-EMDA 2048	Es sind noch Daten vorhanden.
N-NAME	Name des Feldes, wie bei FORMANTGEN oder bei n_inf festgelegt.
N-LAENGE	Länge des Datenfeldes.

inhalt — Datenfeld für den Feldinhalt. FORMANT überträgt nur so viele Zeichen, wie im Parameter N-LAENGE angegeben ist. Ist das Eingabefeld länger als das Feld 'inhalt', kehrt der Aufruf mit EMDA zurück.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
  77 wahl                pic x.

PROCEDURE DIVISION.
  .
  .
   move 1 to n-laenge.
   move "m1-wahl" to n-name.
   move "N-GEF" to f-aufruf.

   call f-aufruf using n-main-par, n-field-par, wahl
                on overflow
                move "OVERFLOW" to fehlerart
                perform fehler.

   if n-ret-code not = N-NOERR
                perform fehler.

   cancel f-aufruf.
  .
  .
```

Der Inhalt des 1 Zeichen langen Feldes m1-wahl wird in das Feld wahl eingetragen.

Datensatz aufbauen - get record

Der Aufruf baut aus den variablen Feldern der aktuellen Maske einen strukturierten Datensatz auf. Die Reihenfolge wurde beim Erstellen der Maske mit FORMANTGEN festgelegt oder beim Aufruf N-INF. Der Aufruf N-GER übernimmt den zuletzt mit N-WIR oder N-RED gelesenen Inhalt. Leere Felder füllt N-GER mit dem jeweils definierten Datenfüllzeichen.

CALL "N-GER" USING N-MAIN-PAR, satz.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
→    05  N-NDSL                PIC 9999.
←    05  N-RET-CODE            PIC S99999.
```

satz
← Datenbereich mit Feldern vom Typ PIC X und PIC 9.

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-NDSL	Länge des strukturierten Datensatzes. Der Aufruf überträgt nur soviele Zeichen, wie hier angegeben sind. Sind in der Maske mehr Daten vorhanden, liefert der Aufruf N-EMDA in N-RET-CODE.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-EMDA 2048	Es sind noch Daten vorhanden.
satz	Bereich für den strukturierten Datensatz. Der Bereich muß in der Anwendung bereitgestellt

sein. N-GER liefert den Datensatz als Zeichenfolge. Bitte beachten Sie die Einschränkungen für numerische Felder mit Vorzeichen bzw. mit Nachkommastellen (siehe Abschnitt 5.3.4).

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".

   01 d-daten.
      05 d-name                    pic x(11).
      05 d-vorname                 pic x(11).
      05 d-strasse                 pic x(12).
      05 d-ort ,                   pic x(21).
      05 d-telefon                 pic x(16).
      05 d-jahre                   pic 99.
      05 d-groesse                 pic 999.
  .

PROCEDURE DIVISION.
  .
  .
    move 76 to n-ndsl.
    move spaces to d-daten.
    move "N-GER" to f-aufruf.

    call f-aufruf using n-main-par, d-daten
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
```

Der Inhalt des Datensatzes d-daten wird gelesen. In n-ndsl gibt man die Laenge des gesamten Datensatzes an.

Feldeigenschaften abfragen - get field attribute

Der Aufruf liefert für ein Feld die aktuellen Werte der Feldeigenschaften (Attribute). Pro Feld ist ein Aufruf nötig.

CALL "N-GFA" USING N-MAIN-PAR, N-ATTR-PAR, N-FIELD-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X              VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-ATTR-PAR.          enthalten im COPY-Element FORFP.

←    05  DARST               PIC XX     VALUE "HE".
←    05  INVERS              PIC X      VALUE "N".
←    05  BLINKEND            PIC X      VALUE "N".
←    05  UNTERSTRICHEN PIC X            VALUE "N".
←    05  FARBE               PIC XX     VALUE " ".
←    05  ZEICHENSATZ         PIC XXX     VALUE " ".
←    05  MUSS                PIC X      VALUE "N".
←    05  GESCHUETZT          PIC X      VALUE "N".
←    05  AUSWEIS             PIC X      VALUE "N".
←    05  DUPLIZIER           PIC X      VALUE "N".
←    05  GROSS               PIC X      VALUE "N".
←    05  TRIGGER             PIC X      VALUE "N".
←    05  VOLLSTAENDIG        PIC X      VALUE "N".
←    05  SKIP                PIC X      VALUE "N".
←    05  GEGENTAST           PIC X      VALUE "N".
←    05  KORRSPERRE          PIC X      VALUE "N".
←    05  F-ATTR              PIC XX     VALUE "AN".
←    05  LINKS               PIC X      VALUE "N".
←    05  VORZ                PIC X      VALUE "N".
←    05  RECHTS              PIC X      VALUE "N".
←    05  DEZIMALST           PIC 99     VALUE  0.
←    05  AUSG-FZ             PIC X       VALUE " ".
←    05  EING-FZ             PIC X       VALUE " ".
```

01	N-FIELD-PAR.		enthalten im COPY-Element FORFP.
→	05	N-NAME	PIC X(8).
←	05	N-F-TYP	PIC X.

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs:
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ENOP 1	Falsche Parameterangabe .
N-ENOF 4	Feldname nicht bekannt.

Feldeigenschaften

Die Erklärung aller Parameter finden Sie beim Aufruf N-CFA.

N-NAME	Name des Feldes, dessen Eigenschaften Sie abfragen wollen.
N-F-TYP	FORMANT liefert den Feldtyp (FIX oder VAR).

Beispiel:

```
WORKING-STORAGE SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .

PROCEDURE DIVISION.
  .
  .
    move "m-ort" to n-name.
    move "N-GFA" to f-aufruf.

    call f-aufruf using n-main-par, n-attr-par, n-field-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.
```

```
if n-ret-code not = n-noerr
              perform fehler.

cancel f-aufruf.

display f-attr-par.
.
.
```

Nach dem Aufruf enthält der Bereich n-attr-par alle Feldeigenschaften des Feldes m-ort der aktuellen Maske.

Feld einfügen - insert field

N-INF fügt ein neues Feld in die aktuelle Maske ein. Dabei definieren Sie:

- die Position des Feldes am Bildschirm,
- die Position des Feldes im Datensatz,
- die Reihenfolge bei der Bearbeitung (bei variablen Feldern).

Mit N-INF-Aufrufen können Sie eine neue Maske aufbauen oder eine vorhandene abändern (siehe auch N-CPI).

CALL "N-INF" USING N-MAIN-PAR, N-FIELD-PAR, N-ATTR-PAR, text.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-FIELD-PAR.         enthalten im COPY-Element FORFP.
→    05  N-ZEILE              PIC 99.
→    05  N-SPALTE             PIC 99.
→    05  N-LAENGE             PIC 9999.
→    05  N-NAME               PIC X(8).
→    05  N-VD-NAME            PIC X(8).
→    05  N-VB-NAME            PIC X(8).
→    05  N-F-TYP              PIC X.
→    05  N-F-VORB             PIC X.

01   N-ATTR-PAR.          enthalten im COPY-Element FORFP.
→    05  DARST                PIC XX         VALUE "HE".
→    05  INVERS               PIC X          VALUE "N".
→    05  BLINKEND             PIC X          VALUE "N".
→    05  UNTERSTRICHEN        PIC X          VALUE "N".
→    05  FARBE                PIC XX         VALUE " ".
→    05  ZEICHENSATZ          PIC XXX        VALUE " ".
→    05  MUSS                 PIC X          VALUE "N".
→    05  GESCHUETZT           PIC X          VALUE "N".
→    05  AUSWEIS              PIC X          VALUE "N".
→    05  DUPLIZIER            PIC X          VALUE "N".
```

→	05	GROSS	PIC X	VALUE "N".
→	05	TRIGGER	PIC X	VALUE "N".
→	05	VOLLSTAENDIG	PIC X	VALUE "N".
→	05	SKIP	PIC X	VALUE "N".
→	05	GEGENTAST	PIC X	VALUE "N".
→	05	KORRSPERRE	PIC X	VALUE "N".
→	05	F-ATTR	PIC XX	VALUE "AN".
→	05	LINKS	PIC X	VALUE "N".
→	05	VORZ	PIC X	VALUE "N".
→	05	RECHTS	PIC X	VALUE "N".
→	05	DEZIMALST	PIC 99	VALUE 0.
→	05	AUSG-FZ	PIC X	VALUE " ".
→	05	EING-FZ	PIC X	VALUE " ".

text
→ alphanumerisches Datenfeld (PIC X) oder Literal.

Erklärung der Parameter

N-SESSION-ID — Sitzungs-Deskriptor (siehe N-OPS).

N-INIT — Lade-Byte: N Funktion ausführen.
J Funktion nur laden.

N-RET-CODE — Ergebnis des Aufrufs:

N-ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR	0	Aufruf erfolgreich.
N-ENOP	1	Fehler bei den Attributen.
N-ENOF	4	Feld ist schon vorhanden.
N-ENOF2	16	N-VD-NAME nicht bekannt.
N-ENOF1	32	N-VB-NAME nicht bekannt.
N-EPOS	64	ungültige Positionsangabe.
N-ETTL	128	Längenfehler.
N-ESEQ	8192	Falsche Reihenfolge: z.B. kein N-OPS oder kein N-CPI.

N-ZEILE — Zeilenposition des Feldes (1-24).

N-SPALTE — Spaltenposition des Feldes(1-80).

N-LAENGE — Länge des Feldes (max. 80 Zeichen bei Beginn in Spalte 1).

N-NAME — Name des Feldes, das Sie einfügen wollen.

N-VD-NAME	Beim Aufruf 'get record' werden die Felder in der hier festgelegten Reihenfolge in einem Satzbereich übergeben. Sie erhalten also einen strukturierten Datensatz. Für das erste Feld geben Sie SPACES an. Die Angabe gilt nur für variable Felder.
N-VB-NAME	Name des vorhergehenden Feldes in der Bearbeitung. Damit legen Sie fest, in welcher Reihenfolge die Felder am Bildschirm eingegeben werden. Für das erste Feld geben Sie SPACES an.
N-F-TYP	gewünschter Typ des Feldes.
FIX	Textfeld
VAR	variables Feld
N-F-VORB	gibt an, ob das Feld mit dem Inhalt von 'text' vorbelegt werden soll.
JA	Feld vorbelegen.
NEIN	Feld leer ausgeben. FORMANT füllt das Feld mit Leerzeichen (= Bildschirmfüllzeichen).
Felddarstellung	
DARST	Felddarstellung am Bildschirm.
HELL	Das Feld wird hell dargestellt (Standard).
HALBHELL	Das Feld wird halbhell dargestellt.
DUNKEL	Das Feld wird nicht sichtbar dargestellt.
INVERS	Das Feld wird invers dargestellt. Werte: JA, NEIN.
BLINKEND	Das Feld blinkt. Werte: JA, NEIN.
UNTERSTRICHEN	Das Feld wird unterstrichen. Werte: JA, NEIN.
FARBE	in Version 1.0 nicht unterstützt.

Zeichensatz

ZEICHENSATZ	Zeichensatz.
STAN	(Standard)
NATIONAL	
MATH	
MOSAICS	
KLAMMERN	
IBM	Alle Zeichensätze finden Sie im Anhang abgebildet.

Bearbeitungseigenschaften

Die folgenden Bearbeitungseigenschaften können Sie kombinieren, soweit sinnvoll. Als Werte sind jeweils möglich:

JA	Eigenschaft wird gesetzt.
NEIN	Eigenschaft wird nicht gesetzt.
MUSS	Mußfeld.
GESCHUETZT	Geschütztes Feld.
AUSWEIS	Ausweislesefeld.
DUPLIZIER	Duplizierfeld.
GROSS	Großschreibung.
TRIGGER	Triggerfeld.
VOLLSTAENDIG	Vollständigkeitsfeld.
SKIP	Skipfeld.
GEGENTAST	Prüffeld.
KORRSPERRE	Korrektursperre für Prüffeld, wirkt nur zusammen mit GEGENTAST.

Feldausrichtung und zulässige Eingabezeichen

F-ATTR	Zulässiger Zeichenvorrat für die Eingabe. Mögliche Werte sind:
ALPHANUM	alphanumerisches Feld (A-Z,a-z,' ',0-9)
ALPHA	alphabetisches Feld (A-Z,a-z,' ')
NUM	numerisches Feld (0-9)
SONDERZ	alphanum. und Sonderzeichen (alle abdruckbaren Zeichen)
LINKS	Linksausrichtung. Werte: NEIN, JA.
VORZ	Vorzeichenfeld. Werte: NEIN, JA.
RECHTS	Rechtsausrichtung. Werte: NEIN,JA.
DEZIMALST	Nachkommastellen. Werte 0 - 15.
Füllzeichen	
AUSG-FZ	Feldfüllzeichen. Zugelassen sind alle abdruckbaren Zeichen.
EING-FZ	Datenfüllzeichen. Zugelassen sind alle abdruckbaren Zeichen.
→ text	definiert den Inhalt von Textfeldern. Bei variablen Feldern wird text nur ausgewertet, wenn Sie bei N-F-VORB JA angegeben haben. Das Feld 'text' muß in der Länge N-LAENGE definiert sein.

Eine Tabelle, wie die Attribute den Angaben bei FORMANTGEN zugeordnet sind, finden Sie in Abschnitt 3.4. Dort sind auch alle Attribute erklärt.

Hinweis:

Die Parameter in N-FIELD-PAR müssen Sie alle versorgen. Für N-ATTR-PAR gelten die entsprechenden VALUE-Angaben.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP" in "/usr/lib/cobol/formant/copy".
  copy "FORFP" in "/usr/lib/cobol/formant/copy".
  copy "FORER" in "/usr/lib/cobol/formant/copy".
  copy "FORVA" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.
  .
  .
        move "f-frage" to n-name.
        move 16 to n-zeile.
        move 54 to n-spalte.
        move 8 to n-laenge.
        move ja to n-f-vorb.
        move fix to n-f-typ.
        move "N-INF" to f-aufruf.

    call f-aufruf using
                n-main-par, n-field-par, n-attr-par, "welches?"
                on overflow
                move "OVERFLOW" to fehlerart
                perform fehler.

    if n-ret-code not = N-NOERR
                perform fehler.

    cancel f-aufruf.
  .
  .
```

Der Aufruf fügt ein Textfeld mit den angegebenen Eigenschaften ein. Alle Felder von n-attr-par, die nicht ausdrücklich versorgt werden, enthalten Standardwerte (oder Werte aus einem vorhergegangenen Aufruf). Das neue Feld erhält den Namen f-frage. Es wird nur in der aktuellen Maske eingefügt und nicht in der Maskendatei.

Umgebungsparameter ändern - modify environment parameters

Der Aufruf ändert während des Programmlaufs die Umgebungsparameter. Für die Umgebungsparameter setzt FORMANT beim Öffnen der Sitzung Standardwerte. Die jeweils aktuellen Werte können Sie mit dem Aufruf N-REP abfragen.

CALL "N-MEP" USING N-MAIN-PAR, N-ENVIRON-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-ENVIRON-PAR.       enthalten im COPY-Element FOREP.
→    05  N-FUELLZEICHEN       PIC X          VALUE " ".
→    05  N-HINTERGRUND        PIC X          VALUE "N".
→    05  N-CONTROL            PIC X          VALUE "S".
→    05  N-FELDLEERZ          PIC X          VALUE "_".
→    05  N-VOR-AUFB           PIC X          VALUE "J".
→    05  N-RUECK-AUFB         PIC X          VALUE "J".
→    05  N-LOESCHLINKS        PIC X          VALUE "N".
→    05  N-AUTODUPVORF        PIC X          VALUE "N".
→    05  N-STRTERM            PIC X          VALUE "N".
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Erfolgreicher Aufruf.
N-ENOP 1	Fehlerhafter Parameterwert.
N-ESEQ 8192	Falsche Reihenfolge: z.B. kein N-OPS oder kein N-CPI oder Aufruf in Benutzerroutine.

N-FUELLZEICHEN — Bildschirmfüllzeichen. Dies ist das Zeichen, mit dem alle Stellen des Bildschirms aufgefüllt werden, die nicht von Feldern belegt sind. Auch Felder mit Attribut 'leer' füllt FORMANT mit Bildschirmfüllzeichen.

" " — Leerzeichen (Standard).

alle abdruckbaren Zeichen.

Hinweis: Dieser Parameter ist nur zulässig, wenn Masken im Programm neu aufgebaut werden und nicht bei Masken, die mit FORMANTGEN erstellt wurden. Beim Maskenwechsel ist das Bildschirmfüllzeichen immer entsprechend einzustellen!

N-HINTERGRUND — Hintergrunddarstellung des Bildschirms.

NORM — Der Bildchirm wird normal dargestellt (Standard).

INV — Der Bildschirm wird invers dargestellt.

Hinweis: Dieser Parameter ist nur zulässig, wenn Masken im Programm neu aufgebaut werden und nicht bei Masken, die mit FORMANTGEN erstellt wurden. Beim Maskenwechsel ist die Hintergrunddarstelllung immer entsprechend einzustellen!

N-CONTROL — Eingabereaktion, feldweise oder satzweise. Damit stellen Sie ein, wann FORMANT die Kontrolle an das Anwendungsprogramm zurückgibt.

SATZ — Satzweise Eingabe. Nach dem Leseaufruf erhält das Programm erst die Steuerung, wenn alle Datenfelder der Maske ausgefüllt sind (Standard).

FELD — Feldweise Eingabe. Ein Leseaufruf liefert gerade ein Datenfeld.

N-FELDLEERZ — Feldleerzeichen. Dies ist das Zeichen, mit dem ein leeres Feld angezeigt wird, sobald es zur Eingabe ansteht.

"_"	Unterstrich (Standard).
alle abdruckbaren Zeichen.	
N-VOR-AUFB	Feldaufbereitung beim Überspringen von Feldern. Die Angabe bestimmt die Reaktion, wenn ein Feld bei der Eingabe vorwärts übersprungen wird (nicht bei feldweiser Reaktion).
JA	Übersprungene Felder werden aufbereitet (Standard). FORMANT füllt mit Feldfüllzeichen auf.
NEIN	Übersprungene Felder bereitet FORMANT nicht auf. Felder, die bereits ausgefüllt sind, werden nicht gelöscht.
N-RUECK-AUFB	Feldaufbereitung bei Rückpositionierung. Die Angabe bestimmt die Reaktion, wenn ein Feld bei der Eingabe rückwärts verlassen wird (nicht bei feldweiser Reaktion)
JA	Das Feld wird aufbereitet (Standard). Es gilt somit als ausgefüllt.
NEIN	Der Inhalt des Feldes wird gelöscht. Es gilt nicht als ausgefüllt.
N-LOESCHLINKS	zeichenweise Rückpositionierung. Die Angabe bestimmt die Reaktion, wenn man die Taste ← drückt.
NEIN	Die Zeichen werden nicht gelöscht (Standard).
JA	Die Zeichen werden gelöscht. Hinweis: Die Taste ⌫ löscht die Zeichen immer.
N-AUTODUPVORF	Automatisches Duplizieren in vorbelegten Feldern.
NEIN	In Feldern mit dem Attribut 'vorbelegt' wird nicht automatisch dupliziert (Standard). Ist die Duplizierfunktion eingeschaltet, bleiben in diesen Feldern die vorbelegten Daten erhalten (siehe auch Aufruf n_sdr).

JA	Ist die Duplizierfunktion eingeschaltet, werden die vorbelegten Daten mit den Duplizierdaten überschrieben. Die Duplizierfunktion wirkt nur bei variablen Feldern.
N-STRTERM	Stringterminator. Der Stringterminator ist X'00' und begrenzt die Eingabefelder, die FORMANT beim Aufruf 'get record' übergibt.
NEIN	Die Felder werden nicht mit X'00' abgeschlossen. (Standard).
JA	Jedes Feld wird mit X'00' abgeschlossen. Beachten Sie bitte, daß dadurch der Satz für 'get record' länger wird.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FOREP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".
  .
PROCEDURE DIVISION.
  .
  .
    move feld to n-control.
    move "." to n-feldleerz.
    move "N-MEP" to f-aufruf.

    call f-aufruf using n-main-par, n-environ-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

   if n-ret-code not = n-noerr
                perform fehler.

   cancel f-aufruf.
  .
```

Der Aufruf ändert die Eingabereaktion auf feldweisen Betrieb. Gleichzeitig wird als Feldleerzeichen das Zeichen Punkt festgelegt.

Reaktion auf Systemsteuertasten - modify input reaction table

Der Aufruf legt die Reaktion auf Systemsteuertasten fest. Pro Systemsteuertaste ist ein Aufruf nötig. Sie können

- Tasten sperren,
- als Reaktion eigene Benutzerroutinen ausführen,
- wieder Standardbehandlung vereinbaren,
- die Eingabe beenden, d.h. die Steuerung geht wieder an das Anwendungsprogramm zurück.

Funktionstasten sind ein Sonderfall der Systemsteuertasten. Deren Bedeutung müssen Sie immer selbst definieren.

Die standardmäßige Wirkung aller Tasten ist beschrieben in Abschnitt 2.3.

CALL "N-MIR" USING N-MAIN-PAR, N-EVENT-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-EVENT-PAR.         enthalten im COPY-ELEMENT FORCP.
     05  N-EVENT-FELD          PIC X(8).
     05  N-EVT REDEFINES N-EVENT-FELD.
→        10  N-EVENT-KEY       PIC 999.
         10  FILLER            PIC X(5).
→    05  N-REACTION            PIC X(3)       VALUE  "STD".
→    05  N-USER-REACT          PIC X(8).
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).	
N-INIT	Lade-Byte:	N Funktion ausführen. J Funktion nur laden.

N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Erfolgreicher Aufruf.
N-ENOP 1	Falscher Parameterwert.
N-ENUP 2	N-USER-REACT nicht versorgt.
N-EVENT-KEY	Logischer Code der Systemsteuertaste. Erlaubt sind die Angaben wie in der Tabelle unten.
N-REACTION	Reaktion auf die Systemsteuertaste:
E-STD	Standardbehandlung. Die Taste löst die Funktion aus, wie in Abschnitt 2.3 beschrieben.
	Funktionstasten:
	F1 bis F27, MENU, HELP, START, END:
	FORMANT gibt die Steuerung sofort an das Anwendungsprogramm und liefert mit dem Leseaufruf den logischen Code der Taste. Beim nächsten Leseaufruf setzt FORMANT die Eingabe mit dem nächsten Feld fort. War das aktuelle Feld bereits das letzte der Maske, kehrt dieser nächste Leseaufruf mit 'N-AUSG' zurück.
E-IGN	Taste sperren. FORMANT meldet 'E1 Systemsteuerzeichen unzulaessig', wenn die Taste gedrückt wird und es ist keine Funktionstaste. Auf gesperrte Funktionstasten reagiert FORMANT nicht.
E-USR	Die in N-USER-REACT angegebene Benutzerroutine soll ausgeführt werden. Anschließend setzt FORMANT fort, wie in der Benutzerroutine definiert (siehe Kapitel 7).
E-APL	Das Anwendungsprogramm soll die Steuerung erhalten. Der Leseaufruf wird beendet und liefert den logischen Code der Systemsteuertaste. Dies entspricht genau der Standardbehandlung einer Funktionstaste.
	E-APL wirkt bei Funktionstasten wie STD.

Die Tabelle unten zeigt, welche Angabe bei welchen Tasten erlaubt ist.

N-USER-REACT — Name einer Benutzerroutine. Das ist ein Unterprogramm das ausgeführt werden soll, wenn die angegebene Taste gedrückt wurde. N-USER-REACT wird nur bei E-USR ausgewertet. Bei E-STD, E-IGN oder E-APL können in N-USR-REACT Leerzeichen stehen.

Ergebnis:

Tabelle der änderbaren Tasten

Änderbar sind:	F1 bis F27	E-STD E-IGN E-USR
	MENU	
	HELP START END	
	SEND LVD	E-STD E-IGN E-USR E-APL
	FE+ FE-	
	LAST HOME NEXT	
	BACKWARDS FORWARDS	
nur sperrbar sind:	FKOR KOR ANZ	E-IGN
	VALID	
	DUP	

Alle übrigen Tasten können Sie nicht ändern. Zur Tastenbelegung siehe auch Abschnitt 2.3.8 und Anhang.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORCP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORSK.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .

PROCEDURE DIVISION.
  .
  .
```

```
move n-f10 to n-event-key.
move e-usr to n-reaction.
move "u-prog" to n-user-react.
move "N-MIR" to f-aufruf.

call f-aufruf using n-main-par, n-event-par
                    on overflow
                    move "OVERFLOW" to fehlerart
                    perform fehler.

if n-ret-code not = n-noerr
                    perform fehler.
cancel f-aufruf.
.
.
```

Der Aufruf ordnet der Taste F10 das COBOL-Unterprogramm u-prog.INT zu (Benutzerroutine). FORMANT ruft diese Benutzerroutine auf, wenn die Taste F10 gedrückt wird (siehe auch Beispiel in Abschnitt 7.1).

Eingabeparameter ändern - modify operating mode

Der Aufruf ändert Eingabeparameter. In Version 1.0 ist dies nur

- Gegentastprüfen ein- und ausschalten.

CALL "N-MOM" USING N-MAIN-PAR, N-OP-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-OP-PAR.            enthalten im COPY-Element FORCP.
→    05  N-GEGENTASTP        PIC X          VALUE "N".
```

Erklärung der Parameter

N-SESSION-ID — Sitzungs-Deskriptor (siehe N-OPS).

N-INIT — Lade-Byte: N Funktion ausführen. J Funktion nur laden.

N-RET-CODE — Ergebnis des Aufrufs.

N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ENOP 1	Falscher Parameterwert.

N-GEGENTASTP — Gegentastprüfen. Die Eingabe in ein Datenfeld wird gegen einen vorgegebenen Inhalt geprüft. Folgende Voraussetzungen müssen erfüllt sein:

1. Das Datenfeld muß die Eigenschaft 'Prüffeld' oder 'Prüffeld nicht korrigierbar' haben.
2. Sie müssen Gegentastprüfen einschalten.
3. Sie müssen einen Datensatz vorgeben, gegen dessen Inhalt zu prüfen ist. Das kann ein Datensatz aus einer Datei sein oder ein

Datensatz, der bereits eingegeben und mit 'get record' geholt wurde.
Den Datensatz geben Sie mit 'change record' vor, nachdem Sie Gegentastprüfen eingeschaltet haben. Soll dieser Satz nicht am Bildschirm gezeigt werden, ist vor dem 'change record' ein 'clear record' aufzurufen.

4. Mit einem Leseaufruf fordern Sie die Eingabe im Prüfmodus an Aus diesem Leseaufruf kehrt FORMANT zurück mit dem logischen Code der Systemsteuertaste wie üblich.

   Beachten Sie aber: Im Prüfmodus fordert FORMANT nur Prüffelder zur Eingabe an und füllt alle anderen Felder selbst mit dem Inhalt der Vorgabe aus (logischer Code der Systemsteuertaste: AUSG).

   FORMANT prüft die Eingabe gegen die Vorgabe. Bei Abweichungen meldet FORMANT 'E13 Prueffehler !'.

   Zum Ablauf der Eingabe und zu den Korrekturmöglichkeiten siehe Abschnitt 2.3.6.

   Prüffelder kann man vorzeitig nur über Funktionstasten verlassen. Damit können Sie zum Beispiel einen Ausgang für 'nicht korrigierbare Prüffelder' vorsehen.

5. Wollen Sie einen Datensatz bzw. ein Feld normal einlesen, müssen Sie das Gegentastprüfen ausschalten.

NEIN — Gegentastprüfen aus (Standard): normale Eingabe eines Datensatzes bzw. Datenfeldes.

JA — Gegentastprüfen ein: FORMANT fordert nur Prüffelder an und prüft die Eingabe gegen einen mit 'change record' vorgegebenen Datensatz. Die übrigen Felder füllt FORMANT mit dem Inhalt dieses Datensatzes

selbst aus. Das gilt bei feldweiser und satzweiser Eingabereaktion.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORCP.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.
  .
  .
    move ja to n-gegentastp.
    move "N-MOM" to f-aufruf.

    call f-aufruf using n-main-par, n-op-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
  .
```

Der Aufruf schaltet den Prüfmodus ein. Siehe auch Beispiel in Abschnitt 4.3.

Sitzungsparameter ändern - modify session parameters

Der Aufruf ändert während des Programmlaufs die Sitzungsparameter. Für die Sitzungsparameter setzt FORMANT beim Eröffnen der Sitzung Standardwerte. Die jeweils aktuellen Werte können Sie mit dem Aufruf 'request session parameters' abfragen.

CALL "N-MSP" USING N-MAIN-PAR, N-SESSION-PAR.

Parameter:

```
01  N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→   05  N-SESSION-ID          PIC 99.
→   05  N-INIT                PIC X          VALUE "N".
←   05  N-RET-CODE            PIC S99999.

01  N-SESSION-PAR.       enthalten im Copy-Element FORSP.
→   05  N-GERAETEKLASSE     PIC X          VALUE "T".
→   05  N-GERAETE-
        TYP                 PIC X(20)      VALUE "fo978-
                                                  01".
→   05  N-BERECHTIGUNG      PIC X          VALUE "B".
→   05  N-FEHLERTEXTE       PIC X          VALUE "D".
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Erfolgreicher Aufruf.
N-ENOP 1	Fehlerhafter Parameterwert.
N-ESEQ 8192	Falsche Reihenfolge: z.B. kein N-OPS oder kein N-CPI oder Aufruf in Benutzerroutine.
N-GERAETEKLASSE	Geräteklasse, die unterstützt werden soll.
TERM	Datensichtstation (Standard).

DRUCKER

Drucker. Diesen Eintrag ermöglicht es, Druckermasken auszugeben, die mit FORMANTGEN aus mehreren Formaten zusammengesetzt wurden.
Wenn N-GERAETEKLASSE auf DRUCKER gesetzt ist,

- können Sie eine Druckermaske ausgeben. Dazu verwenden Sie den Aufruf N-WRT oder den Aufruf N-PPI. Beide Aufrufe wirken in diesem Fall gleich.
- dürfen Sie folgende Aufrufe nicht verwenden:
  N-WIR
  N-RED

  Diese Aufrufe liefern das Ergebnis N-ESEQ.
- haben folgende Aufrufe keine Wirkung:
  N-MIR
  N-DUE
  N-SEC
  N-SDR

Die Länge eines Nettodatensatzes ist begrenzt auf 5kB.

IN-PER

interaktive Peripherie (noch nicht unterstützt).

GERAETETYP

Gerätetyp, der unterstützt werden soll. FORMANT nimmt den entsprechenden Eintrag aus der Datei /etc/termcap.

"fo97801"

Datensichtstation 97801 (Standard).

"9001"

Drucker 9001.

"9004"

Drucker 9004.

Hinweis: Die Drucker 9001 und 9004 werden nur mit dem vorhandenen Standardzeichensatz unterstützt. FORMANT bedient den Drucker über das Kommando lpr. In der Shell-Variablen FOPRT können Sie auch ein ande-

res Kommando oder ein eigenes Programm vereinbaren, das die Druckausgabe macht (siehe auch N-PPI).

N-BERECHTIGUNG — möglicher Zugriff auf das Gerät.

BEIDES — Es ist Lesen und Schreiben erlaubt (Standard).

LESEN — Es ist nur Lesen erlaubt.

SCHREIBEN — Es ist nur Schreiben erlaubt.

N-FEHLERTEXTE — gibt an, aus welcher Datei die FORMANT-Fehlermeldungen gelesen werden sollen.

F-STD — FORMANT liest aus der Datei 'ftext.std' (Standard).

F-NAT — FORMANT liest aus der Datei 'ftext.nat'.

Beide Dateien sind im Dateiverzeichnis /usr/lib/formant. Standardmäßig enthält:

ftext.std	englische Fehlertexte.
ftext.nat	deutsche Fehlertexte.

Den Inhalt der Fehlertextdateien können Sie ändern. Die Fehlernummern müssen unverändert bleiben. Die Texte können Sie z.B. in eine andere Sprache übersetzen.

Beispiel:

```
WORKING-STORAGE SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORSP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.
  .
  .
    move f-nat to n-fehlertexte.
```

```
move "N-MSP" to f-aufruf.

call f-aufruf using n-main-par, n-session-par
                on overflow
                move "OVERFLOW" to fehlerart
                perform fehler.

if n-ret-code not = n-noerr
                perform fehler.

cancel f-aufruf.
.
.
```

Der Aufruf bewirkt, daß Fehlertexte aus der Datei /usr/lib/formant/ftext.nat genommen werden (standardmäßig deutsch).

Sitzung eröffnen - open session

N-OPS eröffnet eine FORMANT-Sitzung. Für die Sitzung teilt FORMANT einen Sitzungs-Deskriptor zu. Über den Sitzungs-Deskriptor identifiziert FORMANT alle Datenbereiche dieser Sitzung. N-OPS setzt die Standardwerte für die Sitzungs-Parameter. Es muß der erste FORMANT-Aufruf im Programm sein und darf erst wieder aufgerufen werden, wenn die Sitzung geschlossen wurde.

CALL "N-OPS" USING N-MAIN-PAR.

Parameter:

01	N-MAIN-PAR.	enthalten im COPY-Element FORGP.		
←	05	N-SESSION-ID	PIC 99.	
→	05	N-VERSION	PIC 999	VALUE 100.
→	05	N-INIT	PIC X	VALUE "N".
←	05	N-RET-CODE	PIC S99999.	

Erklärung der Parameter

N-SESSION-ID — Sitzungs-Deskriptor. N-OPS liefert diesen Wert zurück. Er wird in allen weiteren FORMANT-Aufrufen gebraucht.

N-INIT — Lade-Byte: N Funktion ausführen. J Funktion nur laden.

N-RET-CODE — Ergebnis des Aufrufs.

N-ENSA	-4	Kein Speicherplatz verfügbar.
N-ENSD	-2	Maximale Sitzungsanzahl überschritten.
N-NOERR	0	Aufruf erfolgreich.
N-ENOP	1	Fehlerdatei nicht ladbar.
N-ENOF	4	Felder für Fehlertext und Status nicht anlegbar.
N-ENVS	1024	Falsche FORMANT-Version gewünscht.
N-ESEQ	8192	Falsche Reihenfolge: z.B. N-OPS schon durchlaufen oder Aufruf in Benutzerroutine.

N-VERSION — FORMANT-Version. Anzugeben ist die FORMANT-Version mit der gearbeitet werden soll:

100 für Version 1.0 101 für Version 1.01 usw.

Ist die tatsächliche Versionsnummer kleiner als die hier angegebene, kehrt FORMANT mit ENVS zurück. Damit wird vermieden, daß Programme mit Funktionen neuerer FORMANT-Versionen mit einer alten FORMANT-Version laufen.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.

  .
    move "N-OPS" to f-aufruf.

    call f-aufruf using n-main-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
  .
```

Der Aufruf eröffnet die FORMANT-Sitzung. Der Sitzungs-Deskriptor steht jetzt im Feld n-session-id (in n-main-par).

Maske ausdrucken - print presentation image

Der Aufruf druckt die aktuelle Maske auf dem Drucker aus. Alle aktuellen Feldinhalte werden ausgedruckt, auch die der variablen Felder. Felder mit der Eigenschaft DUNKEL druckt N-PPI nicht aus. Alle anderen Darstellungseigenschaften berücksichtigt N-PPI nicht.

```
CALL "N-PPI" USING N-MAIN-PAR.
```

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→  05  N-SESSION-ID          PIC 99.
→  05  N-INIT                PIC X          VALUE "N".
←  05  N-RET-CODE            PIC S99999.
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSA -4	Kein Speicherplatz verfügbar.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ESEQ 8192	Falsche Reihenfolge: z.B. kein N-OPS oder kein N-CPI.

Hinweis:

n_ppi erzeugt eine temporäre Datei. Diese wird standardmäßig mit lpr ausgedruckt und anschließend gelöscht.

Dateiname: _ppi_#.tmp (# von 0 bis 9).

lpr wird mit dem Schalter -pb3 aufgerufen. Das bedeutet, daß die Zeichenbreite auf 17 Zeichen/Zoll eingestellt wird und in einer Zeile max. 136 Zeichen möglich sind.

Die Standardverarbeitung mit dem Kommando lpr läßt sich abändern. In der Shell-Variablen FOPRT können Sie ein anderes Kommando vereinbaren, mit dem die temporäre Datei verarbeitet wird.

Beispiele für die Variable FOPRT:

FOPRT=lpr vor dem Programmaufruf bewirkt, daß Formate in der normalen Zeichenbreite ausgedruckt werden.

FOPRT=prog bewirkt, daß die temporäre Datei an das (benutzereigene) Programm 'prog' übergeben wird.

FOPRT ist zu exportieren (Kommando export FOPRT).

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
PROCEDURE DIVISION.
  .
  .
  move "N-PPI" to f-aufruf.

  call f-aufruf using n-main-par
                on overflow
                move "OVERFLOW" to fehlerart
                perform fehler.

  if n-ret-code not = n-noerr
                perform fehler.

  cancel f-aufruf.
  .
  .
```

Der Aufruf druckt die aktuelle Maske mit allen Feldinhalten auf dem Drucker aus.

Daten einlesen - read

Der Aufruf wirkt wie der Aufruf 'write read' ohne den Teil 'Schreiben'.

Lesen: FORMANT erwartet die Eingabe in die Datenfelder der Maske. FORMANT prüft und verarbeitet die Eingabe entsprechend den aktuell definierten Bedingungen (siehe Abschnitt 2.3).

Anschließend erhält das Programm die Steuerung zurück.

In der Version 1.0 sind 'read' und 'write read' identisch.

```
CALL "N-RED" USING N-MAIN-PAR.
```

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-SYS             PIC 999.
←    05  N-AKT-FELD            PIC X(8).
←    05  N-RET-CODE            PIC S99999.
```

Erklärung der Parameter

N-SESSION-ID — Sitzungs-Deskriptor (siehe N-OPS).

N-INIT — Lade-Byte: N Funktion ausführen. J Funktion nur laden.

N-RET-SYS — Zeiger auf den logischen Code der Systemsteuertaste. Dieser Wert gibt an, wodurch die Eingabe der Daten beendet wurde. Mögliche Werte siehe Aufruf 'write read'.

N-AKT-FELD — Name des aktuellen Feldes. Dies ist dasjenige Feld der Maske, das FORMANT als letztes bearbeitet hat.

N-RET-CODE — Ergebnis des Aufrufs.

N-ERR -32 — Ablauffehler: ein früher aufgetretener Fehler wurde erkannt.

N-ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR	0	Aufruf erfolgreich.
N-ENOF	4	Ablauffehler: ein früher aufgetretener Fehler wurde erkannt.
N-APL	8	Unterbrechung durch Benutzerroutine.
N-EPOS	64	Positionierungsfehler: ein früher aufgetretener Fehler wurde erkannt.
N-FFELD	256	Erstes Feld in der Bearbeitungsreihenfolge erreicht (nur bei feldweisem Betrieb).
N-ESEQ	8192	Falsche Reihenfolge: z.B. kein N-OPS oder kein N-CPI oder Aufruf in Benutzerroutine.

Es gelten dieselben Hinweise, wie beim Aufruf 'write read'.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
PROCEDURE DIVISION.
  .
  .
    move "N-RED" to f-aufruf.

    call f-aufruf using n-main-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
  .
```

Sie können diesen Aufruf anstelle des Aufrufs 'write read' einsetzen. In V1.0 ist die Wirkung identisch. Später müßte vor dem Aufruf 'read' noch ein Aufruf 'write' eingefügt werden.

Umgebungsparameter abfragen - request environment parameters

Der Aufruf liefert den aktuellen Wert der Umgebungsparameter.

CALL "N-REP" USING N-MAIN-PAR, N-ENVIRON-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-ENVIRON-PAR.       enthalten im COPY-Element FOREP.
←    05  N-FUELLZEICHEN       PIC X          VALUE " ".
←    05  N-HINTERGRUND        PIC X          VALUE "N".
←    05  N-CONTROL            PIC X          VALUE "S".
←    05  N-FELDLEERZ          PIC X          VALUE "_".
←    05  N-VOR-AUFB           PIC X          VALUE "J".
←    05  N-RUECK-AUFB         PIC X          VALUE "J".
←    05  N-LOESCHLINKS        PIC X          VALUE "N".
←    05  N-AUTODUPVORF        PIC X          VALUE "N".
←    05  N-STRTERM            PIC X          VALUE "N".
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Erfolgreicher Aufruf.
N-ESEQ 8192	Falsche Reihenfolge: z.B. kein N-OPS.

Umgebungsparameter

Die ausführlichen Erläuterungen finden Sie beim Aufruf 'modifiy environment parameters'.

Parameter	Bedeutung	Standard	weitere Werte
N-FUELLZEICHEN	Bildschirmfüllzeichen	' '	alle abdruck-baren Zeichen
N-HINTERGRUND	Hintergrunddarstellung	NORM	INV
N-CONTROL	Eingabereaktion	SATZ	FELD
N-FELDLEERZ	Feldleerzeichen	"_"	alle abdruck-baren Zeichen
N-VOR-AUFB	Feld aufbereiten beim Überspringen	JA	NEIN
N-RUECK-AUFB	Feld aufbereiten bei Rückpositionierung	JA	NEIN
N-LOESCHLINKS	zeichenweise Rückposit.	NEIN	JA
N-AUTODUPVORF	Automatisches Duplizieren in vorbelegten Feldern.	NEIN	JA
N-STRTERM	Stringterminator	NEIN	JA

Beispiel:

Dieser Aufruf hat in Version 1.0 keine Bedeutung, da nur eine Sitzung erlaubt ist. Die aktuellen Parameter stehen immer im Bereich n-environ-par zur Verfügung.

Eingabeparameter abfragen - request operating mode

Der Aufruf liefert den aktuellen Wert der Eingabeparameter. In Version 1.0 ist dies nur Gegentastprüfen ein oder aus.

CALL "N-ROM" USING N-MAIN-PAR, N-OP-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→  05  N-SESSION-ID          PIC 99.
→  05  N-INIT                PIC X              VALUE "N".
←  05  N-RET-CODE            PIC S99999.

01   N-OP-PAR.            enthalten im COPY-Element FORCP.
→  05  N-GEGENTASTP        PIC X              VALUE "N".
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ENOP 1	Falscher Parameter.
N-GEGENTASTP	Gegentastprüfen.
AUS	Gegentastprüfen aus.
EIN	Gegentastprüfen ein.

Beispiel:

```
WORKING-STORAGE SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORCP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".
```

```
.
.
PROCEDURE DIVISION.
.
.
  move "N-ROM" to f-aufruf.

  call f-aufruf using n-main-par, n-op-par
                on overflow
                move "OVERFLOW" to fehlerart
                perform fehler.
  if n-ret-code not = n-noerr
                perform fehler.

  cancel f-aufruf.
.
.
```

Der Aufruf liefert in n-op-par den Wert JA oder NEIN, jenachdem ob der Prüfmodus ein- oder ausgeschaltet ist.

Sitzungsparameter abfragen - request session parameters

Der Aufruf liefert den aktuellen Wert der Sitzungsparameter.

CALL "N-RSP" USING N-MAIN-PAR, N-SESSION-PAR.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-CODE            PIC S99999.

01   N-SESSION-PAR.       enthalten im Copy-Element FORSP.
←    05  N-GERAETEKLASSE     PIC X       VALUE "T".
←    05  N-GERAETETYP        PIC X(20)   VALUE "fo97801".
←    05  N-BERECHTIGUNG      PIC X       VALUE "B".
←    05  N-FEHLERTEXTE       PIC X       VALUE "D".
```

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Erfolgreicher Aufruf.
N-ESEQ 8192	Falsche Reihenfolge: z.B. kein N-OPS.

Sitzungsparameter

Die ausführlichen Erläuterungen finden Sie beim Aufruf 'modifiy session parameters'.

Parameter	Bedeutung	Standard	weitere Werte
N-GERAETEKLASSE	Geräteklasse	TERM	DRUCKER IN-PER
N-GERAETETYP	Gerätetyp	"97801"	"9001" "9004"
N-BERECHTIGUNG	möglicher Zugriff	BEIDES	LESEN SCHREIBEN
N-FEHLERTEXTE	Fehlerdatei	F-STD	F-NAT

Beispiel:

Dieser Aufruf hat in Version 1.0 keine Bedeutung, da nur eine Sitzung erlaubt ist. Die aktuellen Parameter stehen immer im Bereich n-session-par zur Verfügung.

Signalton erzeugen - set beeper

Der Aufruf erzeugt einen Signalton, dessen Länge von der angegebenen Zahl abhängt. Während des Signaltones akzeptiert FORMANT keine Eingabe.

CALL "N-SBE" USING N-MAIN-PAR, anzahl.

Parameter:

01 N-MAIN-PAR. enthalten im COPY-Element FORGP.
→ 05 N-SESSION-ID PIC 99.
→ 05 N-INIT PIC X VALUE "N".
← 05 N-RET-CODE PIC S99999.

anzahl
→ Datenfeld PIC 999 oder Literal.

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
anzahl	Zahlenwert von 1 bis 128: Anzahl der auslösenden Steuerzeichen. Diese bestimmt die Länge des Signaltones.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  77 anz                  pic 999.
```

```
.
.
PROCEDURE DIVISION.
  .
  .
    move "N-SBE" to f-aufruf.

    call f-aufruf using n-main-par, anz
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.
    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
  .
```

Der Aufruf erzeugt ein Signal, dessen Länge durch die Variable anz bestimmt wird (höchstens 128).

Dupliziersatz setzen - set dup record

Der Aufruf übernimmt den angegebenen Bereich als Dupliziersatz und schaltet die Duplizierfunktion ein. Dupliziert werden können dann:

- mit der Dupliziertaste beliebige Felder,
- automatisch alle Felder, die die Feldeigenschaft Duplizierfeld haben.

Zum Duplizieren siehe auch Abschnitt 2.3.7. Die Duplizierfunktion ausschalten können Sie, indem Sie als Dupliziersatz SPACES angeben.

Sie müssen eine Maske angelegt haben, bevor Sie 'set dup record' aufrufen.

CALL "N-SDR" USING N-MAIN-PAR, dupsatz.

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
→    05  N-NDSL                PIC 9999.
←    05  N-RET-CODE            PIC S99999.
```

dupsatz
→ Datenbereich mit Feldern vom Typ PIC X und PIC 9 oder Literal.

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).	
N-INIT	Lade-Byte:	N Funktion ausführen. J Funktion nur laden.
N-NDSL	Länge des Dupliziersatzes.	
N-RET-CODE	Ergebnis des Aufrufs.	
N-ENSA -4	Kein Speicherplatz vorhanden.	
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.	
N-NOERR 0	Aufruf erfolgreich.	
N-ENUP 2	Versuch, Duplizierfunktion auszuschalten, obwohl sie nicht eingeschaltet war.	

→ dupsatz

Dupliziersatz. Der Satz muß so strukturiert sein, wie er mit 'get record' aufgebaut wurde, das heißt die Reihenfolge der Felder im Datensatz muß mit der definierten Reihenfolge übereinstimmen. Da mit der Dupliziertaste beliebige Felder dupliziert werden können, sollten Sie den Dupliziersatz vollständig versorgen.

Der Dupliziersatz darf keine Zeichen enthalten, deren Eingabe syntaktisch nicht erlaubt ist. Sie müssen eventuell Datenfüllzeichen im Satz z.B. in Leerzeichen umwandeln (= Bildschirmfüllzeichen).

Wenn Sie SPACES angeben, wird die Duplizierfunktion ausgeschaltet.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".

   01 dup-satz.
      05 d-name                    pic x(11).
      05 d-vorname                 pic x(11).
      05 d-strasse                 pic x(12).
      05 d-ort                     pic x(21).
      05 d-telefon                 pic x(16).
      05 d-jahre                   pic 99.
      05 d-groesse                 pic 999.
  .
  .
PROCEDURE DIVISION.
  .
  .
    move 76 to n-ndsl.
    move "N-SDR" to f-aufruf.

    call f-aufruf using n-main-par, dup-satz
                 on overflow
                 move "OVERFLOW" to fehlerart
```

```
                    perform fehler.

    if n-ret-code not = n-noerr
                    perform fehler.

    cancel f-aufruf.
.
.
```

Der Aufruf bewirkt, daß beim nächsten Lesen Felder der Maske dupliziert werden können. Der Feldinhalt wird aus dup-satz entnommen. Felder mit der Feldeigenschaft 'Duplizierfeld' werden automatisch dupliziert. Das heißt, FORMANT fordert sie nicht zur Eingabe an, sondern füllt sie sofort mit dem entsprechenden Inhalt aus dup-satz.

Schreibmarke positionieren - set cursor

Der Aufruf positioniert die Schreibmarke auf eine beliebige Stelle innerhalb der Datenfelder der Maske.

CALL "N-SEC" USING N-MAIN-PAR, N-FIELD-PAR, offset.

Parameter:

```
01   N-MAIN-PAR.        enthalten im COPY-Element FORGP.
→  05  N-SESSION-ID          PIC 99.
→  05  N-INIT                PIC X          VALUE "N".
←  05  N-RET-CODE            PIC S99999.

01   N-FIELD-PAR.       enthalten im COPY-Element FORFP.
→  05  N-NAME              PIC X(8).
```

offset
→ Datenfeld PIC 99 oder Literal.

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.
N-ENOF 4	Feldname nicht bekannt.
N-ENOF1 32	N-NAME ist vom Typ FIX.
N-NAME	Name des Feldes, in das die Schreibmarke zu positionieren ist. Bei der Angabe SPACES nimmt FORMANT das aktuelle Feld.
→ offset	Datenfeld (PIC 99) für die Position der Schreibmarke innerhalb des Feldes. Feldanfang ist 0. Bei ungültiger Angabe wird 0 angenommen.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORFP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".

77 o-set                pic 99.
  .
  .

PROCEDURE DIVISION.
  .
  .
  move "m-ort" to n-name.
  move 5 to o-set.
  move "N-SEC" to f-aufruf.

  call f-aufruf using n-main-par, n-field-par, o-set
               on overflow
               move "OVERFLOW" to fehlerart
               perform fehler.

  if n-ret-code not = n-noerr
               perform fehler.

  cancel f-aufruf.
  .
  .
```

Der Aufruf positioniert die Schreibmarke in das Feld m-ort auf das sechste Zeichen.

Maske abspeichern - save presentation image

Der Aufruf speichert die aktuelle Maske in einer Datei ab. Die Datei steht im aktuellen Dateiverzeichnis und erhält den angegebenen Namen der Maske. Unter diesem Namen können Sie die Maske wieder laden, wenn Sie N_CPI aufrufen. N-SPI speichert alle aktuellen Feldinhalte ab, auch die der variablen Felder.

```
CALL "N-SPI" USING N-MAIN-PAR, N-PI-PAR.
```

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→  05  N-SESSION-ID            PIC 99.
→  05  N-INIT                  PIC X              VALUE "N".
←  05  N-RET-CODE              PIC S99999.

01   N-PI-PAR.            enthalten im COPY-Element FORGP.
→  05  N-PI-NAME             PIC X(8).
```

Erklärung der Parameter

N-SESSION-ID		Sitzungs-Deskriptor (siehe N-OPS).
N-INIT		Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE		Ergebnis des Aufrufs.
N-ENSA	-4	Kein Speicherplatz vorhanden.
N-ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR	0	Aufruf erfolgreich.
N-ENOF	4	Kein Feld vorhanden.
N-ESAV	512	Abspeichern nicht erfolgreich.
N-ENFL	4096	Fehler in der Maske.
N-ESEQ	8192	Falsche Reihenfolge: z.B. kein N-OPS oder kein N-CPI.
N-PI-NAME		Name, unter dem die Maske abgelegt wird. Der Name ist gleich dem Namen der Datei für die Maske. Eine bestehende Datei dieses Namens wird überschrieben.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .

PROCEDURE DIVISION.
  .
  .

    move "m-save" to n-pi-name.
    move "N-SPI" to f-aufruf.

    call f-aufruf using n-main-par, n-pi-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
  .
```

Der Aufruf speichert die aktuelle Maske in der Datei m-save ab. Sie können diese Maske z.B. später mit 'create presentation image' erneut laden (einschließlich der aktuellen Feldinhalte).

Maske ausgeben und Daten einlesen - write read

Der Aufruf ist eine Kombination der Aufrufe 'write' und 'read'. Er bewirkt nacheinander:

1. Schreiben: Die Maske wird entsprechend den aktuellen Parametern und Feldinhalten ausgegeben bzw. abgeändert (Funktion von 'write'). Diese Funktion ist in Version 1.0 unwirksam (siehe 'write').

2. Lesen: FORMANT erwartet die Eingabe in die Datenfelder der Maske. FORMANT prüft und verarbeitet die Eingabe entsprechend den aktuell definierten Bedingungen (siehe Abschnitt 2.3). Das ist die Funktion von 'read'.

Anschließend erhält das Programm die Steuerung zurück.

```
CALL "N-WIR" USING N-MAIN-PAR.
```

Parameter:

```
01   N-MAIN-PAR.          enthalten im COPY-Element FORGP.
→    05  N-SESSION-ID          PIC 99.
→    05  N-INIT                PIC X          VALUE "N".
←    05  N-RET-SYS             PIC 999.
←    05  N-AKT-FELD            PIC X(8).
←    05  N-RET-CODE            PIC S99999.
```

Erklärung der Parameter

N-SESSION-ID — Sitzungs-Deskriptor (siehe N-OPS).

N-INIT — Lade-Byte: N Funktion ausführen. J Funktion nur laden.

N-RET-SYS — Zeiger auf den logischen Code der Systemsteuertaste. Dieser Wert gibt an, wodurch die Eingabe der Daten beendet wurde. Folgende Werte sind möglich:

Wert	Bedeutung	Taste
N-AUSG	Feld bzw. letztes Feld der Maske ausgefüllt, je nach Betriebsart 'feld-	

	weise' oder 'satzweise' (siehe Abschnitt 2.3).	
N-FEPLUS	Positives Feldende.	FE+
N-FEMINUS	Negatives Feldende.	FE-
N-HOME	Sprung auf erstes Feld.	HOME
N-NEXT	Sprung auf nächste Zeile.	NEXT
N-LAST	Sprung auf vorige Zeile.	LAST
N-BACKWARDS	Sprung auf voriges Feld.	BACKWARDS
N-FORWARDS	Sprung auf nächstes Feld.	FORWARDS
N-LVD	Lösche Datenfelder.	LVD
N-SEND	Satzende.	SEND
N-PRINT	akt. Maske drucken	PRINT
N-F1	Funktionstaste.	F1
.		
N-F27	Funktionstaste.	F27
N-MENU	Funktionstaste.	MENU
N-HELP	Funktionstaste.	HELP
N-START	Funktionstaste.	START
N-END	Funktionstaste.	END
N-CARDBAD	Ausweis konnte nicht gelesen werden.	
N-CARDOUT	Ausweis wurde aus dem Leser entnommen.	

Zur Wirkung der einzelnen Systemsteuertasten siehe Abschnitt 2.3.

N-AKT-FELD — Name des aktuellen Feldes. Dies ist dasjenige Feld der Maske, das FORMANT als letztes bearbeitet hat.

N-RET-CODE — Ergebnis des Aufrufs.

N-ERR -32	Ablauffehler: ein früher aufgetretener Fehler wurde erkannt.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.

N-ENOF	4	Ablauffehler: ein früher aufgetretener Fehler wurde erkannt.
N-APL	8	Unterbrechung durch Benutzerroutine.
N-EPOS	64	Positionierungsfehler: ein früher aufgetretener Fehler wurde erkannt.
N-FFELD	256	Erstes Feld in der Bearbeitungsreihenfolge erreicht (nur bei feldweisem Betrieb).
N-ESEQ	8192	Falsche Reihenfolge: z.B. kein N-OPS oder kein N-CPI oder Aufruf in Benutzerroutine.

Hinweise:

- Die Steuerung geht an das Programm zurück, wenn
  - ein Feld fertig ausgefüllt ist (bei feldweisem Betrieb). Das erkennen Sie am logischen Code N-AUSG.
  - das letzte Feld der Maske ausgefüllt ist (bei satzweisem Betrieb). Das erkennen Sie ebenfalls am logischen Code N-AUSG.
  - eine Systemsteuertaste gedrückt wurde. Das erkennen Sie an einem logischen Code ungleich N-AUSG (siehe auch Abschnitt 4.2.1 und Aufruf N-MIR).
  - eine Benutzerroutine mit dem Rückgabewert E-APL verlassen wurde (siehe Kapitel7). Das erkennen Sie am Ergebnis N-APL.
- Wurde bei feldweisem Betrieb das letzte Feld der Maske ausgefüllt, liefert der Aufruf das Ergebnis N-FFELD.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .

PROCEDURE DIVISION.
  .
  .
    move "N-WIR" to f-aufruf.
```

```
call f-aufruf using n-main-par
               on overflow
               move "OVERFLOW" to fehlerart
               perform fehler.

if n-ret-code not = n-noerr
               perform fehler.

cancel f-aufruf.
.
.
```

Der Aufruf fordert die Eingabe in die Maske an. Anschließend kann man den logischen Code der Systemsteuertaste abfragen (siehe auch ausführliches Beispiel in Abschnitt 4.1.2.

Maske ausgeben - write

Der Aufruf ist in V1.0 unwirksam. Alle Aufrufe, die eine Veränderung der Maske bewirken, wirken in dieser Version direkt, z.B. 'change record' usw. In späteren Versionen wirkt kein Aufruf mehr direkt auf den Bildschirm, sondern erst bei einem Schreibaufruf: 'write' oder 'write_read'.

Zukünftige Funktion:

Schreiben: Die Maske wird entsprechend den aktuellen Parametern und Feldinhalten ausgegeben bzw. abgeändert. Anschließend erhält das Programm die Steuerung zurück.

CALL "N-WRT" USING N-MAIN-PAR.

Parameter:

01	N-MAIN-PAR.	enthalten im COPY-Element FORGP.		
→	05	N-SESSION-ID	PIC 99.	
→	05	N-INIT	PIC X	VALUE "N".
←	05	N-RET-CODE	PIC S99999.	

Erklärung der Parameter

N-SESSION-ID	Sitzungs-Deskriptor (siehe N-OPS).
N-INIT	Lade-Byte: N Funktion ausführen. J Funktion nur laden.
N-RET-CODE	Ergebnis des Aufrufs.
N-ENSD -2	Sitzungs-Deskriptor nicht bekannt.
N-NOERR 0	Aufruf erfolgreich.

Beispiel:

```
WORKING-STORAGE-SECTION.
  .
  .
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  .
  .
```

```
PROCEDURE DIVISION.
  .
  .
    move "N-WRT" to f-aufruf.

    call f-aufruf using n-main-par
                 on overflow
                 move "OVERFLOW" to fehlerart
                 perform fehler.

    if n-ret-code not = n-noerr
                 perform fehler.

    cancel f-aufruf.
  .
  .
```

6 Anwendungsprogramme in C

Für C-Programme stellt FORMANT einen Satz von Funktionen bereit. Alle FORMANT-Funktionen werden in das ablauffähige Programm mit eingebunden.

6.1 Programmrahmen eines C-Programms

Ein C-Programm mit FORMANT-Aufrufen hat folgende Form:

```
include <formant.h>
  .
  .
main  {
  .
  .
int version = 100;     /* FORMANT-Version 1.00 */
int sd;                /* Sitzungs-Deskriptor */
int ret;               /* Rückgabewert */
  .
  .   weitere Definitionen für FORMANT
  .
  .
ret = n_ops (version,&sd);            /* Sitzung eröffnen */
  .
  .
  .   weitere FORMANT-Aufrufe
  .
  .
ret = n_cls (sd);                     /* Sitzung schließen */
  .
  .
}
```

6.2 Parameterübergabe

Einfache Variable:

Eingangsparameter, die das Programm liefern muß, werden an FORMANT als Werte übergeben. Der Aufruf kann diese Variable nicht verändern.

Ergebnisparameter, die FORMANT zurückliefert, werden als Adressen übergeben. Die Variable, deren Adresse Sie übergeben, muß immer im Programm selbst definiert sein.

Zeichenfolgen (char-Felder):

Zeichenfolgen werden immer als Adressen übergeben (Zeiger auf das erste Zeichen, Ende der Zeichenfolge bei \0). Das ist unabhängig davon, ob eine Zeichenfolge ein Eingabe- oder ein Ergebnisparameter ist.

Zeichenfolgen verwendet FORMANT z.B. bei Datensätzen, Feldnamen usw.

Grundregel:

Alle Parameterbereiche mußsen Sie im Programm zur Verfügung stellen. Bei Ergebnisparametern übergeben Sie FORMANT die Adresse. FORMANT trägt an dieser Adresse den gelieferten Wert ein.

Wahlfreie Angaben

Bei jedem Funktionsaufruf müssen Sie für alle Parameter Werte angeben. Trifft in einem Fall keine Wahlmöglichkeit zu, geben Sie als Wert NULL an. Beispiel: Feldeigenschaften beim Aufruf n_inf.

Parameterwerte

Für die Parameterwerte verwenden Sie die Definitionen aus der Include-Datei /usr/include/formant.h.

Diese Include Datei müssen Sie immer angeben. Es ist die einzige, die Sie für FORMANT benötigen. Ihr Inhalt ist in Abschnitt 6.4 aufgelistet. Alle Angaben in der Beschreibung der Aufrufe beziehen sich auf die darin enthaltenen Definitionen.

Verhalten bei fehlerhaften Angaben

FORMANT prüft übergebene Parameter nur auf den zulässigen Wertebereich. Anzahl und Reihenfolge der Parameter kann FORMANT nicht

überprüfen. Liegt eine Angabe außerhalb des zulässigen Bereiches, gibt es zwei Möglichkeiten:

- Das Ergebnis ist > NOERR. Dann hat FORMANT einen Standardwert gesetzt.
- Das Ergebnis ist < NOERR. Dann konnte FORMANT den Wert nicht korrigieren. Die Funktion liefert kein sinnvolles Ergebnis.

Werden Adressen (Zeiger) übergeben, kann FORMANT nicht überprüfen, ob die Angabe sinnvoll ist. Fehlerhafte Angaben führen hier meist zum Absturz oder zumindest zu sehr merkwürdigem Verhalten des Programms.

6.3 Ergebniswerte

Alle FORMANT-Funktionen liefern ein Ergebnis vom Typ int zurück. Bei jeder Funktion sind die möglichen Werte mit ihrer Bedeutung angegeben. Alle Werte zusammengefaßt finden Sie in formant.h unter 'Fehlerfälle'.

FORMANT unterscheidet:

Ergebniswert = NOERR (bzw. 0)
: Der Aufruf ist erfolgreich abgelaufen.

Ergebniswert > NOERR (bzw. 0)
: Leichter Fehler. Die Sitzung kann u.U. ohne weitere Maßnahmen fortgesetzt werden, weil der Fehler ohne Einfluß auf den Ablauf weiterer Aufrufe ist. Jedoch ist das Ergebnis dieses Aufrufs meist nicht das, was Sie wollten.

Ergebniswert < NOERR (bzw. 0)
: Schwerer Fehler. Weitere FORMANT-Aufrufe könnten nicht laufen, z.B wenn bei create p.i. ein falscher Maskenname angegeben wurde. Sie müssen den Fehler im Programm behandeln oder das Programm abbrechen.

6.4 Definitionen für Parameterwerte - formant.h

Die Include-Datei formant.h liegt im Dateiverzeichnis /usr/include. Sie hat folgenden Inhalt:

```
/*====================        Allgemeines        ========================*/

#define         ABFRAGEN        1

#define         GERAETEKLASSE   1

#define         TERMINAL        "\001"
#define         DRUCKER         "\002"
#define         IN_PER          "\004"        /* Interaktive Peripherie */

#define         GERAETETYP      2

#define         STAN            "fo97801"     /* Standard */


#define         BERECHTIGUNG    4

#define         LESEN           "\001"
#define         SCHREIBEN       "\002"
#define         BEIDES          "\004"


#define         FEHLERTEXTE     8

#define         F_STD           "\001"        /* Fehlertext Standardbel. */
#define         F_NAT           "\002"        /* Fehlertext nationale Bel. */

/*=================          Enviromentparameter          ==================*/


#define         FUELLZEICHEN    1             /*  Des Schirmes  */
#define         HINTERGRUND     2             /*  Des Schirmes  */
#define         BILDAUFBAU      4             /*  Fest / Fliessend */
#define         FORMANTCONTROL  8             /*  Satz / Feld  */
#define         FELDLEERZ       16            /*  Feldleerzeichen  */
#define         VOR_AUFB        32            /*  Feldaufbereitung */
#define         RUECK_AUFB      64            /*  Feldaufbereitung */
#define         LOESCHLINKS     128           /*  bei <- Zeichen loeschen */
#define         AUTODUP         256           /*  Autoduplizieren ein     */
#define         STRTERM         1024          /*  Felder im Satz  */

#define         BLANK           ' '

#define         UNTERS          '_'

/*              INVERS          1       */
#define         NORMAL          2

#define         FEST            1
#define         FLIESSEND       2

#define         SATZ            1
#define         FELD            2

#define         JA              1
#define         NEIN            2
```

```
#define         JEIN            4

#define         STD             1           /* Standardbehandlung  */
#define         IGN             2           /* Ignoriere Funktionstaste */
#define         USR             4           /* Benutzerspez. Behandlung */
#define         APL             8           /* Zurueck zur Anwendung */

#define         K1              10

/*==================  Operating Control    ================================*/

#define         GEGENTASTP      1

#define         AUS             1
#define         EIN             2


/*==================       PI  Control     ================================*/


#define         NEU             1
#define         VORHANDEN       2


/*================             Fehlerfaelle              ================*/

#define         NOERR           0           /* no error  */

/*================          Nicht behebare Fehler        ================*/

#define         ERFA            -1          /* fatal error  */
#define         ENSD            -2          /* no session deskriptor  */
#define         ENSA            -4          /* no space available  */
#define         ENFO            -8          /* no FORMANT map */
#define         ENSY            -16         /* no system control key */
#define         ERR             -32         /* error not identified   */

/*================           Behebare Fehler            ================*/

#define         ENOP             1          /* no parameter value */
#define         ENUP             2          /* NULL pointer */
#define         ENOF             4          /* no field name */
/*  Hier wird APL zurueckgeliefert -> 8  */  /* aplication reaction */
#define         ENOF2            16         /* no field name for record */
#define         ENOF1            32         /* no field name for processng*/
#define         EPOS             64         /* wrong position */
#define         ETTL             128        /* text to long */
#define         FFELD            256        /* first field reached */
#define         ESAV             512        /* saving denied  */
#define         ENVS             1024       /* no correct version */
#define         EMDA3            2048       /* more data available */
#define         ENFL             4096       /* no FORMANT file */
#define         ESEQ             8192       /* sequence error */


/*================             Feldattribute              ================*/

#define         DARSTELLUNG     1
#define         ZEICHENSATZ     2
#define         BEARBEITUNG     4
#define         FELDATT         8
#define         NACHKOMMA       16
#define         FUELLFELD       32
#define         FUELLDATEN      64
```

```
#define         FELDTYP         128

/*================          Darstellungsattribut          ================*/

#define         INVERS          1
#define         BLINKEND        2
#define         HELL            4
#define         HALBHELL        8
#define         DUNKEL          16
#define         UNTERSTRICHEN   32

/*================        Farben und Zeichensaetze        ================*/

#define         WEISS           1
#define         ROT             2
#define         GRUEN           4
#define         GELB            8
#define         BLAU            16
#define         STANDARD        0x00
#define         NATIONAL        0x20
#define         MATH            0x40
#define         MOSAICS         0x60
#define         KLAMMERN        0x80
#define         IBM             0xA0


/*================        Bearbeitungsattribut    ========================*/

#define         MUSS            1
#define         GESCHUETZT      2
#define         AUSWEISLESE     4
#define         DUPLIZIER       8
#define         GROSS           16
#define         TRIGGER         32
#define         VOLLSTAENDIG    64
#define         SKIP            128

/*========================       Feldattribut     ============================*/

#define         ALPHA           1
#define         ALPHANUM        2
#define         NUM             4
#define         LINKS           8
#define         VORZ            16
#define         SONDERZ         32
#define         RECHTS          64

/*========================   Nachk. und Prueffeld   ========================*/

#define         KGEGENTAST      16
#define         KORRSPERRE      32
#define         USRF            64

/*========================       Feldtyp          ============================*/
#define         FIX             1
#define         VAR             2
#define         VORB            4
#define         LEER            8

/*========================   Systemsteuerzeichen    ========================*/

#define         AUSG            0
#define         FEPLUS          1
#define         FEMINUS         2
```

```
#define         FEMINUS         2
#define         HOME            3
#define         NEXT            4
#define         LAST            5
#define         BACKWARDS       6
#define         FORWARDS        7
#define         LVD             8
#define         SEND            9
#define         DUP             10
#define         VALID           11
#define         KOR             12
#define         FKOR            13
#define         ANZ             14
#define         BACKSPACE       15
#define         LOEF            16
#define         CUL             17
#define         CUR             18
#define         F1              19              /* ACHTUNG */
#define         F2              20              /*   F1    */
#define         F3              21              /*   bis   */
#define         F4              22              /*   END   */
#define         F5              23              /*  immer  */
#define         F6              24              /*  dicht  */
#define         F7              25              /*  lassen */
#define         F8              26
#define         F9              27
#define         F10             28
#define         F11             29
#define         F12             30
#define         F13             31
#define         F14             32
#define         F15             33
#define         F16             34
#define         F17             35
#define         F18             36
#define         F19             37
#define         F20             38
#define         F21             39
#define         F22             40
#define         F23             41
#define         F24             42
#define         F25             43
#define         F26             44
#define         F27             45
#define         MENU            46
#define         PRINT           47
#define         HELP            48
#define         START           49
#define         END             50
#define         DUMMY           51
#define         CARDBAD         52
#define         CARDOUT         53
#define         SMK             54

#define         MAXSYSTEM       55
```

6.5 Einschränkungen

- Die Schnittstellen stdin und stdout sind sinnvoll nur außerhalb der FORMANT-Sitzung zu verwenden, z.B. zu Programmbeginn und Programmende für Meldungen. FORMANT arbeitet über diese Schnittstellen.
- stderr können Sie verwenden, sollten die Ausgabe aber auf Datei umlenken.
- Namen für externe Variablen oder Funktionen dürfen nicht mit n_ beginnen. Diese Namen sind für FORMANT reserviert.
- Dynamisch angelegter Arbeitsspeicher:
  - Anfordern ist nur mit malloc erlaubt, nicht mit break.
  - Freigeben ist nur mit free erlaubt, nicht mit sbrk.
- - Das NIL-Zeichen (\0) ist als Zeichen nicht erlaubt, da es der Begrenzer für Zeichenfolgen ist. Es ist also kein abdruckbares Zeichen und Sie können es z.B. nicht als Füllzeichen für Datenfelder am Bildschirm verwenden.

6.6 FORMANT-C-Aufrufe

Im folgenden Teil sind alle FORMANT-C-Aufrufe beschrieben. Die Aufrufe sind alphabetisch sortiert. Zu jedem Aufruf finden Sie ein Beispiel, aus dem Sie entnehmen können, welche Datendefinitionen nötig sind und wie Sie die Parameter versorgen.

Die Pfeile bei den Parametern geben an, ob Sie den Parameter versorgen müssen oder ob FORMANT den Wert des Parameters liefert:

→ Ein Eingangsparameter ist vom Programm zu versorgen

← Ein Ergebnisparameter wird vom FORMANT-Aufruf zurückgeliefert.

FORMANT-Aufrufe nach Funktionsgruppen

Sitzungskontrolle

n_ops	Sitzung eröffnen - open session
n_msp	Sitzungsparameter ändern - modify session parameters
n_rsp	Sitzungsparameter abfragen - request session parameters
n_cls	Sitzung schließen - close session

Umgebungskontrolle

n_mep	Umgebungsparameter ändern - modify environment parameters
n_rep	Umgebungsparameter abfragen - request environment parameters
n_mir	Reaktion auf Systemsteuertasten - modify input reaction table
n_due	Benutzerroutinen für Triggerfelder - define user exits

Maskenbehandlung

Maske

n_cpi	Maske anlegen - create presentation image
n_dpi	Maske schließen - destroy presentation image
n_spi	Maske abspeichern - save presentation image
n_ppi	Maske ausdrucken - print presentation image

Datenfelder

n_gef	Datenfeld übernehmen - get field
n_chf	Feldinhalt ändern - change field
n_clf	Feldinhalt löschen - clear field
n_inf	Feld einfügen - insert field
n_def	Feld ausfügen - delete field
n_gfa	Feldeigenschaften abfragen - get field attribute
n_cfa	Feldeigenschaften ändern - change field attribute
n_cft	Feldtyp und Reihenfolge ändern - change field type

Datensatz

n_ger	Datensatz aufbauen - get record

n_chr Datensatz in die Maske übertragen - change record
n_clr Alle Felder der Maske löschen - clear record
n_sdr Dupliziersatz setzen - set dup record

Schreibmarke

n_sec Schreibmarke positionieren - set cursor
n_gec Schreibmarkenposition abfragen - get cursor

Meldungen

n_cem Fehlermeldung ausgeben - change error message
n_csm Statusmeldung ausgeben - change status message
n_sbe Signalton erzeugen - set beeper

Datentransport

n_wrt Maske ausgeben - write
n_wir Maske ausgeben und Daten einlesen - write read
n_red Daten einlesen - read

Eingabeparameter

n_mom Eingabeparameter ändern - modify operating mode
n_rom Eingabeparameter abfragen - request operating mode

Fehlermeldung ausgeben - change error message

Mit dem Aufruf können Sie eigene Fehlermeldungen ausgeben. Die Meldung erscheint in der Systemzeile von Spalte 1 bis Spalte 40. FORMANT sperrt die Tastatur und erzeugt einen Signalton. Nach einer Fehlermeldung kann erst wieder eingegeben werden, wenn die Taste [WORD] (ERS) oder [q] gedrückt wurde. Drücken einer dieser Tasten löscht gleichzeitig den Fehlertext.

n_cem(sd,errormes)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *errormes	Text der Fehlermeldung. Der Text darf bis zu 40 Zeichen lang sein. Ist er länger, wird der Rest abgeschnitten. Ist er kürzer, füllt FORMANT mit Leerzeichen auf.

Ergebnis:

ENSD	-2	Session Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf NULL.

Hinweis:
Der Text der Fehlermeldung wird überschrieben, wenn gleich danach ein zweiter Aufruf n_cem folgt.

Beispiel:

```
   .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
   .
ret = n_cem (sd,"Satz nicht gefunden");
teste ("n_cem",ret);
   .
```

Der Text 'Satz nicht gefunden' erscheint in der untersten Bildschirmzeile als Fehlermeldung.

Feldeigenschaften ändern - change field attribute

Der Aufruf ändert für ein Feld den Wert von Feldeigenschaften (Attributen). Pro Feld ist ein Aufruf nötig.

Alle Feldeigenschaften sind in Abschnitt 3.4 erklärt.

n_cfa(sd,fieldname,attribute,value)

Parameter:

→ int sd — Sitzungs-Deskriptor.

→ char *fieldname — Name des Feldes, dessen Feldeigenschaften Sie ändern wollen.

→ int attribute — Feldeigenschaft. Sie können eine der folgenden Angaben machen. Die Bedeutung dieser Angaben und die möglichen Werte finden Sie beim Aufruf 'insert field'.

DARSTELLUNG	at1	Felddarstellung am Bildschirm Farbe des Feldes (nicht in V1.00)
ZEICHENSATZ	at2	Zeichensatz
BEARBEITUNG	at3	Bearbeitungseigenschaften
FELDATT	at4	Feldausrichtung zulässige Eingabezeichen
NACHKOMMA	at5	Eingabeeigenschaften: Nachkommastellen Prüffeld
FUELLFELD	at6	Feldfüllzeichen
FUELLDATEN	at7	Datenfüllzeichen

→ char value — Wert der Feldeigenschaft. Es sind alle Werte zulässig, wie bei 'insert field beschrieben'.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENOP	1	Falsche Parameterangabe .
ENUP	2	Zeiger auf NULL bei fieldname.
ENOF	4	Feldname nicht bekannt.

Beispiel:

```
   .
int sd;                    /* Sitzungs-Deskriptor */
int ret;                   /* Ergebniswert */
   .
   .
t_rout (taste,fname)
int taste;
char *fname;     {

ret = n_cfa (sd,fname,DARSTELLUNG,INVERS);
teste ("n_cfa",ret);

return (STD);
}
     .
```

t_rout ist eine Benutzerroutine (siehe Beispiel bei 'define user exits'). Das Feld, dessen Name FORMANT übergibt, wird invers dargestellt.

Feldtyp ändern - change field type

Der Aufruf ändert für ein Feld den Feldtyp von FIX auf VAR oder umgekehrt. Damit ändert sich u.U. auch die Reihenfolge in der Bearbeitung bzw. im Datensatz.

n_cft(sd,fld1,fld2,name,type)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *fld1	Name des vorhergehenden Feldes in der Bearbeitung. Damit legen Sie fest, in welcher Reihenfolge die Felder am Bildschirm eingegeben werden. Für das erste Feld schreiben Sie NULL.
→ char *fld2	Name des vorhergehenden Feldes im Datensatz. Beim Aufruf get_record werden die Felder in der hier festgelegten Reihenfolge in einem Satzbereich übergeben. Sie erhalten also einen strukturierten Datensatz. Für das erste Feld schreiben Sie (char *)NULL. Die Angabe gilt nur für variable Felder.
→ char *name	Name des zu ändernden Feldes.
→ char type	gewünschter Typ des Feldes: FIX Textfeld VAR variables Feld

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENOP	1	Typ nicht bekannt.
ENUP	2	Zeiger auf NULL bei fieldname.
ENOF	4	Feldname nicht bekannt.
ENOF2	16	Feld2 nicht bekannt.
ENOF1	32	Feld1 nicht bekannt.

Beispiel:

```
  .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
  .
  .
ret = n_cft (sd,NULL,NULL,"mfld2",FIX)
teste ("n_cft",ret);
  .
```

Das Feld 'mfld2' wird in ein Textfeld umgewandelt.

Feldinhalt ändern - change field

Der Aufruf ändert den aktuellen Feldinhalt des angegebenen Feldes. 'change field' bearbeitet variable Felder und Textfelder.

n_chf(sd,fieldname,text)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *fieldname	Name des Feldes, wie mit FORMANTGEN oder n_inf festgelegt.
→ char *text	Inhalt des Feldes. Ist die angegebene Zeichenfolge kürzer als die Feldlänge, füllt FORMANT mit Feldfüllzeichen auf. Ist sie länger, wird der Rest abgeschnitten.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf NULL bei fieldname.
ENOF	4	Feldname nicht bekannt.
ENOF1	32	Ausgabefehler.
ETTL	128	Feldinhalt länger als 80 Zeichen.

Beispiel:

```
  .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
  .
ret = n_chf(sd,"F1","Bereich:     ");
teste ("n_chf",ret);
  .
```

Der Aufruf ändert den Inhalt des Feldes mit Namen 'F1' in 'Bereich:'. Das Feld in diesem Beispiel ist ein Fixfeld. Dabei ist darauf zu achten, daß FORMANT mit Feldfüllzeichen auffüllt, wenn der neue Text kürzer ist. Deshalb wurden nach 'Bereich:' noch Leerzeichen geschrieben.

Datensatz in die Maske übertragen - change record

Der Aufruf überträgt einen Datensatz in die aktuelle Maske. Der Satz muß so strukturiert sein, wie er mit 'get record' aufgebaut wurde, das heißt die Reihenfolge der Felder im Datensatz muß mit der definierten Reihenfolge übereinstimmen. Der Aufruf ändert nur die variablen Felder der Maske.

n_chr(sd,record,length)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *record	Strukturierter Datensatz. Das ist der Inhalt aller Felder der Maske, die auszugeben sind.
→ int length	Laenge des Datensatzes. FORMANT ersetzt Zeichen nur bis zur angegebenen Länge. Ist die angegebene Länge kleiner als die tatsächliche Länge der Daten in der Maske bleibt der Rest unverändert. Ist die angegebene Länge größer, ignoriert FORMANT die restlichen Zeichen.

Ergebnis:

ENSA	-4	Kein Speicher verfügbar (nur bei Gegentastprüfen).
ENSD	-2	Situngs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf NULL bei record.
ENOF	4	Keine variablen Felder in der Maske definiert.
N-ENOF1	32	Ein vorangegangener Fehler wurde erkannt.

Beispiel:

```
   .
#define SATZLAENGE 75
   .
int sd;                    /* Sitzungs-Deskriptor */
int ret;                   /* Ergebniswert */
```

```
char satz[SATZLAENGE+1];/* Datensatz */
   .
   .
ret = n_chr (sd,satz,SATZLAENGE)
teste ("n_chr",ret);
   .
```

Der Inhalt des Datensatzes 'satz' wird in die Maske übertragen.

Feldinhalt löschen - clear field

Der Aufruf löscht den Inhalt des angegebenen Feldes in der Maske mit Bildschirmfüllzeichen (Leerzeichen). Mit dem Aufruf können Sie auch den Inhalt von Textfeldern löschen.

n_clf(sd,fieldname)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *fieldname	Name des Feldes, dessen Inhalt zu löschen ist.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENOP	1	Zeiger auf NULL bei fieldname.
ENOF	4	Feldname nicht bekannt.
ENOF1	32	Ausgabefehler.

Beispiel:

```
   .
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
int t_code;             /* logischer Code der Systemsteuertaste */
char akfeld[9];         /* Name des aktuellen Feldes */
   .
   .
ret = n_wir(sd,t_code,akfeld);
teste ("n_wir",ret);
if (t_code == F10)      {
        ret = n_clf (sd,akfeld);
        teste ("n_clf",ret);
        }
   .
```

Wird die Taste F10 gedrückt, wird der Inhalt des aktuellen Feldes gelöscht. Die Schreibmarke steht im nächsten Feld.

Variable Felder der Maske löschen - clear record

Der Aufruf löscht den Inhalt aller variablen Felder der Maske mit Bildschirmfüllzeichen (Standard: Leerzeichen).

n_clr(sd)

Parameter:

→ int sd		Sitzungs-Deskriptor.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENOF	4	Keine variablen Felder in der Maske definiert.
ENOF1	32	Ausgabefehler.

Beispiel:

```
   .
int sd;                    /* Sitzungs-Deskriptor */
int ret;                   /* Ergebniswert */
   .
   .
ret = n_clr (sd);
teste ("n_clr",ret);
   .
```

Alle Datenfelder der aktuellen Maske werden mit Leerzeichen gelöscht.

Sitzung schließen - close session

n_cls schließt die FORMANT-Sitzung. Von FORMANT belegter Speicherplatz wird freigegeben. n_cls muß der letzte FORMANT-Aufruf im Programm sein.

n_cls(sd)

Parameter:

→ int sd		Sitzungs-Deskriptor

Ergebnis:

ENSD	-2	Es existiert keine Sitzung mit diesem Deskriptor.
NOERR	0	Die Sitzung wurde geschlossen.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi.

Beispiel:

```
  .
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
  .
  .
ret = n_cls (sd);               /* Sitzung schliessen */
teste ("n_cls",ret);
  .
                                /* Umschalten auf 24-Zeilen-Modus und
                                   Bildschirm löschen. */
printf("\33[2J\33[1uProgramm %s beendet\n",argv[0]);
exit (0);
  .
```

Die FORMANT-Sitzung wird geschlossen und anschließend wird aufgeräumt.

Maske anlegen - create presentation image

n_cpi legt den Namen der aktuellen Maske fest. Gibt es bereits eine Maske dieses Namens, wird sie geladen. Soll die Maske neu erzeugt werden, reserviert n_cpi nur Speicherplatz. Bevor Sie n_cpi erneut aufrufen, müssen Sie mit n_dpi die aktuelle Maske schliessen. Es kann immer nur eine Maske angelegt sein.

n_cpi(sd,name,mode)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *name	Name der Maske: beliebige Zeichenfolge bei Angabe NEU, SINIX-Dateiname bei Angabe VORHANDEN (siehe unten). Dateinamen können Sie auch mit Pfad angeben.
→ int mode	gibt an, ob die Maske vorhanden ist oder ob sie neu angelegt werden soll.

NEU	Die Maske soll neu angelegt werden. Die Felder der neuen Maske definieren Sie mit n_inf-Aufrufen.
VORHANDEN	Die Maske steht in einer Datei: der Dateiname ist gleich dem Maskennamen.

Standard ist: NEU. Den Wert trägt FORMANT bei einer ungültigen Angabe ein.

Ergebnis:

ENFO	-8	Name ist keine Maske.
ENSA	-4	Kein Speicherplatz verfügbar.
ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENOF	4	Kein gültiger Name oder keine gültigen Felder.
ENFL	4096	Maske inkonsistent.

ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi oder Aufruf in einer Benutzerroutine.

Beispiel:

```
  .
int sd;                    /* Sitzungs-Deskriptor */
int ret;                   /* Ergebniswert */
  .
  .
ret = n_cpi (sd,"masklib/maske1",VORHANDEN);
teste ("n_cpi",ret);
  .
```

Der Aufruf lädt die vorhandene Maske 'maske1'. Das Dateiverzeichnis 'masklib' muß im aktuellen Dateiverzeichnis liegen.

Statusmeldung ausgeben - change status message

Mit dem Aufruf können Sie eine Statusmeldung ausgeben. Die Statusmeldung erscheint in der Systemzeile von Spalte 41 bis Spalte 79. FORMANT sperrt die Tastatur nicht. Die Statusmeldung löschen können Sie, indem Sie ein Leerzeichen als Statusmeldung ausgeben.

n_csm(sd,statusmes)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *statusmes	Text der Statusmeldung. Die Statusmeldung darf bis zu 39 Zeichen lang sein. Ist Sie länger, wird der Rest abgeschnitten. Ist Sie kürzer, füllt FORMANT mit Leerzeichen auf.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf NULL bei statusmes.

Beispiel:

```
   .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
int zaehler = 0;         /* Anzahl geschriebener Saetze */
char meld[41];           /* Feld für Meldungstexte */
   .
   .
sprintf (meld,"Satz geschrieben. Anzahl: %d",zaehler);
ret = n_csm (sd,meld);
teste ("n_csm",ret);
   .
```

Der in meld definierte Text wird als Fehlermeldung ausgegeben.

Feld ausfügen - delete field

n_def fügt ein Feld aus der aktuellen Maske aus. An den Positionen der restlichen Felder ändert sich nichts.

n_def kann variable Felder und Textfelder ausfügen. Beim Ausfügen variabler Felder ändert sich der Aufbau des Datensatzes.

n_def(sd,fieldname)

Parameter:

→ int sd	Sitzungs-Deskriptor (siehe n_ops).
→ char *fieldname	Name des Feldes, das auszufügen ist.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf NULL bei fieldname.
ENOF	4	Feldname nicht bekannt.

Beispiel:

```
   .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
char *fname;             /* Zeiger auf Feldnamen */
   .
   .
fname = "index";
   .
ret = n_def (sd,fname);
teste ("n_def",ret);
   .
```

Das Feld mit Namen 'index' wird ausgefügt.

Maske schließen - destroy presentation image

n_dpi schließt dis aktuelle Maske und gibt deren Speicherplatz frei. Anschließend können Sie mit n_cpi eine neue Maske anlegen oder mit n_cls die FORMANT-Sitzung abschließen.
n_dpi löscht den Bildschirm und ist unbedingt vor dem Schließen der FORMANT-Sitzung aufzurufen.

n_dpi(sd)

Parameter:

→ int sd	Sitzungs-Deskriptor (siehe n_ops).

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi oder Aufruf in einer Benutzerroutine.

Beispiel:

```
  .
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
  .
  .
ret = n_dpi (sd);
teste ("n_dpi",ret);
```

Die aktuelle Maske wird geschlossen. FORMANT löscht den Bildschirm.

Benutzerroutinen für Triggerfelder - define user exits

Mit n_due geben Sie den Namen einer Funktion an, die aufzurufen ist, wenn ein Triggerfeld ausgefüllt wurde. Die Eigenschaft Triggerfeld wird beim Erstellen der Maske vergeben. Pro Triggerfeld ist ein Aufruf n_due nötig. Sie können natürlich für verschiedene Triggerfelder dieselbe Funktion vereinbaren.

'define user exits' ist nach 'create p.i.' aufzurufen.

n_due(sd,fieldname,ptf)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *fieldname	Name eines Triggerfeldes. Wird dieses Feld bei der Eingabe fertig ausgefüllt, ruft FORMANT die mit 'ptf' angegebene Benutzerroutine auf. Hat das Feld nicht die Eigenschaft Triggerfeld, passiert nichts.
→ int (*ptf)()	Zeiger auf eine Benutzerroutine. Das ist eine Funktion, die FORMANT aufruft, wenn das angegebene Triggerfeld ausgefüllt wurde.

Ergebnis:

ENSA	-4	Keine Benutzerroutine mehr möglich (Speichermangel).
ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Erfolgreicher Aufruf.
ENOP	1	Falscher Parameter.
ENOF	4	Feldname nicht bekannt.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi.

Beispiel:

```
  .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
int t_rout();            /* Benutzerroutine fuer Triggerfeld */
  .
```

```
.
ret = n_due (sd,"name",t_rout);
teste ("n_due",ret);
.
```

Der Aufruf definiert die Funktion t_trout als Benutzerroutine für ein Triggerfeld. FORMANT durchläuft die Benutzerroutine, wenn das Feld 'name' der aktuellen Maske bei der Eingabe verlassen wird. Das Feld 'name' muß die Feldeigenschaft 'Triggerfeld' haben. Siehe dazu auch die Beispiele bei 'change field attribute' und in Abschnitt 7.2.

Schreibmarkenposition abfragen - get cursor

Der Aufruf liefert die aktuelle Position der Schreibmarke.

n_gec(sd,fieldname,offset)

Parameter:

→ int sd	Sitzungs-Deskriptor.
← char *fieldname	Name des Feldes, in dem die Schreibmarke steht.
← char *offset	Zeiger auf die Position der Schreibmarke innerhalb des Feldes. Feldanfang ist 0. Beachten Sie, daß offset in 1 Byte geliefert wird (char).

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf NULL bei feldname oder offset.

Beispiel:

```
  .
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
int t_code;             /* logischer Code der Systemsteuertaste */
char akfeld[9];         /* aktuelles Feld */
char offset;            /* Spaltenposition der Schreibmarke */
  .
  .
ret = n_wir (sd,&t_code,akfeld);
teste ("n_wir",ret);

switch (t_code)  {
       case F6  : ret = n_gec (sd,akfeld,&offset);
                  teste ("n_gec",ret);
                  offset +=6;
                  ret = n_sec (sd,akfeld,offset);
                  teste ("n_sec",ret);
                  break;
       .
      }
  .
```

Wird die Taste F6 gedrückt, dann springt die Schreibmarke um 6 Zeichen im Feld weiter. Geht die Position über das Feldende hinaus, springt die Schreibmarke auf den Feldanfang.

Datenfeld übernehmen - get field

Der Aufruf übernimmt den aktuellen Inhalt eines Datenfeldes in einen Datenbereich des Anwendungsprogramms. Das Feld wird als Zeichenfolge übergeben. Der aktuelle Inhalt hängt davon ab ,was beim vorhergehenden Leseaufruf (n_red oder n_wir) eingegeben wurde. Ist kein Leseaufruf vorangegangen oder ist es ein Textfeld (FIX), ist der aktuelle Inhalt die Vorbelegung. Leere Felder füllt FORMANT mit Datenfüllzeichen, ebenso wie nicht ausgefüllte Feldreste.

n_gef(sd,fieldname,buffer)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *fieldname	Name des Feldes, wie bei FORMANTGEN oder bei n_inf festgelegt.
← char *buffer	Zeiger auf einen Bereich für den Feldinhalt. Der Bereich muß so groß definiert sein, daß der Feldinhalt hineinpaßt (+1 für das abschließende \0).

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf NULL bei fieldname.
ENOF	4	Feldname nicht bekannt.

Beispiel:

```
  .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
char s_key[13];          /* Satzschluessel */
  .
  .
ret = n_gef (sd,"index",s_key);
teste ("n_gef",ret);
  .
```

Der Aufruf überträgt den aktuellen Inhalt des Feldes mit Namen 'index' in den Bereich s_key. Das Feld 'index' ist 12 Zeichen lang.

Datensatz aufbauen - get record

Der Aufruf baut aus den variablen Feldern der aktuellen Maske einen strukturierten Datensatz auf. Die Reihenfolge wurde beim Erstellen der Maske mit FORMANTGEN festgelegt oder beim Aufruf n_inf. Der Aufruf n_ger übernimmt den zuletzt mit n_wir oder n_red gelesenen Inhalt. Leere Felder füllt n_ger mit dem jeweils definierten Datenfüllzeichen.

n_ger(sd,record,length)

Parameter:

→ int sd	Sitzungs-Deskriptor (siehe n_ops).
← char *record	Zeiger auf einen Bereich für den strukturierten Datensatz. Der Bereich muß in der Anwendung bereitgestellt sein. n_ger liefert den Datensatz als Zeichenfolge, die mit NULL abgeschlossen ist. Mit dem Umgebungsparameter STRTERM können Sie einstellen, daß jedes Datenfeld mit NULL abgeschlossen wird (siehe Aufruf 'modify environment parameters'.
← int *length	Zeiger auf die Länge der übergebenen Daten ohne das abschließende NULL-Zeichen. Wenn der angegebene Bereich zu klein ist, können schwere Fehler passieren.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf NULL bei record.
ENOF	4	Kein variables Datenfeld vorhanden oder Verkettung inkonsistent.

Beispiel:

```
#define SATZLAENGE 75
```

```
  .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
char satz[SATZLAENGE+1];  /* Datensatz */
int s_laenge;            /* Laenge der uebergebenen Daten */
FILE *a_datei;           /* Ausgabedatei */
  .
  .
ret = n_ger (sd,satz,&s_laenge);
teste ("n_ger",ret);
if (s_laenge > SATZLAENGE)
      teste ("n_ger: Satzlaengenfehler",-300);
fprintf (a_datei,"%s\n",satz);
  .
```

Der Aufruf überträgt alle Datenfelder der Maske in den Bereich 'satz'. Anschließend wird 'satz' in eine Textdatei geschrieben.

Feldeigenschaften abfragen - get field attribute

Der Aufruf liefert für ein Feld den aktuellen Wert der angegeben Feldeigenschaft. Pro Feld und Feldeigenschaft ist ein Aufruf nötig.

n_gfa(sd,fieldname,attribute,r_value)

Parameter:

→ int sd — Sitzungs-Deskriptor.

→ char *fieldname — Name des Feldes, dessen Feldeigenschaften Sie abfragen wollen.

→ int attribute — Feldeigenschaft. Sie können eine der folgenden Angaben machen. Die Bedeutung dieser Angaben und die möglichen Werte finden Sie beim Aufruf 'insert field'. Die Beschreibung der Attribute steht in Abschnitt 3.4.

DARSTELLUNG	at1	Felddarstellung am Bildschirm Farbe des Feldes (nicht in V1.00)
ZEICHENSATZ	at2	Zeichensatz
BEARBEITUNG	at3	Bearbeitungseigenschaften
FELDATT	at4	Feldausrichtung zulässige Eingabezeichen
NACHKOMMA	at5	Eingabeeigenschaften: Nachkommastellen Prüffeld
FUELLFELD	at6	Feldfüllzeichen
FUELLDATEN	at7	Datenfüllzeichen
FELDTYP	type	Feldtyp

← char *r_value — Zeiger auf den aktuellen Wert der Feldeigenschaft (1 Byte). Die möglichen Werte, die FORMANT zurückliefert, entsprechen den Angaben beim Aufruf 'insert field', z.B. MUSS, GESCHUETZT usw. für die Feldeigenschaft BEARBEITUNG.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENOP	1	Feldeigenschaft nicht gefunden oder nicht angegeben.
ENUP	2	Zeiger auf NULL bei fieldname.
ENOF	4	Feldname nicht bekannt.

Beispiel:

```
   .
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
char akfeld[9];         /* Aktuelles Feld */
char a_wert;            /* Attribut-Wert */
   .
   .
ret = n_gfa(sd,akfeld,NACHKOMMA,&a_wert);
teste ("n_gfa",ret);
if (a_wert & KGEGENTAST)     {
    .
    .                   /* Ist das aktuelle Feld ein Prueffeld ? */
    }

if (a_wert & 15)     {  /* Hat das aktuelle Feld Nachkommastellen ? */
                        /* Anzahl Nachkommastellen ausgeben */

    sprintf (meld,"Kommastellen: %u",a_wert);
    ret = n_csm (sd,meld);
    teste ("n_csm",ret);
    }
   .
```

Das Beispiel prüft, ob das aktuelle Feld Prüffeld ist und ob für das Feld Nachkommastellen definiert sind. Beachten Sie, daß bei der Feldeigenschaft NACHKOMMA unterschiedliche Werte addiert sein können (Abfrage mit logischem 'und').

Feld einfügen - insert field

n_inf fügt ein neues Feld in die aktuelle Maske ein. Dabei definieren Sie:

- die Position des Feldes am Bildschirm,
- die Position des Feldes im Datensatz,
- die Reihenfolge bei der Bearbeitung (bei variablen Feldern).

Mit n_inf-Aufrufen können Sie eine neue Maske aufbauen oder eine vorhandene abändern (siehe auch n_cpi).

```
n_inf(sd,fld1,fld2,name,line,column,length,at1,at2,at3,at4,at5,at6,at7,
      type,text)
```

Parameter:

→ int sd	Sitzungs-Deskriptor (siehe n_ops).
→ char *fld1	Name des vorhergehenden Feldes in der Bearbeitung. Damit legen Sie fest, in welcher Reihenfolge die Felder am Bildschirm eingegeben werden. Für das erste Feld schreiben Sie NULL.
→ char *fld2	Name des vorhergehenden Feldes im Datensatz. Beim Aufruf get_record werden die Felder in der hier festgelegten Reihenfolge in einem Satzbereich übergeben. Sie erhalten also einen strukturierten Datensatz. Für das erste Feld schreiben Sie NULL. Die Angabe gilt nur für variable Felder.
→ char *name	Name des Feldes bis zu 8 Zeichen.
→ char line	Zeilenposition des Feldes (1-24).
→ char column	Spaltenposition des Feldes (1-80).
→ char length	Länge des Feldes (maximal soviele Zeichen wie in der Zeile noch Platz haben.
→ char at1	Felddarstellung am Bildschirm Sie können folgende Angaben miteinander kombinieren (durch bitweises 'oder'):

HELL, HALBHELL, DUNKEL, BLINKEND, INVERS, UNTERSTRICHEN

Beispiel: HELL | UNTERSTRICHEN

Standard ist: HELL.

→ char at2

Farbe des Feldes (in Version 1.0 nicht unterstützt):

GRUEN, WEISS, ROT, GELB, BLAU

Zeichensatz

STANDARD, NATIONAL, KLAMMERN, MATH, MOSAICS, IBM

Sie können eine Farbe und einen Zeichensatz angeben, z.B. WEISS | NATIONAL. Die Zeichensätze finden Sie im Anhang abgebildet.

Standard ist: STANDARD.

→ char at3

Bearbeitungseigenschaften Folgende Angaben können Sie miteinander kombinieren, soweit das sinnvoll ist:

MUSS	Mussfeld
VOLLSTAENDIG	Vollständigkeitsfeld
GESCHUETZT	Geschütztes Feld
AUSWEISLESE	Ausweislesefeld
DUPLIZIER	Duplizierfeld
TRIGGER	Triggerfeld
GROSS	Großschreibung
SKIP	Skipfeld

Beispiel: MUSS | DUPLIZIER | GROSS

Standard ist: keine dieser Eigenschaften.

→ char at4

Feldausrichtung

LINKS, RECHTS

Zulässiger Zeichenvorrat für die Eingabe

ALPHANUM, NUM, ALPHA, VORZ, SONDERZ

Die Angaben für Feldausrichtung und Eingabezeichen werden addiert, z.B. LINKS | NUM.

Standard ist: ALPHANUM | RECHTS.

→ char at5

Eingabeeigenschaften. Folgende Angaben können Sie miteinander kombinieren:

Nachkommastellen 0 bis 15. Die Angabe gilt nur für numerische Felder. Sie bewirkt, daß FORMANT bei der Eingabe an der entsprechenden Stelle einen Dezimalpunkt verlangt. Beim Aufbereiten eines nicht vollständig ausgefüllten Feldes setzt FORMANT den Dezimalpunkt an die richtige Stelle.

KGEGENTAST	Prüffeld für Gegentastprüfen (siehe Aufruf 'modify operation mode').
KORRSPERRE	Korrektursperre für Prüffeld, wirkt nur zusammen mit KGEGENTAST.

Beispiel: 3 | KGEGENTAST

Standard ist: keine dieser Angaben.

→ char at6

Feldfüllzeichen. Damit wird das Feld bei der Ausgabe am Bildschirm aufgefüllt.

Zulässig ist jedes abdruckbare Zeichen.

→ char at7

Datenfüllzeichen. Damit wird das Empfangsfeld im Programm aufgefüllt (bei n_ger und n_gef).

Zulässig ist jedes abdruckbare Zeichen.

→ char type

Feldtyp

FIX	Textfeld.
VAR	variables Feld.

	VAR | VORB	geben Sie an, wenn Sie ein variables Feld mit Text vorbelegen wollen.
	VAR | LEER	Feld nicht vorbelegen (Standard für variable Felder).
→ char *text		definiert den Inhalt von Textfeldern Bei variablen Feldern wird text nur ausgewertet, wenn Sie bei type zusätzlich VORB angegeben haben. Sie müssen aber in jedem Fall eine Zeichenfolge in der Länge des Feldes angeben (z.B. Leerzeichen).

Eine Tabelle, wie die Attribute den Angaben bei FORMANTGEN zugeordnet sind, finden Sie in Abschnitt 3.4. Dort sind auch alle Attribute erklärt.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENOP	1	Fehler bei den Attributen.
ENOF	4	Feld ist schon vorhanden.
ENOF2	16	fld2: Name nicht bekannt.
ENOF1	32	fld1: Name nicht bekannt.
EPOS	64	ungültige Positionsangabe.
ETTL	128	Längenfehler.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi.

Beispiel:

```
   .
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
   .
   .
ret = n_inf (sd,
             NULL,      /* Vorgaenger Bearbeitung */
             NULL,      /* Vorgaenger Datensatz */
             "index",   /* Feldname */
             22,        /* Zeile */
             5,         /* Spalte */
```

```
                8,          /* Laenge */
                HELL,       /* Darstellung */
                STANDARD,   /* Zeichensatz */
                NULL,       /* Bearbeitung */
                LINKS|NUM,  /* Feldatt. */
                0,          /* Nachkomma */
                ' ',        /* Fuellfeld */
                ' ',        /* Fuelldaten */
                VAR,        /* Feldtyp */
                "          " /* Feldinhalt */
                );
teste ("n_inf",ret);
```

Der Aufruf fügt ein variables Feld mit den angegebenen Eigenschaften ein. Es wird das erste Feld in der Bearbeitungsreihenfolge und im Datensatz.

Umgebungsparameter ändern - modify environment parameters

Der Aufruf ändert während des Programmlaufs die Umgebungsparameter. Für die Umgebungsparameter setzt FORMANT beim Öffnen der Sitzung Standardwerte. Die jeweils aktuellen Werte könne Sie mit dem Aufruf n_rep abfragen.

n_mep(sd,environpar,value)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ int environpar	Zu ändernder Umgebungsparameter (siehe Tabelle). Für jeden zu ändernden Paramter ist ein Aufruf nötig.
→ char value	Wert des Parameters (siehe Tabelle).

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Erfolgreicher Aufruf.
ENOP	1	Fehlerhafter Parameterwert.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi oder Aufruf in einer Benutzerroutine.

Tabelle der Umgebungsparameter

FUELLZEICHEN Bildschirmfüllzeichen. Dies ist das Zeichen, mit dem alle Stellen des Bildschirms aufgefüllt werden, die nicht von Feldern belegt sind. Auch Felder mit Attribut 'leer' füllt FORMANT mit Bildschirmfüllzeichen.

Werte: ' ' Leerzeichen (Standard).

alle abdruckbaren Zeichen.

Hinweis: Dieser Parameter ist nur zulässig, wenn Masken im Programm neu aufgebaut werden und nicht bei Masken, die mit FORMANTGEN erstellt wurden. Beim Maskenwechsel ist das Bildschirmfüllzeichen immer entsprechend einzustellen!

HINTERGRUND Hintergrunddarstellung des Bildschirms.

Werte: NORMAL Der Bildchirm wird normal dargestellt

INVERS: Der Bildschirm wird invers dargestellt. alle abdruckbaren Zeichen.

Hinweis: Dieser Parameter ist nur zulässig, wenn Masken im Programm neu aufgebaut werden und nicht bei Masken, die mit FORMANTGEN erstellt wurden. Beim Maskenwechsel ist die Hintergrunddarstelllung immer entsprechend einzustellen!

FORMANTCONTROL Eingabereaktion, feldweise oder satzweise. Damit stellen Sie ein, wann FORMANT die Kontrolle an das Anwendungsprogramm zurückgibt.

Werte: SATZ Satzweise Eingabe. Nach dem Leseaufruf erhält das Programm erst die Steuerung, wenn alle Datenfelder der Maske ausgefüllt sind (Standard).

FELD Feldweise Eingabe. Ein Leseaufruf liefert gerade ein Datenfeld.

FELDLEERZ Feldleerzeichen. Dies ist das Zeichen, mit dem ein leeres Feld angezeigt wird, sobald es zur Eingabe ansteht.

Werte: '_' Unterstrich (Standard). alle abdruckbaren Zeichen.

RUECK_AUFB Feldaufbereitung bei Rückpositionierung. Die Angabe bestimmt die Reaktion, wenn ein Feld bei der Eingabe rückwärts verlassen wird (nicht bei feldweiser Reaktion).

Werte: JA Das Feld wird aufbereitet (Standard). Es gilt somit als ausgefüllt.

NEIN Der Inhalt des Feldes wird gelöscht. Es gilt nicht als ausgefüllt.

VOR_AUFB Feldaufbereitung beim Überspringen von Feldern. Die Angabe bestimmt die Reaktion, wenn ein Feld bei der

Eingabe vorwärts übersprungen wird (nicht bei feldweiser Reaktion).

Werte:

JA	Übersprungene Felder werden aufbereitet (Standard). FORMANT füllt mit Feldfüllzeichen auf.
NEIN	Übersprungene Felder bereitet FORMANT nicht auf. Felder, die bereits ausgefüllt sind, werden nicht gelöscht.

LOESCHLINKS zeichenweise Rückpositionierung. Die Angabe bestimmt die Reaktion, wenn man die Taste [←] drückt.

Werte:

NEIN	Die Zeichen werden nicht gelöscht (Standard).
JA	Die Zeichen werden gelöscht.

Hinweis: Die Taste [⌫] löscht die Zeichen immer.

AUTODUPVORF Automatisches Duplizieren in vorbelegten Feldern.

Werte:

NEIN	In Feldern mit dem Attribut 'vorbelegt' wird nicht automatisch dupliziert (Standard). Ist die Duplizierfunktion eingeschaltet, bleiben in diesen Feldern die vorbelegten Daten erhalten (siehe auch Aufruf n_sdr).
JA	Ist die Duplizierfunktion eingeschaltet, werden die vorbelegten Daten mit den Duplizierdaten überschrieben.

Die Duplizierfunktion wirkt nur bei variablen Feldern.

STRTERM Stringterminator. Der Stringterminator ist NULL und begrenzt die Eingabefelder, die FORMANT beim Aufruf 'get record' übergibt.

Werte:

NEIN	Nur der Satz wird mit NULL abgeschlossen (Standard).

JA	Jedes Feld wird mit NULL abgeschlossen. Beachten Sie bitte, daß dadurch der Satz für 'get record' länger wird.

Beispiel:

```
   .
int i;                  /* Laufvariable */
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */

static int up_tab[] = { RUECK_AUFB,     NEIN,   /* Umgebungsparameter */
                        FELDLEERZ,      '.',
                        VOR_AUFB,       NEIN,
                        STRTERM,        JA   };
   .
   .
                                                /* Umgebungsparameter
                                                   einstellen nach up_tab */
for (i=0; i<8; i+=2)    {
        ret = n_mep (sd,up_tab[i],up_tab[i+1]);
        teste ("n_mep",ret);
        }
   .
```

Mit vier Aufrufen n_mep in einer Schleife werden die Umgebungsparameter eingestellt, die in der Tabelle festgelegt sind.

Reaktion auf Systemsteuertasten - modify input reaction table

Der Aufruf legt die Reaktion auf Systemsteuertasten fest. Pro Systemsteuertaste ist ein Aufruf nötig. Sie können

- Tasten sperren,
- als Reaktion eigene Benutzerroutinen ausführen,
- wieder Standardbehandlung vereinbaren,
- die Eingabe beenden, d.h. die Steuerung geht wieder an das Anwendungsprogramm zurück.

Funktionstasten sind ein Sonderfall der Systemsteuertasten. Deren Bedeutung müssen Sie immer selbst definieren.

Die standardmäßige Wirkung aller Tasten ist beschrieben in Abschnitt 2.3.

n_mir(sd,key,reaction,ptf)

Parameter:

→ int sd — Sitzungs-Deskriptor (siehe n_ops).

→ int key — Logischer Code der Systemsteuertaste. Erlaubt sind die Angaben wie in der Tabelle unten.

→ int reaction — Reaktion auf die Systemsteuertaste:

STD — Standardbehandlung. Die Taste löst die Funktion aus, wie in Abschnitt 2.3 beschrieben.

Funktionstasten:

F1 bis F27, MENU, HELP, START, END:

FORMANT gibt die Steuerung sofort an das Anwendungsprogramm und liefert mit dem Leseaufruf den logischen Code der Taste. Beim nächsten Leseaufruf setzt FORMANT die Eingabe mit dem nächsten Feld fort. War

das aktuelle Feld bereits das letzte der Maske, kehrt dieser nächste Leseaufruf mit 'AUSG' zurück.

IGN — Taste sperren. FORMANT meldet 'E1 Systemsteuerzeichen unzulaessig', wenn die Taste gedrückt wird und es ist keine Funktionstaste. Auf gesperrte Funktionstasten reagiert FORMANT nicht.

USR — Die mit 'ptf' angegebene Benutzerroutine soll ausgeführt werden. Anschließend setzt FORMANT fort, wie in der Benutzerroutine definiert (siehe Kapitel 7).

APL — Das Anwendungsprogramm soll die Steuerung erhalten. Der Leseaufruf wird beendet und liefert den logischen Code der Systemsteuertaste. Dies entspricht genau der Standardbehandlung einer Funktionstaste.

APL wirkt bei Funktionstasten wie STD.

Die Tabelle unten zeigt, welche Angabe bei welchen Tasten erlaubt ist.

→ int (*ptf)() — Zeiger auf eine Benutzerroutine. Das ist eine Funktion, die ausgeführt werden soll, wenn die angegebene Taste gedrückt wurde. ptf wird nur bei USR ausgewertet. Bei STD, IGN oder APL können Sie für ptf NULL angeben.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Erfolgreicher Aufruf.
ENOP	1	Falscher Parameterwert.
ESEQ	8192	Falsche Reihenfolge: kein n_ops.

Tabelle der änderbaren Tasten

Änderbar sind:	F1 bis F27		E-STD	E-IGN	E-USR	
	MENU					
	HELP	START	END			
	SEND	LVD	E-STD	E-IGN	E-USR	E-APL
	FE+	FE-				
	LAST	HOME	NEXT			
	BACKWARDS		FORWARDS			
nur sperrbar sind:	FKOR	KOR	ANZ	E-IGN		
	VALID					
	DUP					

Alle übrigen Tasten können Sie nicht ändern. Zur Tastenbelegung siehe auch Abschnitt 2.3.8 und Anhang.

Beispiele:

```
  .
int i;                  /* Laufvariable */
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
int r_f10();            /* Benutzerroutine fuer F10 */
  .
  .
for (i=F13; i<=F26; i++)        {       /* Funktionstasten F13 bis
        ret = n_mir (sd,i,IGN,NULL);    /* F26 sperre */
        teste ("n_mir",ret);
        }

ret = n_mir (sd,F10,USR,r_f10);         /* F10 Routine r_F10 zuordnen */
teste ( "n_mirF10,ret);
  .
```

Nach diesen Aufrufen sind F13 bis F26 gesperrt. Wird F10 gedrückt, springt FORMANT die Funktion r_f10 an. Dazu siehe auch das Beispiel in Abschnitt 7.2.

Eingabeparameter ändern - modify operating mode

Der Aufruf ändert Eingabeparameter. In Version 1.0 ist dies nur

- Gegentastprüfen ein- und ausschalten.

n_mom(sd,oppar,value)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char oppar	Eingabeparameter: GEGENTASTP
→ char value	Wert des angegebenen Eingabeparameters.

Tabelle der Eingabeparameter

GEGENTASTP Gegentastprüfen. Die Eingabe in ein Datenfeld wird gegen einen vorgegebenen Inhalt geprüft. Folgende Voraussetzungen müssen erfüllt sein:

1. Das Datenfeld muß die Eigenschaft 'Prüffeld' oder 'Prüffeld nicht korrigierbar' haben.

2. Sie müssen Gegentastprüfen einschalten.

3. Sie müssen einen Datensatz vorgeben, gegen dessen Inhalt zu prüfen ist. Das kann ein Datensatz aus einer Datei sein oder ein Datensatz, der bereits eingegeben und mit 'get record' geholt wurde.
   Den Datensatz geben Sie mit 'change record' vor, nachdem Sie Gegentastprüfen eingeschaltet haben. Soll dieser Satz nicht am Bildschirm gezeigt werden, ist vor dem 'change record' ein 'clear record' aufzurufen.

4. Mit einem Leseaufruf fordern Sie die Eingabe im Prüfmodus an Aus diesem Leseaufruf kehrt FORMANT zurück mit dem logischen Code der Systemsteuertaste wie üblich.

Beachten Sie aber: Im Prüfmodus fordert FORMANT nur Prüffelder zur Eingabe an und füllt alle anderen Felder selbst mit dem Inhalt der Vorgabe aus (logischer Code der Systemsteuertaste: AUSG).

FORMANT prüft die Eingabe gegen die Vorgabe. Bei Abweichungen meldet FORMANT 'E13 Prueffehler !'.

Zum Ablauf der Eingabe und zu den Korrekturmöglichkeiten siehe Abschnitt 2.3.6.

Prüffelder kann man vorzeitig nur über Funktionstasten verlassen. Damit können Sie zum Beispiel einen Ausgang für 'nicht korrigierbare Prüffelder' vorsehen.

5. Wollen Sie einen Datensatz bzw. ein Feld normal einlesen, müssen Sie das Gegentastprüfen ausschalten.

Werte:

AUS	Gegentastprüfen aus (Standard): normale Eingabe eines Datensatzes bzw. Datenfeldes.
EIN	Gegentastprüfen ein: FORMANT fordert nur Prüffelder an und prüft die Eingabe gegen einen mit 'change record' vorgegebenen Datensatz. Die übrigen Felder füllt FORMANT mit dem Inhalt dieses Datensatzes selbst aus. Das gilt bei feldweiser und satzweiser Eingabereaktion.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.

Beispiel:

```
   .
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
int t_code;             /* logischer Code der Systemsteuertaste */
char akfeld[9];
char mode;              /* Pruefmodus */
int zaehler = 0;
char satz[SATZLAENGE+1];/* Datensatz */
int s_laenge;           /* Laenge der uebergebenen Daten */
char meld[41];          /* Feld fuer Meldungen */
FILE *a_datei;          /* Ausgabedatei */
   .
   .
do      {                                       /* Leseschleife */

        ret = n_wir (sd,&t_code,akfeld);        /* Ein- Ausgabe */
        teste ("v_maske1-n_wir",ret);

        switch (t_code) {
               case FORWARDS:
               case NEXT    :
               case FEPLUS  :
               case AUSG    :                   /* wenn fertig ausgefuellt */

                                                /* Pruefmodus ein ? */
            ret = n_rom (sd,GEGENTASTP,&mode);
            teste ("n_rom",ret);
            if (mode == AUS){
                                                /* Datensatz über-
                                                   tragen */
                 ret = n_ger (sd,satz,&s_laenge);
                 teste ("n_ger",ret);
                                                /* Satz nicht anzeigen
                                                   beim Gegentasten */
                 ret = n_clr(sd);
                 teste ("n_clr",ret);
                                                /* Gegentastpruefen
                                                   einschalten */
                 ret = n_mom (sd,GEGENTASTP,EIN);
                 teste ("n_mom",ret);
                 ret = n_csm (sd,"Pruefmodus ein.");
                 teste ("n_csm",ret);
                                                /* Pruefdaten */
                 ret = n_chr (sd,satz,SATZLAENGE);
                 teste ("n_chr",ret);
                 }

            else if (t_code == AUSG){

                 zaehler++;
                                                /* Datensatz aufbauen */
                 ret = n_ger (sd,satz,&s_laenge);
                 teste ("v_maske1-n_ger",ret);
```

```
                                        /* Datensatz schreiben */
                    fprintf(a_datei,"%s\n",satz);
                    sprintf(meld,"Satz geschrieben. Anzahl: %d",zaehler);
                    ret = n_csm (sd,meld);
                    teste ("n_csm",ret);
                                        /* Gegentastpruefen
                                           ausschalten */
                    ret = n_mom (sd,GEGENTASTP,AUS);
                    teste ("n_mom",ret);

                    n_clr (sd);
                    teste ("n_clr",ret);
                    }
                    break;

            case ...      :
            .
            .

            }
    } while (t_code != END);
```

Das Beispiel zeigt eine Leseschleife, in die die Funktion Gegentastprüfen eingebaut ist. Wurde die Maske fertig ausgefüllt, wird der Prüfmodus eingeschaltet und der Leseaufruf erneut durchlaufen. Der Leseaufruf kehrt aus dem Prüfmodus immer mit Tastencode 'AUSG' zurück, wenn das letzte Feld der Maske kein Prüffeld ist. Ist das Prüfen beendet, wird der gelesene Satz geschrieben und der Prüfmodus ausgeschaltet. Dann kann man den nächsten Datensatz eingeben.

Ein Beispiel zum Prüfen bereits erfaßter Sätze finden Sie in Abschnitt 4.3.2.

Sitzungsparameter ändern - modify session parameters

Der Aufruf ändert während des Programmlaufs die Sitzungsparameter. Für die Sitzungsparameter setzt FORMANT beim Eröffnen der Sitzung Standardwerte. Die jeweils aktuellen Werte können Sie mit dem Aufruf 'request session parameters' abfragen.

n_msp(sd,sessionpar,value)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ int sessionpar	Zu ändernder Sitzungsparameter (siehe Tabelle).
→ char *value	Zeiger auf den Wert des Parameters (siehe Tabelle). Wird ein falscher Wert angegeben, setzt n_msp den Standardwert ein.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Erfolgreicher Aufruf.
ENOP	1	Fehlerhafter Parameterwert.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi oder Aufruf in einer Benutzerroutine.

Tabelle der Sitzungsparameter

Die angegebenen Werte, z.B. TERMINAL usw. sind Zeiger!

GERAETEKLASSE Geräteklasse, die unterstützt werden soll.

Werte:	TERMINAL	Datensichtstation (Standard).
	DRUCKER	Drucker. Diesen Eintrag ermöglicht es, Druckermasken auszugeben, die mit FORMANTGEN aus mehreren Formaten zusammengesetzt wurden. Wenn N-GERAETEKLASSE auf DRUCKER gesetzt ist,

- können Sie eine Druckermaske ausgeben. Dazu verwenden Sie den Aufruf N-WRT oder den Aufruf N-PPI. Beide Aufrufe wirken in diesem Fall gleich.
- dürfen Sie folgende Aufrufe nicht verwenden:
  N-WIR
  N-RED

  Diese Aufrufe liefern das Ergebnis N-ESEQ.
- haben folgende Aufrufe keine Wirkung:
  N-MIR
  N-DUE
  N-SEC
  N-SDR

INPER interaktive Peripherie (noch nicht unterstützt).

GERAETETYP Gerätetyp, der unterstützt werden soll. FORMANT nimmt den entsprechenden Eintrag aus der Datei /etc/termcap.

Werte:
- "fo97801" Datensichtstation 97801 (Standard).
- "9001" Drucker 9001.
- "9004" Drucker 9004.

Hinweis: Die Drucker 9001 und 9004 werden nur mit dem vorhandenen Standardzeichensatz unterstützt. FORMANT bedient den Drucker über das Kommando lpr. In der Shell-Variablen FOPRT können Sie auch ein anderes Kommando oder ein eigenes Programm vereinbaren, das die Druckausgabe macht (siehe auch N-PPI).

BERECHTIGUNG möglicher Zugriff auf das Gerät.

Werte:
- BEIDES Es ist Lesen und Schreiben erlaubt (Standard).

	LESEN	Es ist nur Lesen erlaubt.
	SCHREIBEN	Es ist nur Schreiben erlaubt.

FEHLERTEXTE gibt an, aus welcher Datei die FORMANT-Fehlermeldungen gelesen werden sollen.

Werte:	F_STD	FORMANT liest aus der Datei 'ftext.std' (Standard).
	F_NAT	FORMANT liest aus der Datei 'ftext.nat'.

Beide Dateien sind im Dateiverzeichnis /usr/lib/formant. Standardmäßig enthält:

ftext.std	englische Fehlertexte.
ftext.nat	deutsche Fehlertexte.

Den Inhalt der Fehlertextdateien können Sie ändern. Die Fehlernummern müssen unverändert bleiben. Die Texte können Sie z.B. in eine andere Sprache übersetzen.

Beispiel:

```
  .
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
  .
  .
                                /* Fehlertexte deutsch ausgeben */
ret = n_msp (sd,FEHLERTEXTE,F_NAT);
teste ("n_msp",ret);
  .
```

Der Aufruf bewirkt, daß Fehlertexte aus der Datei /usr/lib/formant/ftext.nat genommen werden (standardmäßig deutsch).

Sitzung eröffnen - open session

n_ops eröffnet eine FORMANT-Sitzung. Für die Sitzung teilt FORMANT einen Sitzungs-Deskriptor zu. Über den Sitzungs-Deskriptor identifiziert FORMANT alle Datenbereiche dieser Sitzung. n_ops setzt die Standardwerte für die Sitzungs-Parameter Es muß der erste FORMANT-Aufruf im Programm sein und darf erst wieder aufgerufen werden, wenn die Sitzung geschlossen wurde.

n_ops(version,z_sd)

Parameter:

→ int version	FORMANT-Version. Anzugeben ist die FORMANT-Version mit der gearbeitet werden soll: 100 für Version 1.0 101 für Version 1.01 usw. Ist die tatsächliche Versionsnummer kleiner als die hier angegebene, kehrt FORMANT mit ENVS zurück. Damit wird vermieden, daß Programme mit Funktionen neuerer FORMANT-Versionen mit einer alten FORMANT-Version laufen.
← int *z_sd	Sitzungs-Deskriptor. n_ops liefert einen Zeiger auf den Sitzungs-Deskriptor. Der Wert von z_sd ist in allen weiteren FORMANT-Aufrufen anzugeben.

Ergebnis:

ENSA	-4	Kein Speicherplatz verfügbar.
ENSD	-2	Maximale Sitzungsanzahl überschritten.
NOERR	0	Aufruf erfolgreich.
ENOP	1	Fehlerdatei nicht ladbar.
ENOF	4	Felder für Fehlertext und Status nicht anlegbar.
ENVS	1024	Falsche FORMANT-Version gewünscht.
ESEQ	8192	Falsche Reihenfolge: kein n_cls oder Aufruf in einer Benutzerroutine.

Beispiel:

```
   .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
int version = 100;       /* FORMANT-Version 1.00 */
   .
   .
ret = n_ops (version,&sd);
teste ("n_ops",ret);
   .
```

Der Aufruf eröffnet die FORMANT-Sitzung. Der Sitzungs-Deskriptor wird als Adresse übergeben.

Maske ausdrucken - print presentation image

Der Aufruf druckt die aktuelle Maske auf dem Drucker aus. Alle aktuellen Feldinhalte werden ausgedruckt, auch die der variablen Felder. Felder mit der Eigenschaft DUNKEL druckt n_ppi nicht aus. Alle anderen Darstellungseigenschaften berücksichtigt n_ppi nicht.

n_ppi(sd)

Parameter:

→ int sd	Sitzungs-Deskriptor (siehe n_ops).

Ergebnis:

ENSA	-4	Kein Speicherplatz verfügbar.
ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi.

Hinweis:

n_ppi erzeugt eine temporäre Datei. Diese wird standardmäßig mit lpr ausgedruckt und anschließend gelöscht.

Dateiname: _ppi_#.tmp (# von 0 bis 9).

lpr wird mit dem Schalter -pb3 aufgerufen. Das bedeutet, daß die Zeichenbreite auf 17 Zeichen/Zoll eingestellt wird und in einer Zeile max. 136 Zeichen möglich sind.

Die Standardverarbeitung mit dem Kommando lpr läßt sich abändern. In der Shell-Variablen FOPRT können Sie ein anderes Kommando vereinbaren, mit dem die temporäre Datei verarbeitet wird.

Beispiele für die Variable FOPRT:

FOPRT=lpr vor dem Programmaufruf bewirkt, daß Formate in der normalen Zeichenbreite ausgedruckt werden.

FOPRT=prog bewirkt, daß die temporäre Datei an das (benutzereigene) Programm 'prog' übergeben wird.

FOPRT ist zu exportieren (Kommando export FOPRT).

Beispiel:

```
   .
int sd;                    /* Sitzungs-Deskriptor */
int ret;                   /* Ergebniswert */
   .
   .
ret = n_ppi (sd);
teste ("n_ppi",ret);
   .
```

Der Aufruf druckt die aktuelle Maske mit allen Feldinhalten auf dem Drucker aus.

Daten einlesen - read

Der Aufruf wirkt wie der Aufruf 'write read' ohne den Teil 'Schreiben'.

Lesen: FORMANT erwartet die Eingabe in die Datenfelder der Maske. FORMANT prüft und verarbeitet die Eingabe entsprechend den aktuell definierten Bedingungen (siehe Abschnitt 2.3).

Anschließend erhält das Programm die Steuerung zurück.

In der Version 1.0 sind 'read' und 'write read' identisch.

n_red(sd,r_sys,fieldname)

Parameter:

→ int sd	Sitzungs-Deskriptor.
← int *r_sys	Zeiger auf den logischen Code der Systemsteuertaste. Dieser Wert gibt an, wodurch die Eingabe der Daten beendet wurde. Mögliche Werte siehe Aufruf 'write read'.
← char *fieldname	Name des aktuellen Feldes. Dies ist dasjenige Feld der Maske, das FORMANT als letztes bearbeitet hat.

Ergebnis:

ERR	-32	Ablauffehler: ein früher aufgetretener Fehler wurde erkannt.
ENSY	-16	Unbekanntes Systemsteuerzeichen: ein früher aufgetretener Fehler wurde erkannt. Möglicherweise stack überschrieben.
ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf Null bei fieldname oder r_sys.
ENOF	4	Ablauffehler: ein früher aufgetretener Fehler wurde erkannt.
APL	8	Unterbrechung durch Benutzerroutine.

EPOS	64	Positionierungsfehler: ein früher aufgetretener Fehler wurde erkannt.
FFELD	256	Erstes Feld in der Bearbeitungsreihenfolge erreicht (nur bei feldweisem Betrieb).
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi oder Aufruf in einer Benutzerroutine.

Es gelten dieselben Hinweise, wie beim Aufruf 'write read'.

Beispiel:

```
  .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
int t_code;              /* logischer Code der Systemsteuertaste */
char akfeld[9];          /* Name des aktuellen Feldes */
  .
  .
ret = n_red (sd,&t_code,akfeld);
teste ("n_red",ret);
  .
```

Umgebungsparameter abfragen - request environment parameters

Der Aufruf liefert den aktuellen Wert des angegebenen Umgebungsparameters. Für jeden gewünschten Parameter ist ein Aufruf nötig.

n_rep(sd,environpar,r_value)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ int environpar	Zurückzuliefernder Umgebungsparameter (siehe Tabelle).
← char *r_value	Zeiger auf den Wert des Umgebungsparameters (siehe Tabelle).

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Erfolgreicher Aufruf.
ENOP	1	Gewünschter Parameter nicht gefunden, die Angabe liegt nicht im zulässigen Wertebereich.
ENUP	2	r_value zeigt auf NULL.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi.

Tabelle der Umgebungsparameter

Die Tabelle mit ausführlichen Erläuterungen finden Sie beim Aufruf 'modifiy environment parameters'.

Parameter	Bedeutung	Standard	weitere Werte
FUELLZEICHEN	Bildschirmfüllzeichen	' '	alle abdruckbaren Zeichen
HINTERGRUND	Hintergrunddarstellung	NORMAL	INVERS
FORMANTCONTROL	Eingabereaktion	SATZ	FELD
FELDLEERZ	Feldleerzeichen	'_'	alle abdruckbaren Zeichen
RUECK_AUFB	Feld aufbereiten bei Rückpositionierung	JA	NEIN

VOR_AUFB	Feld aufbereiten beim Überspringen	JA	NEIN
LOESCHLINKS	zeichenweise Rückposit.	NEIN	JA
AUTODUPVORBEF	Automatisches Duplizieren in vorbelegten Feldern.	NEIN	JA
STRTERM	Stringterminator	NEIN	JA

Beispiel:

```
   .
int sd;                    /* Sitzungs-Deskriptor */
int ret;                   /* Ergebniswert */
char wert;                 /* Wert des Parameters */
   .
   .
ret = n_rep (sd,VOR_AUFB,&wert);
teste ("n_rep",ret);

if ( wert == JA ){
        ret = n_mep (sd,VOR_AUFB,NEIN);
        teste ("n_mep",ret);
        ret = n_csm (sd,"Vor-Aufbereiten aus.");
        teste ("n_csm",ret);
        }
else    {
        ret = n_mep (sd,VOR_AUFB,JA);
        teste ("n_mep",ret);
        ret = n_csm (sd,"Vor-Aufbereiten ein.");
        teste ("n_csm",ret);
        }
   .
```

Die Aufruffolge ändert den Wert des Parameters VOR_AUFB abhängig vom aktuell eingestellten Wert.

Eingabeparameter abfragen - request operating mode

Der Aufruf liefert den aktuellen Wert der Eingabeparameter. In Version 1.0 ist dies nur Gegentastprüfen ein oder aus.

n_rom(sd,oppar,r_value)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char oppar	Eingabeparameter: GEGENTASTP Gegentastprüfen.
← char *r_value	Zeiger auf den Wert des angegebenen Eingabeparameters: AUS Gegentastprüfen aus. EIN Gegentastprüfen ein.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf NULL bei r_value.

Beispiel:

```
   .
int sd;                    /* Sitzungs-Deskriptor */
int ret;                   /* Ergebniswert */
char mode;
   .
   .
ret = n_rom (sd,GEGENTASTP,&mode);
teste ("n_rom",ret);
   .
```

Der Aufruf liefert in 'mode' den Wert EIN oder AUS zurück. Ein ausführliches Beispiel finden Sie beim Aufruf 'modify operation mode'.

Sitzungsparameter abfragen - request session parameters

Der Aufruf liefert den aktuellen Wert des angegebenen Sitzungsparameters. Für jeden gewünschten Parameter ist ein Aufruf nötig.

n_rsp(sd,sessionpar,r_value)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ int sessionpar	Zurückzuliefernder Sitzungsparameter (siehe Tabelle).
← char *r_value	Zeiger auf den Wert des Sitzungsparameters (siehe Tabelle). Beachten Sie: Die Angaben F_NAT usw. sind bereits Zeiger.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Erfolgreicher Aufruf.
ENOP	1	Gewünschter Parameter nicht gefunden, die Angabe liegt nicht im zulässigen Wertebereich.
ENUP	2	r_value zeigt auf NULL.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi.

Tabelle der Sitzungsparameter

Die Tabelle mit ausführlichen Erläuterungen finden Sie beim Aufruf 'modifiy session parameters'.

Parameter	Bedeutung	Standard	weitere Werte
GERAETEKLASSE	Geräteklasse	TERMINAL	DRUCKER INPER
GERAETETYP	Gerätetyp	"fo97801"	"9001" "9004"
BERECHTIGUNG	möglicher Zugriff	BEIDES	LESEN SCHREIBEN
FEHLERTEXTE	Fehlerdatei	F_STD	F_NAT

Beispiel:

```
  .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
char pwert;              /* Parameterwert */
  .
  .
ret = n_rsp (sd,FEHLERTEXTE,&pwert);
teste ("n_rsp",ret);

if ( pwert == *F_STD )       {
        ret = n_msp (sd,FEHLERTEXTE,F_NAT);
        teste ("n_mep",ret);
        }
else    {
        ret = n_msp (sd,FEHLERTEXTE,F_STD);
        teste ("n_mep",ret);
        }
  .
```

Die Aufruffolge ändert die Einstellung des Parameters FEHLERTEXTE abhängig vom aktuell eingestellten Wert.

Signalton erzeugen - set beeper

Der Aufruf erzeugt einen Signalton, dessen Länge von der angegebenen Zahl abhängt. Während des Signaltones akzeptiert FORMANT keine Eingabe.

n_sbe(sd,number)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char number	Zahlenwert von 1 bis 128: Anzahl der auslösenden Steuerzeichen. Diese bestimmt die Länge des Signaltones.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.

Beispiel:

```
  .
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
int t_code;             /* logischer Code der Systemsteuertaste */
  .
  .
switch (t_code) {
      case FORWARDS:
      case NEXT     :
      case FEPLUS   :
      case AUSG     :       {              /* wenn fertig ausgefuellt, */

          ret = n_sbe (sd,2);              /* Signalton */
          teste ("n_sbe",ret);
              .
          }
  .
```

Wenn die Maske fertig ausgefüllt ist, wird ein Signalton erzeugt.

Dupliziersatz setzen - set dup record

Der Aufruf übernimmt den angegebenen Bereich als Dupliziersatz und schaltet die Duplizierfunktion ein. Dupliziert werden können dann:

- mit der Dupliziertaste beliebige Felder,
- automatisch alle Felder, die die Feldeigenschaft Duplizierfeld haben.

Zum Duplizieren siehe auch Abschnitt 2.3.7. Die Duplizierfunktion ausschalten können Sie, indem Sie als Dupliziersatz einen Zeiger auf NULL angeben.

Sie müssen eine Maske angelegt haben, bevor Sie 'set dup record' aufrufen.

n_sdr(sd,dup_record,length)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *dup_record	Dupliziersatz. Der Satz muß so strukturiert sein, wie er mit 'get record' aufgebaut wurde, das heißt die Reihenfolge der Felder im Datensatz muß mit der definierten Reihenfolge übereinstimmen. Da mit der Dupliziertaste beliebige Felder dupliziert werden können, sollten Sie den Dupliziersatz vollständig versorgen. Der Dupliziersatz darf keine Zeichen enthalten, deren Eingabe syntaktisch nicht erlaubt ist. Sie müssen eventuell Datenfüllzeichen im Satz z.B. in Leerzeichen umwandeln (= Bildschirmfüllzeichen). Wenn Sie einen Zeiger auf NULL angeben, wird die Duplizierfunktion ausgeschaltet.
→ int length	Länge des Dupliziersatzes.

Ergebnis:

ENSA	-4	Speichermangel.
ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Versuch, Duplizierfunktion auszuschalten, obwohl sie nicht eingeschaltet war.

Beispiel:

```
   .
#define SATZLAENGE 75
   .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
char satz[SATZLAENGE]    /* Datensatz */
   .
   .
n_sdr (sd,satz,SATZLAENGE);
teste ("n_sdr",ret);
   .
```

Der Aufruf bewirkt, daß beim nächsten Lesen Felder der Maske dupliziert werden können. Der Feldinhalt wird aus 'satz' entnommen. Felder mit der Feldeigenschaft 'Duplizierfeld' werden automatisch dupliziert. Das heißt, FORMANT fordert sie nicht zur Eingabe an, sondern füllt sie sofort mit dem entsprechenden Inhalt aus 'satz'.

Schreibmarke positionieren - set cursor

Der Aufruf positioniert die Schreibmarke auf eine beliebige Stelle innerhalb der Datenfelder der Maske.

n_sec(sd,fieldname,offset)

Parameter:

→ int sd	Sitzungs-Deskriptor.
→ char *fieldname	Name des Feldes, in das die Schreibmarke zu positionieren ist. Anzugeben ist der Name, wie er mit FORMANTGEN oder 'insert field' definiert wurde. Wenn Sie einen Zeiger auf NULL angeben, nimmt FORMANT das aktuelle Feld.
→ char offset	Position innerhalb des angegeben Feldes. Feldanfang ist 0. Beachten Sie, daß offset in der Länge 1 Byte anzugeben ist (char). Bei ungültiger Angabe wird 0 angenommen.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENOF	4	Feldname nicht bekannt.
ENOF1	32	Das Feld ist vom Typ FIX.

Beispiel:

```
  .
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Ergebniswert */
int t_code;              /* logischer Code der Systemsteuertaste */
char akfeld[9];          /* Name des aktuellen Feldes */
  .
  .
ret = n_wir(sd,&t_code,akfeld);
teste ("n_wir",ret);
if (t_code == F10)       {
        ret = n_clf (sd,akfeld);
        teste ("n_wir",ret);
```

```
        ret = n_sec (sd,akfeld,0);
        teste ("n_sec",ret);
        }
```

Wird die Taste F10 gedrückt, wird der Inhalt des aktuellen Feldes gelöscht. Die Schreibmarke wird auf den Anfang des aktuellen Feldes gesetzt.

Maske abspeichern - save presentation image

Der Aufruf speichert die aktuelle Maske in einer Datei ab. Die Datei steht im aktuellen Dateiverzeichnis und erhält den angegebenen Namen der Maske. Unter diesem Namen können Sie die Maske wieder laden, wenn Sie n_cpi aufrufen. n_spi speichert alle aktuellen Feldinhalte ab, auch die der variablen Felder.

n_spi(sd,name)

Parameter:

→ int sd	Sitzungs-Deskriptor (siehe n_ops).
→ char *name	Name, unter dem die Maske abgelegt wird. Der Name ist gleich dem Namen der Datei für die Maske. Eine bestehende Datei dieses Namens wird überschrieben.

Ergebnis:

ENSA	-4	Kein Speicherplatz vorhanden.
ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENOF	4	Kein Feld vorhanden.
ESAV	512	Abspeichern nicht erfolgreich.
ENFL	4096	Fehler in der Maske.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi.

Beispiel:

```
  .
int sd;                    /* Sitzungs-Deskriptor */
int ret;                   /* Ergebniswert */
  .
ret = n_spi (sd,"maske1.s")
teste ("n_spi",ret);
  .
```

Der Aufruf speichert die aktuelle Maske in der Datei 'maske1.s' ab. Sie können diese Maske z.B. später mit 'create presentation image' erneut laden (einschließlich der aktuellen Feldinhalte).

Maske ausgeben und Daten einlesen - write read

Der Aufruf ist eine Kombination der Aufrufe 'write' und 'read'. Er bewirkt nacheinander:

1.Schreiben: Die Maske wird entsprechend den aktuellen Parametern und Feldinhalten ausgegeben bzw. abgeändert (Funktion von 'write'). Diese Funktion ist in Version 1.0 unwirksam (siehe 'write').

2.Lesen: FORMANT erwartet die Eingabe in die Datenfelder der Maske. FORMANT prüft und verarbeitet die Eingabe entsprechend den aktuell definierten Bedingungen (siehe Abschnitt 2.3). Das ist die Funktion von 'read'.

Anschließend erhält das Programm die Steuerung zurück.

n_wir(sd,r_sys,fieldname)

Parameter:

→ int sd — Sitzungs-Deskriptor.

← int *r_sys — Zeiger auf den logischen Code der Systemsteuertaste.
Dieser Wert gibt an, wodurch die Eingabe der Daten beendet wurde. Folgende Werte sind möglich:

Wert	Bedeutung	Taste
AUSG	Feld bzw. letztes Feld der Maske ausgefüllt, je nach Betriebsart 'feldweise' oder 'satzweise' (siehe Abschnitt 2.3).	
FEPLUS	Positives Feldende.	FE+
FEMINUS	Negatives Feldende.	FE-
HOME	Sprung auf erstes Feld.	HOME
NEXT	Sprung auf nächste Zeile.	NEXT
LAST	Sprung auf vorige Zeile.	LAST

BACKWARDS	Sprung auf voriges Feld.	BACKWARDS
FORWARDS	Sprung auf nächstes Feld.	FORWARDS
LVD	Lösche Datenfelder.	LVD
SEND	Satzende.	SEND
PRINT	akt. Maske drucken.	PRINT
F1 - F27	Funktionstasten.	F1 bis F27
MENU	Funktionstaste.	MENU
HELP	Funktionstaste.	HELP
START	Funktionstaste.	START
END	Funktionstaste.	END
CARDBAD	Ausweis konnte nicht gelesen werden.	
CARDOUT	Ausweis wurde aus dem Leser entnommen.	

Zur Wirkung der einzelnen Systemsteuertasten siehe Abschnitt 2.3.

← char *fieldname — Name des aktuellen Feldes. Dies ist dasjenige Feld der Maske, das FORMANT als letztes bearbeitet hat.

Ergebnis:

ERR	-32	Ablauffehler: ein früher aufgetretener Fehler wurde erkannt.
ENSY	-16	Unbekanntes Systemsteuerzeichen: ein früher aufgetretener Fehler wurde erkannt. Möglicherweise stack überschrieben.
ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ENUP	2	Zeiger auf Null bei fieldname oder r_sys.
ENOF	4	Ablauffehler: ein früher aufgetretener Fehler wurde erkannt.
APL	8	Unterbrechung durch Benutzerroutine.
EPOS	64	Positionierungsfehler: ein früher aufgetretener Fehler wurde erkannt.
FFELD	256	Erstes Feld in der Bearbeitungsreihenfolge erreicht (nur bei feldweisem Betrieb).

ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi oder Aufruf in einer Benutzerroutine.

Hinweise:

- Die Steuerung geht an das Programm zurück, wenn
  - ein Feld fertig ausgefüllt ist (bei feldweiser Eingabereaktion). Das erkennen Sie am logischen Code AUSG.
  - das letzte Feld der Maske ausgefüllt ist (bei satzweiser Eingabereaktion). Das erkennen Sie ebenfalls am logischen Code AUSG.
  - eine Systemsteuertaste gedrückt wurde. Das erkennen Sie an einem logischen Code ungleich AUSG.
  - eine Benutzerroutine mit dem Rückgabewert APL verlassen wurde (siehe Kapitel 7). Das erkennen Sie am Ergebnis APL.
- Wurde bei feldweisem Betrieb das letzte Feld der Maske ausgefüllt, liefert der Aufruf das Ergebnis FFELD.

Beispiel:

```
.
int sd;                 /* Sitzungs-Deskriptor */
int ret;                /* Ergebniswert */
int t_code;             /* logischer Code der Systemsteuertaste */
char akfeld[9];         /* Name des aktuellen Feldes */
  .
  .
do    {                                       /* Leseschleife */

      ret = n_wir (sd,&t_code,akfeld);        /* Ein- Ausgabe */
      teste ("n_wir",ret);

      switch (t_code) {
          case FORWARDS:
          case NEXT    :
          case FEPLUS  :
          case AUSG    : ....  ; break;
               .
               .
          case SEND    : ....  ; break;
               .
          case END     : break;
          case default : ....  ; break;
          }

      }  while (t_code != END);
  .
```

Die Ein- Ausgabe wird in einer Schleife durchlaufen. Die Verarbeitung ist abhängig vom logischen Code der Systemsteuertaste. Wird die Taste END gedrückt, bricht die Schleife ab.

Den Speicherplatz für den Namen des aktuellen Feldes müssen Sie immer zur Verfügung stellen, auch wenn Sie nicht auf den Inhalt zugreifen.

Maske ausgeben - write

Der Aufruf ist in V1.0 unwirksam. Alle Aufrufe, die eine Veränderung der Maske bewirken, wirken in dieser Version direkt, z.B. 'change record' usw. In späteren Versionen wirkt kein Aufruf mehr direkt auf den Bildschirm, sondern erst bei einem Schreibaufruf: 'write' oder 'write_read'.

Zukünftige Funktion:

Schreiben: Die Maske wird entsprechend den aktuellen Parametern und Feldinhalten ausgegeben bzw. abgeändert. Anschließend erhält das Programm die Steuerung zurück.

n_wrt(sd)

Parameter:

→ int sd	Sitzungs-Deskriptor.

Ergebnis:

ENSD	-2	Sitzungs-Deskriptor nicht bekannt.
NOERR	0	Aufruf erfolgreich.
ESEQ	8192	Falsche Reihenfolge: kein n_ops oder kein n_cpi.

Beispiel:

```
   .
int sd;                    /* Sitzungs-Deskriptor */
int ret;                   /* Ergebniswert */
   .
   .
ret = n_wrt (sd);
teste ("n_wrt",ret);
   .
```

7 Benutzerroutinen

Eine Benutzerroutine ist ein Unterprogramm (in COBOL) bzw. eine Funktion (in C). Eine Benutzerroutine rufen Sie im Anwendungsprogramm nicht selbst auf. Sie wird von FORMANT aufgerufen, während FORMANT die Kontrolle über die Eingabe hat (passive Schnittstelle).

Es gibt zwei Möglichkeiten, wann eine Benutzerroutine aufgerufen wird:

1. Systemsteuertaste: Der Bediener hat eine Systemsteuertaste gedrückt, der Sie eine Benutzerroutine zugeordnet haben. FORMANT ruft dann die zugeordnete Benutzerroutine auf.

2. Triggerfeld: Der Bediener hat ein Feld ausgefüllt, das als Triggerfeld definiert ist. Bei Verlassen des Feldes ruft FORMANT die Benutzerroutine auf, deren Namen Sie dem Feld zugeordnet haben.

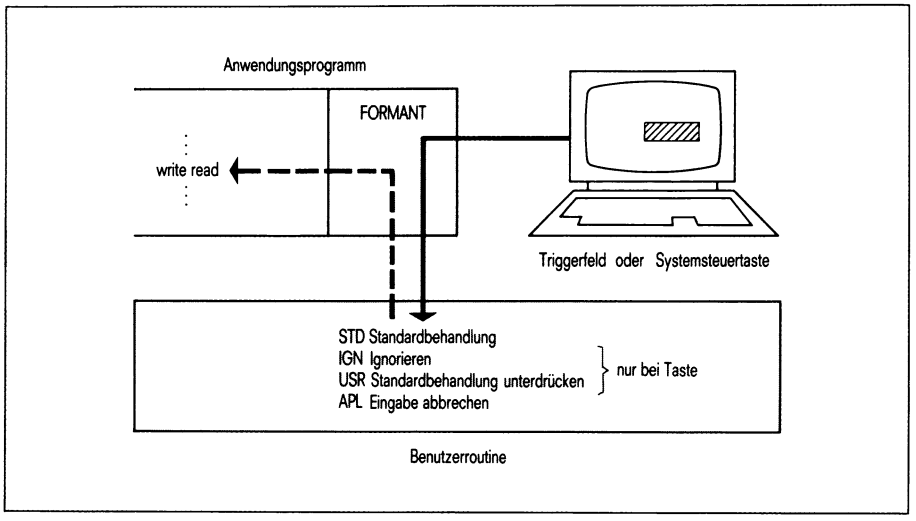


Bild 7-1: Aufruf von Benutzerroutinen

Treffen beide Fälle zusammen, wird erst die Benutzerroutine für die Systemsteuertaste durchlaufen, dann die für das Triggerfeld.

Wie ordnen Sie Benutzerroutinen zu?

Systemsteuertasten: Aufruf 'modify input reaction table'.

Triggerfelder: Aufruf 'define user exits'.

Was geschieht, wenn die Benutzerroutine durchlaufen wurde?

Sie geben einen von vier möglichen Parameterwerten zurück:

bei COBOL im Feld nu-reaction,

bei C mit return.

Die möglichen Angaben sind:

COBOL	C	Bedeutung bei Systemsteuertasten
E-STD	STD	Standardfunktion der Taste ausführen. Wurde z.B die Taste [CE] (FE+) gedrückt, bereitet FORMANT das Feld auf und die Schreibmarke steht im nächsten Feld. Wurde z.B. eine Funktionstaste gedrückt, geht die Steuerung an die Anwendung und der Leseaufruf liefert den Tastencode. Beim nächsten Leseaufruf setzt FORMANT die Eingabe mit dem nächsten Feld fort.
E-APL	APL	Eingabe abbrechen. Das Anwendungsprogramm erhält die Steuerung. Der Leseaufruf liefert den logischen Code der Taste. Das Ergebnis des Leseaufrufs ist N-APL (bzw. APL bei C). Beim nächsten Leseaufruf setzt FORMANT die Eingabe mit dem nächsten Feld fort.
E-USR	USR	Standardfunktion nicht ausführen. FORMANT führt die Reaktion nicht aus, die der Taste standardmäßig zugeordnet ist. Beispiel Taste [CE]: FORMANT bereitet das Feld nicht auf. Die Schreibmarke steht im nächsten Feld. Beispiel Funktionstaste: Die Anwendung erhält die Steuerung nicht. Die Schreibmarke steht im nächsten Feld.
E-IGN	IGN	Taste ignorieren. FORMANT meldet 'E1 Systemsteuerzeichen unzulässig!'. Das aktuelle Feld bleibt unverändert, ebenso die Position der Schreibmarke.

		Bedeutung bei Triggerfeldern
E-STD	STD	Eingabe fortsetzen. Die Steuerung geht nicht an die Anwendung. FORMANT setzt die Eingabe mit dem nächsten Feld fort.
E-APL	APL	Eingabe abbrechen. Das Anwendungsprogramm erhält die Steuerung. Der Leseaufruf liefert den logischen Code der Taste. Das Ergebnis des Leseaufrufs ist N-APL (bzw. APL bei C). Beim nächsten Leseaufruf setzt FORMANT die Eingabe mit dem nächsten Feld fort.
E-USR	USR	wie STD.
E-IGN	IGN	wie STD.

Mit welchem Feld die Eingabe fortgesetzt wird, können Sie auch mit dem Aufruf 'set cursor' bestimmen (außer bei IGN).

Welche Aufrufe sind in einer Benutzerroutine verboten?

Folgende Aufrufe dürfen Sie nicht verwenden:

close session, create p.i., destroi p.i., modify environment par., modify session par., open session, read, write read

Der Ergebniswert eines solchen Aufrufs ist ESEQ (Falsche Reihenfolge).

7.1 Benutzerroutinen in COBOL

Eine Benutzerrroutine schreiben Sie als COBOL-Unterprogramm. Dieses Unterprogramm übersetzen Sie wie üblich. Die Zwischencodedatei muß im selben Dateiverzeichnis liegen, wie die Zwischencodedatei des Anwendungsprogramms. Der Programmname darf maximal 8 Zeichen lang sein (ohne .INT), denn er muß in das Feld N-USER-REACT passen. Dieses Feld ist bei den Aufrufen 'define user exits' (N-DUE) und 'modify input reaction table' (N-MIR) anzugeben.

Programmrahmen einer Benutzerroutine

```
IDENTIFICATION DIVISION.
    .
    .
    .
DATA DIVISION.
LINKAGE SECTION.
copy "FORUP.LIB" in "/usr/lib/cobol/formant/copy".
    .
    .
    .
PROCEDURE DIVISION USING N-USER-PAR.
    .
    .
    .  alle FORMANT-Aufrufe lt. Tabelle in Kap. 7
    .
    .
EXIT PROGRAM.
```

Parameter:

Sie haben über N-USER-PAR Zugriff auf folgende Parameter.

Eingangparameter:

```
01  N-USER-PAR          enthalten im COPY-Element FORUP.LIB.
→   05  NU-SESSION-ID        PIC 99.
→   05  NU-VERSION           PIC 999.
→   05  NU-RET-SYS           PIC 999.
→   05  NU-AKT-FELD          PIC X(8).
→   05  NU-PI-NAME           PIC X(8).
→   05  NU-PI-MODE           PIC X.
```

Ausgangsparameter:

```
←
01  NU-REACTION             PIC XXX.
```

Erklärung der Parameter:

NU-SESSION-ID — Sitzungsdeskriptor. Dieser ist bei FORMANT-Aufrufen in der Benutzerroutine anzugeben.

NU-VERSION — FORMANT-Version, wie im Anwendungsprogramm angegeben. Diese Angabe wird in Version 1.0 nicht gebraucht.

NU-RET-SYS — Logischer Code der Systemsteuertaste. Eine Tabelle der möglichen Werte finden Sie beim Aufruf 'write read'.

NU-AKT-FELD — Name des aktuellen Feldes. Das ist dasjenige Feld der Maske, das FORMANT zuletzt bearbeitet hat.

NU-PI-NAME — Name der aktuellen Maske.

NU-PI-MODE — Angabe, ob die aktuelle Maske neu oder vorhanden war (siehe Aufruf N-CPI).

Ausgangsparameter:

NU-REACTION — Hier übergeben Sie einen der Werte aus der folgenden Tabelle.

Werte, die Sie in NU-REACTION an FORMANT zurückgeben:

Wert	bei Systemsteuertaste	bei Triggerfeld
STD	Standardfunktion ausführen.	Eingabe fortsetzen.
APL	Eingabe abbrechen.	Eingabe abbrechen.
USR	Standardfunktion nicht ausführen.	wie STD.
IGN	Taste ignorieren.	wie STD.

Die anschließende Reaktion von FORMANT hängt davon ab, ob die Benutzerroutine durch eine Systemsteuertaste ausgelöst wurde oder durch ein Triggerfeld. Die ausführliche Beschreibung der Reaktion finden Sie in Kapitel 7: "Was geschieht, wenn ...".

Hinweis:

Weitere Daten können Sie mit dem Anwendungsprogramm nicht austauschen. Die Benutzerroutine wird direkt von FORMANT aufgerufen, deshalb ist eine Datenverbindung nur über das COPY-Element FORUP möglich.

Beispiel:

Das folgende Beispiel zeigt, wie man je eine Benutzerroutine für Systemsteuertasten und für Triggerfelder anschließt.

Anwendungsprogramm

Das Beispiel 1 aus Kap. 4 ist wie folgt zu ergänzen:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.             beispielbr.
   .
   .
 WORKING-STORAGE SECTION.
*****************************************************************
*  Die COPY-Elemente enthalten die Datenbereiche fuer FORMANT:   *
*    FORGP  Globale Parameter                                    *
*    FORSK  FORMANT-Steuertasten                                 *
*    FORCP  FORMANT-Ereignisparameter                            *
*    FORER  FORMANT-Rueckmeldungen                               *
*    FORVA  FORMANT-Parameterwerte                               *
*****************************************************************
  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORSK.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORCP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".

   .
   .
*
 PROCEDURE DIVISION.
*****************************************************************
*  Steuerteil                                                   *
*  Alle noetigen FORMANT Funktionen laufen nacheinander ab      *
*****************************************************************
*
 ablauf section.

    display init-par at 0505.
    open extend adresse.
    perform sitzung-oeffnen.
    perform maske-laden.
    perform rout-init.
    perform einaus-maske until ende = "J".
    perform maske-loeschen.
    perform sitzung-schliessen.
    close adresse.
    display end-par at 0505.

    STOP RUN.
   .
   .
```

```
*****************************************************************
*  Benutzerroutinen zuordnen:                                   *
*  trout    fuer Triggerfeld tfeld1 und tfeld2,                 *
*  f10rout  fuer Taste F10.                                     *
*****************************************************************
*
 rout-init section.

     move "tfeld1" to n-event-feld.
     move "trout" to n-user-react.
     move "N-DUE" to f-aufruf.

     call f-aufruf using n-main-par, n-event-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     move "tfeld2" to n-event-feld.

     call f-aufruf using n-main-par, n-event-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.

     move n-f10 to n-event-key.
     move e-usr to n-reaction.
     move "f10rout" to n-user-react.
     move "N-MIR" to f-aufruf.

     call f-aufruf using n-main-par, n-event-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.

     cancel f-aufruf.

*****************************************************************
*  Maske ausgeben und Daten einlesen.                           *
*  N-WIR fordert die Dateneingabe an.                           *
*****************************************************************
*
 einaus-maske section.

     move "N-WIR" to f-aufruf.
     call f-aufruf using n-main-par, n-pi-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.

*    *** Abfrage auf Benutzerroutine.   ***
*
     if n-ret-code = n-apl
                  perform trout-reakt
```

```
     else if n-ret-code not = n-noerr
                    perform fehler.
     cancel f-aufruf.

*    *** Abfrage der Systemsteuertaste. ***
*
     if  n-ret-sys = n-ausg
                 or n-next
                 or n-forwards
                 or n-feminus
                 or n-feplus

*    *** Status-Meldung, wenn fertig ausgefüllt ***
*
                 move "Fertig. Abspeichern mit F14"
                      to n-status-message

                 perform melde

*    *** Satz schreiben, wenn F14 (SEND) gedrueckt ***
*
     else if n-ret-sys = n-send
                 perform satz-schreiben

*    *** Programmende, wenn END gedrueckt ***
*
     else if n-ret-sys = n-end
                 move "J" to ende
*    *** Meldung, wenn falsche Taste gedrueckt wurde. ***
*
     else  move "Taste nicht definiert" to n-status-message
                 perform melde.

*****************************************************************
*  Reaktion auf Benutzerroutine.                                *
*  Zur Demonstration wird eine Fehlermeldung ausgegeben.        *
*****************************************************************
*
 trout-reakt section.

     move "Eingabe abgebrochen" to n-error-message.

     move "N-CEM" to f-aufruf.
     call f-aufruf using n-main-par, n-pi-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
     if n-ret-code not = n-noerr
                  perform fehler.
     cancel f-aufruf.
   .
   .
   .
```

Benutzerroutine trout

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.              trout.
*
*
*
 ENVIRONMENT DIVISION.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
*****************************************************************
*  Datei fuer Fehlerprotokoll                                   *
*****************************************************************
*
 select fehl assign "f-datei"
            organization is line sequential.

*
 DATA DIVISION.
 FILE SECTION.

*****************************************************************
*  Datensatz fuer Fehlerprotokoll                               *
*****************************************************************
*
 FD fehl.
 01 f-info.
    05 f-meld              pic x(19).
    05 f-aufruf            pic x(5).
    05 filler              pic x.
    05 f-code              pic -99999.
*
 WORKING-STORAGE SECTION.

*****************************************************************
*  Die COPY-Elemente enthalten die Datenbereiche fuer FORMANT:  *
*    FORGP  Globale Parameter                                   *
*    FORER  FORMANT-Rueckmeldungen                              *
*    FORVA  FORMANT-Parameterwerte                              *
*****************************************************************

  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".

  77 fehlerart                  pic x(8).

*
 LINKAGE SECTION.

*****************************************************************
*  COPY-Element fuer Benutzerroutine.                           *
*****************************************************************

  copy "FORUP.LIB" in "/usr/lib/cobol/formant/copy".

*
 PROCEDURE DIVISION USING N-USER-PAR.
```

```
*********************************************************************
*  Sitzungsdeskriptor versorgen.                                    *
*********************************************************************
*
 init section.

  move nu-session-id to n-session-id.

*********************************************************************
*  Verarbeitung bei tfeld1: Eingabe fortsetzen.                     *
*  Verarbeitung bei tfeld2: Eingabe abbrechen.                      *
*  Zur Demonstration wird jeweils eine Meldung ausgegeben.          *
*********************************************************************
*
 steuer section.

  if nu-akt-feld = "tfeld1"

     perform v-tfeld1

  else

     perform v-tfeld2.

  EXIT PROGRAM.

*********************************************************************
*  Verarbeitung bei tfeld1: Eingabe fortsetzen.                     *
*********************************************************************
*
 v-tfeld1 section.
     move "trout: Eingabe fortsetzen." to n-status-message.

     move "N-CSM" to f-aufruf.
     call f-aufruf using n-main-par, n-pi-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.

     if n-ret-code not = n-noerr
                  perform fehler.

     cancel f-aufruf.

     move e-std to nu-reaction.

*********************************************************************
*  Verarbeitung bei tfeld2: Eingabe abbrechen.                      *
*********************************************************************
*
 v-tfeld2 section.

     move "trout: Eingabe abbrechen." to n-status-message.

     move "N-CSM" to f-aufruf.
     call f-aufruf using n-main-par, n-pi-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.
```

```
     if n-ret-code not = n-noerr
                  perform fehler.

     cancel f-aufruf.

     move e-apl to nu-reaction.

*****************************************************************
*  Fehler-Routine.                                              *
*  Die Routine wird aufgerufen, wenn ein FORMANT-Aufruf wegen   *
*  Speichermangel nicht ausgefuehrt werden kann (overflow)      *
*  oder mit einem Fehlercode ungleich N-NOERR zurueckkehrt.     *
*  Aufruf und Fehlercode werden in die Datei f-datei geschrieben.*
*****************************************************************
*
 fehler section.
     .
     .
     .   wie im Anwendungsprogramm
     .
```

Benutzerroutine f10rout

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.             f10rout.
*
*
*
 ENVIRONMENT DIVISION.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
*****************************************************************
*  Datei fuer Fehlerprotokoll                                   *
*****************************************************************
*
 select fehl assign "f-datei"
             organization is line sequential.

*
 DATA DIVISION.
 FILE SECTION.

*****************************************************************
*  Datensatz fuer Fehlerprotokoll                               *
*****************************************************************
*
 FD fehl.
 01 f-info.
    05 f-meld              pic x(19).
    05 f-aufruf            pic x(5).
    05 filler              pic x.
    05 f-code              pic -99999.
*
 WORKING-STORAGE SECTION.
```

```
*****************************************************************
*  Die COPY-Elemente enthalten die Datenbereiche fuer FORMANT:   *
*    FORGP  Globale Parameter                                    *
*    FORER  FORMANT-Rueckmeldungen                               *
*    FORVA  FORMANT-Parameterwerte                               *
*****************************************************************

  copy "FORGP.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORER.LIB" in "/usr/lib/cobol/formant/copy".
  copy "FORVA.LIB" in "/usr/lib/cobol/formant/copy".

  01 meldung.
     05 meld-text                pic x(9)   value "f10rout: ".
     05 meld-feld                pic x(8).

  77 fehlerart                   pic x(8).

*
 LINKAGE SECTION.

*****************************************************************
*  COPY-Element fuer Benutzerroutine.                           *
*****************************************************************

  copy "FORUP.LIB" in "/usr/lib/cobol/formant/copy".

*
 PROCEDURE DIVISION USING N-USER-PAR.

*****************************************************************
*  Sitzungsdeskriptor versorgen.                                *
*****************************************************************
*
 init section.

  move nu-session-id to n-session-id.

*****************************************************************
*  f10rout wird von FORMANT aufgerufen, wenn die Taste F10      *
*  gedrueckt wurde.                                             *
*  Die Anwendung soll die Steuerung nicht erhalten (e-usr).     *
*  Zur Demonstration wird eine Meldung mit dem Namen des        *
*  aktuellen Feldes ausgegeben.                                 *
*****************************************************************
*
 ablauf section.

     move nu-akt-feld to meld-feld.
     move meldung to n-status-message.

     move "N-CSM" to f-aufruf.
     call f-aufruf using n-main-par, n-pi-par
                  on overflow
                  move "OVERFLOW" to fehlerart
                  perform fehler.

     if n-ret-code not = n-noerr
                  perform fehler.

     cancel f-aufruf.
```

```
      move e-usr to nu-reaction.

    EXIT PROGRAM.

****************************************************************
*  Fehler-Routine.                                             *
*  Die Routine wird aufgerufen, wenn ein FORMANT-Aufruf wegen  *
*  Speichermangel nicht ausgefuehrt werden kann (overflow)     *
*  oder mit einem Fehlercode ungleich N-NOERR zurueckkehrt.    *
*  Aufruf und Fehlercode werden in die Datei f-datei geschrieben.*
****************************************************************
*
 fehler section.
     .
     .
     .   wie im Anwendungsprogramm
     .
```

7.2 Benutzerroutinen in C

Eine Benutzerrroutine schreiben Sie als Funktion vom Typ int. Es gelten die üblichen Konventionen.

Definition einer Benutzerroutine

```
int f_name (taste,feld)

int taste;              /* log. Code der Systemsteuertaste */
char *feld;             /* aktuelles Eingabefeld */
{
   .
   .
   .
return ( Ergebniswert )
}
```

Übergebene Argumente

taste — Logischer Code der Systemsteuertaste. Eine Tabelle der möglichen Werte finden Sie beim Aufruf 'write read'.

fname — Name des aktuellen Feldes. Das ist dasjenige Feld der Maske, das FORMANT zuletzt bearbeitet hat.

Ergebniswert

Mit return übergeben Sie einen der Werte:

Wert	bei Systemsteuertaste	bei Triggerfeld
STD	Standardfunktion ausführen.	Eingabe fortsetzen.
APL	Eingabe abbrechen.	Eingabe abbrechen.
USR	Standardfunktion nicht ausführen.	wie STD.
IGN	Taste ignorieren.	wie STD.

Die anschließende Reaktion von FORMANT hängt davon ab, ob die Benutzerroutine durch eine Systemsteuertaste ausgelöst wurde oder durch ein Triggerfeld. Die ausführliche Beschreibung der Reaktion finden Sie in Kapitel 7: "Was geschieht, wenn ...".

Hinweise:

- Der Sitzungsdeskriptor ist global zu definieren, wenn in der Benutzerroutine FORMANT-Aufrufe benutzt werden.
- Sie können über globale Variablen beliebig Werte übergeben.

Beispiel:

Für dieses Beispiel wurde das Beispiel 1 aus Kap. 4 ergänzt.

```
/*********************************************************************
Das Programm brout zeigt den Anschluss von Benutzerroutinen.
tr_rout wird den Triggerfeldern tfeld1 und tfeld2 zugeordnet.
f10_rout reagiert auf die Taste F10.
*********************************************************************/

#include <formant.h>     /* Definitionen für FORMANT */
#include <stdio.h>
#define SATZLAENGE 80    /* max. Satzlänge = Länge var. Daten in der Maske */
int sd;                  /* Sitzungs-Deskriptor */
int ret;                 /* Rueckgabewert */

main (argc, argv)
int argc;
char *argv[];   {

int version = 100;       /* FORMANT-Version 1.00 */
int t_code;              /* logischer Code der Systemsteuertaste */
char akfeld[9];          /* aktuelles Feld */
char satz[SATZLAENGE+1];/* Datensatz */
int s_laenge;            /* Länge der übergebenen Daten */
FILE *a_datei;           /* Ausgabedatei */
```

```
int tr_rout();          /* Benutzerroutine fuer Triggerfeld */
int f10_rout();         /* Benutzerroutine fuer Taste F10 */


                                    /* Ausgabedatei oeffnen */
if ( (a_datei = fopen("p-datei","a")) == NULL ) {
        fprintf (stderr,"%s","Datei p-datei kann nicht geoeffnet werden");
        exit (300);
        }

                                    /* Umschalten auf 25-Zeilen-Modus */
printf ("Programm %s geladen\33[0u",argv[0]);

ret = n_ops (version,&sd);          /* Sitzung oeffnen */
teste ("n_ops",ret);

                                    /* Maske laden */
ret = n_cpi (sd,"maskep",VORHANDEN);
teste ("n_cpi",ret);

                                    /* Dem Triggerfeld tfeld1 die Be- */
ret = n_due (sd,"tfeld1",tr_rout);  /* nutzerrout. tr_rout zuordnen   */
teste ("n_due",ret);

                                    /* Dem Triggerfeld tfeld2 die Be- */
ret = n_due (sd,"tfeld2",tr_rout);  /* nutzerrout. tr_rout zuordnen   */
teste ("n_due",ret);

                                    /* Der Taste F10 die Benutzer-    */
ret = n_mir (sd,F10,USR,f10_rout);  /* routine f10_rout zuordnen      */
teste ("n_due",ret);
do      {                           /* Leseschleife */

        ret = n_wir (sd,&t_code,akfeld);/* Ein- Ausgabe */
        if (ret == APL) {
            ret = n_cem (sd,"Eingabe abgebrochen.");
            teste ("n_cem",ret);
            }
        else
            teste ("n_wir",ret);

        switch (t_code) {
            case FORWARDS:
            case NEXT    :
            case FEPLUS  :
            case FEMINUS :
            case AUSG    :      {       /* wenn fertig ausgefuellt, */
                                        /* Meldung  */
                ret = n_csm (sd,"Fertig. Abspeichern mit F14.");
                teste ("n_csm",ret);
                break;
                }

            case SEND    :      {

                                    /* Datensatz aufbauen */
                ret = n_ger (sd,satz,&s_laenge);
                teste ("n_ger",ret);

                                    /* Datensatz schreiben */
                fprintf(a_datei,"%s\n",satz);

                ret = n_clr (sd);       /* Datenfelder löschen */
                teste ("n_clr",ret);
```

```
                ret = n_csm (sd," ");   /* Statusmeldung löschen */
                teste ("n_csm",ret);

                break;
                }

            case END     : break;

            default      :      {       /* Undefinierte Taste */

                ret = n_csm (sd,"Taste nicht definiert");
                teste ("n_csm",ret);
                }
            }

        } while (t_code != END);

ret = n_dpi (sd);                       /* Maske schliessen */
teste ("n_dpi",ret);
ret = n_cls (sd);                       /* Sitzung schliessen */
teste ("n_cls",ret);

                                        /* Umschalten auf 24-Zeilen-Modus und
                                           Bildschirm löschen. */
printf("\33[2J\33[1uProgramm %s beendet\n",argv[0]);

exit (0);
}

/*************************************************************************
   Die Funktion 'teste' prüft den Rueckgabewert auf NOERR.
   Ist der Rueckgabewert =0(NOERR), wird das Programm fortgesetzt.

   Sonst gibt die Funktion eine Meldung aus und reagiert wie folgt:
   Ist der Rueckgabewert <0, bricht teste das Programm ab.
   Ist der Rueckgabewert >0, wird das Programm fortgesetzt.
*************************************************************************/

teste (aufruf,r_wert)
char *aufruf;
int r_wert;     {

if (r_wert == NOERR)
        return;
else    {
        fprintf(stderr,"Fehler bei Aufruf %s. Fehlercode %d\n",aufruf,r_wert);

        if (r_wert > NOERR)
                return;
        else    {
                n_dpi (sd);
                n_cls (sd);
                                        /* Umschalten auf 24-Zeilen-Modus und
                                           Bildschirm löschen. */
                printf("\33[2J\33[1u\n");
                exit(r_wert);
                }
        }
}
```

```
/*********************************************************************
   Die Funktion 'tr_rout' wird von FORMANT aufgerufen, wenn ein zuge-
   ordnetes Triggerfeld eingegeben wurde.

   Eingabe fortsetzen bei tfeld1 (STD).
   Eingabe abbrechen bei tfeld2 (APL).

   Zur Demonstration wird jeweils eine Meldung ausgegeben.
*********************************************************************/

tr_rout (t_code,a_feld)
int t_code;     /* Tastencode */
char *a_feld;   /* aktuelles Feld */
{

if ( !strcmp(a_feld,"tfeld1") ) {
                                        /* Verarbeitung bei tfeld1 */
        ret = n_csm (sd,"tr_rout: Eingabe fortsetzen.");
        teste ("n_csm",ret);
        return (STD);
        }

else if ( !strcmp(a_feld,"tfeld2") )    {
                                        /* Verarbeitung bei tfeld2 */
        ret = n_csm (sd,"tr_rout: Eingabe abbrechen.");
        teste ("n_csm",ret);
        return (APL);
        }
else
        n_dpi (sd);
        n_cls (sd);
        exit (99);                      /* Falsches Feld, Programmabbruch */
}

/*********************************************************************
   Die Funktion 'f10_rout' wird von FORMANT aufgerufen, wenn die Taste
   F10 gedrückt wurde.

   Die Anwendung soll die Steuerung nicht erhalten (USR).

   Zur Demonstration wird eine Meldung mit dem Namen des aktuellen
   Feldes ausgegeben.
*********************************************************************/

f10_rout (t_code,a_feld)
int t_code;     /* Tastencode */
char *a_feld;   /* aktuelles Feld */
{

char meld[40];

sprintf (meld,"f10_rout: %s",a_feld);

ret = n_csm (sd,meld);
teste ("n_csm",ret);

return (USR);
}
```

8 Das Drumherum

In diesem Kapitel finden Sie:

- Übersetzen und Binden der Anwendungsprogramme.
  - Übersetzen von COBOL-Programmen
  - Prozedur zum Übersetzen und Binden von C-Programmen
  - Hinweise zum Programmtest
- die SINIX-Umgebung.
  - Welche Dateien benutzt FORMANT?
  - Wo liegen diese Dateien?
  - Termcap-Eintrag
- die Installation von FORMANT.
  - Wie Sie installieren
  - Was Sie evtl. dann noch tun müssen
  - Laufzeitroutinen bei COBOL
- Tips und Tricks
  - Was tun Sie, wenn ...

8.1 Übersetzen und Binden, Programmtest

FORMANT-Anwendungen werden grundsätzlich behandelt wie jedes andere Programm in COBOL oder in C. Was Sie berücksichtigen müssen, finden Sie in den folgenden Abschnitten.

8.1.1 COBOL-Anwendungen

Übersetzen des Anwendungsprogramms:

```
cobol progname.CBL
```

progname.CBL	Name des Anwendungsprogramms oder einer Benutzerroutine.

Zu übersetzen sind das Anwendungsprogramm und alle Benutzerroutinen, sofern Sie solche verwenden. Im Menüsystem übersetzen Sie ebenfalls wie gewohnt.

FORMANT benutzt ein eigenes Laufzeitsystem rts2. Dieses verwendet auch der Übersetzer. Versichern Sie sich, daß Sie mit dem richtigen Laufzeitsystem übersetzen (siehe Abschnitt 8.3, Installation).

Programmtest mit ANIMATOR:

Für den Programmtest stellt FORMANT ein eigenes Laufzeitsystem für den ANIMATOR bereit (rts2a, siehe Abschnitt 8.3, Installation).

Wenn Sie Benutzerroutinen verwenden, müssen Sie diese ebenfalls mit der Compileranweisung ANIM übersetzen.

Bei allen CALL-Aufrufen ist die Namenserweiterung .INT anzugeben und alle Zwischencode-Dateien müssen im aktuellen Dateiverzeichnis vorhanden sein. Sonst meldet ANIMATOR Laufzeitfehler 173.

Wenn Sie ANIMATOR beenden, ohne daß der Aufruf N-CLS durchlaufen wurde, kann Ihre Datensichtstation in einem merkwürdigen Zustand sein. Das beheben Sie, wie in Abschnitt 2.1.3 beschrieben.

Einschränkungen beim Programmtest:

- Der Bildaufbau für FORMANT wird von ANIMATOR nicht abgefangen. Die FORMANT-Maske zerstört das ANIMATOR-Bild. Abhilfe: Eingabe eines ANIMATOR-Kommandos, das das ANIMATOR-Bild neu aufbaut, z.B. erst U, dann E.
- ANIMATOR führt die Anweisung CANCEL nicht aus. Abhilfe: wenn möglich, nur die notwendigen (N-OPS, N-CPI ..) und die 'kritischen' Aufrufe durchlaufen.

Notausgang: Wenn ANIMATOR nach einem Fehler nicht mehr reagieren sollte, beenden Sie ihn mit [CTRL] [\].

8.1.2 C-Anwendungen

Zum Übersetzen und Binden gibt es eine Shell-Prozedur. Aufruf:

formantmake [programmname 'modulname ... ']

programmname — Name des zu erzeugenden Anwendungsprogramms (SINIX-Konventionen für Dateinamen).

modulname

Name des einzubindenden Anwendungsmoduls mit der Namenserweiterung '.o'. Ist das zugehörige Quellprogramm noch nicht übersetzt oder neueren Datums, veranlaßt die Prozedur die Übersetzung.
Anzugeben sind alle Module, die Sie benötigen (Anwendungsprogramm, Benutzerroutinen usw), nicht aber die Module, die FORMANT benötigt.

Keine Parameter angegeben:

formantmake fragt, ob Sie eine Beschreibung ausgegeben haben wollen und verlangt anschließend die Parameter im Dialog (siehe Beispiel unten).

Die Prozedur benutzt das Kommando make, das auf die Datei 'makefile' in /usr/lib/formant zugreift. Dort finden Sie nötigenfalls die Informationen, die Sie zum Übersetzen 'von Hand' brauchen. Die Prozedur formantmake steht in /usr/bin.

Beim Übersetzen mit cc wird Schalter c verwendet, das heißt der entstehende Code ist gemeinsam benutzbar. Er wird nur einmal in den Hauptspeicher geladen, wenn mehrere Benutzer das Programm aufrufen.

Beispiel:

1. Parameter in der Kommandozeile. Die Liste der Modulnamen muß in Hochkommas eingeschlossen sein.

```
$ formantmake formprog 'beispiel1.o brout.o'
*************************************************
*************************************************
***                                           ***
***         Generierungsprozedur              ***
***                                           ***
***        F O R M A N T  V 1 . 0             ***
***                                           ***
*************************************************
*************************************************
cc  -c brout.c
ld -n /lib/crt0.o beispiel1.o brout.o /usr/lib/formant/ben.o /usr/lib/formant/
libformant.a /usr/lib/formant/libbat.a  -o formprog1  -ltermcap  -lc
Programm formprog wurde wunschgemaess erzeugt
```

2. Parameter im Dialog eingeben. Die Hochkommas bei den Modulnamen entfallen.

```
$ formantmake
*****************************************************
*****************************************************
***                                               ***
***          Generierungsprozedur                 ***
***                                               ***
***        F O R M A N T  V 1 . 0                 ***
***                                               ***
*****************************************************
*****************************************************
Wollen Sie eine Beschreibung ? (j/n) n
Programmname: formprog1
Anwendungsmodul(e): beispiel1.o brout.o
cc  -c brout.c
     .
     .  wie oben
```

In beiden Beispielen sind zwei Module einzubinden. beispiel1.c war bereits übersetzt.

Programmtest mit adb

Der Programmtest mit dem Testhilfeprogramm adb ist ohne Einschränkung möglich. Allerdings ist die Ausgabe von FORMANT und von adb nicht voneinander zu trennen. Dasgleiche gilt für die Eingabe. Daher kann es auf dem Bildschirm manchmal etwas wirr zugehen.

Wollen Sie das vermeiden, können Sie die Ein- Ausgabe für FORMANT auf eine zweite Datenstation umlenken (nicht bei einem Einplatzsystem):

```
main ( )  {

close (0);
close (1);
open ("/dev/...",0);
open ("/dev/...",1);
  .
  .
  .
```

Wenn Sie adb beenden, ohne daß der Aufruf n_cls durchlaufen wurde, kann Ihre Datensichtstation in einem merkwürdigen Zustand sein. Das beheben Sie, wie in Abschnitt 2.1.3 beschrieben.

8.1.3 Masken testen mit 'formant'.

Mit diesem Programm, das mit FORMANT mitgeliefert wird, können Sie Masken testen. Es zeigt das Bild der gewünschten Maske und fordert die Eingabe aller variablen Felder an. Haben Sie alle Felder ausgefüllt, wartet das Programm noch 20 Sekunden und beendet sich dann.

Druckermasken druckt das Programm am Drucker aus.

Aufruf:

formant

Auf die Aufforderung:

Bitte Maskenname eingeben:

geben Sie einen Maskennamen ein, evtl mit Angabe des Pfades. Anschließend geben Sie noch an, ob Sie eine Bildschirmmaske oder eine Druckermaske bearbeiten wollen:

Bildschirmmaske (b) oder Druckermaske (d):

Bei der Eingabe in die Maske laufen die mit FORMANTGEN definierten Prüffunktionen ab.

Nicht testen können Sie die Funktionen, die im Anwendungsprogramm zusätzliche Maßnahmen erfordern, z.B. Gegentastprüfen, Duplizieren usw. Das Programm erzeugt keine Datensätze. Die eingegebenen Daten gehen 'ins Leere'.

Maske ausdrucken:

PRINT — druckt die Maske mit allen aktuellen Feldinhalten aus (Hardcopy).

8.2 Die SINIX-Umgebung

Dieser Abschnitt soll Ihnen helfen, alle nötigen Dateien an die richtige Stelle zu setzen, bzw. an der richtigen Stelle zu finden.

8.2.1 FORMANT-Dateien

Folgende Programme und Prozeduren sind im Dateiverzeichnis /usr/bin:

formantgen	Interaktiver Formatgenerator.
formant	Testprogramm.
formantmake	Prozedur zum Erstellen von C-Anwendungen.

Include-Datei für C-Anwendungen:

/usr/include/formant.h

Dateien für COBOL-Anwendungen:

Im Dateiverzeichnis /usr/lib/cobol/formant liegen die Unterverzeichnisse:

copy	enthält alle COPY-Elemente.
int	enthält alle FORMANT-Zwischencodedateien.
rts	enthält die Laufzeitsysteme rts2 und rts2a, eine 'makefile', um eigene Laufzeitroutinen in die Laufzeitsysteme einzubringen und die dazu nötigen Module.

Die Datei /etc/termcap:

Die Installationsprozedur für FORMANT macht einen eigenen Eintrag in die Datei /etc/termcap. Der Eintrag ist gekennzeichnet mit:

formant|fo97801:\

FORMANT verwendet diesen Eintrag automatisch.

Von FORMANT intern verwendete Dateien:

Diese Dateien befinden sich alle in /usr/lib/formant.

8.2.2 Die Umgebung beim Ablauf eines COBOL-Programms

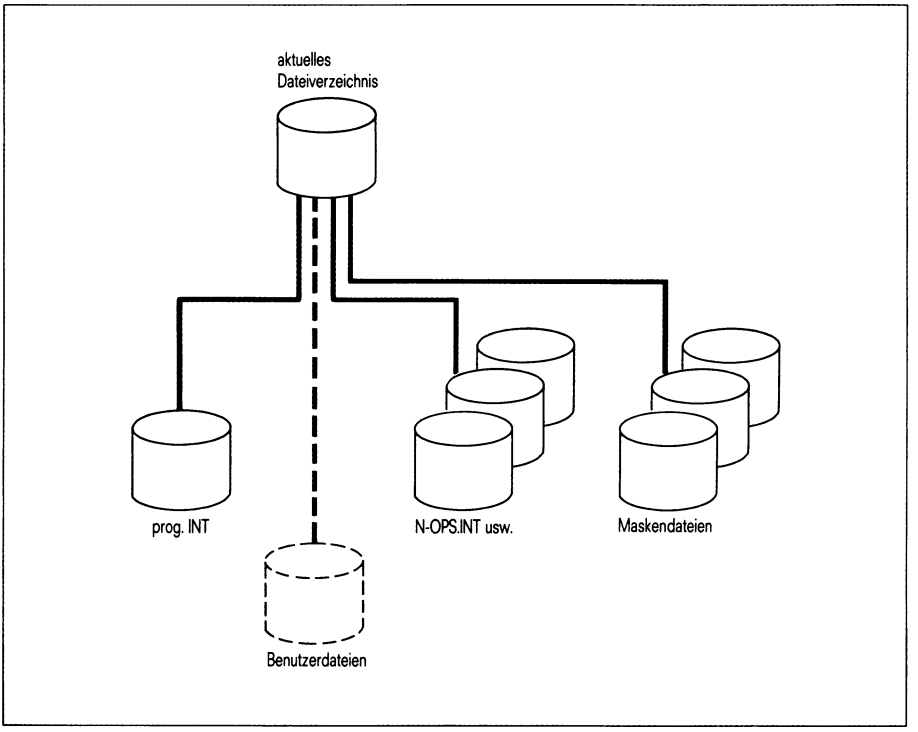


Bild 8-1: Umgebung beim Ablauf eines COBOL-Programms.

Es ist möglich, Maskendateien in andere Dateiverzeichnisse zu legen als in das aktuelle Dateiverzeichnis. Dann darf aber der Name einschließlich Pfad nur 8 Zeichen lang sein. Am einfachsten und übersichtlichsten ist es, wenn Sie Verweise benutzen.

Die Zwischencode-Dateien (N-OPS.INT usw.) müsse im aktuellen Dateiverzeichnis eingetragen sein, wenn Sie nicht beim Aufruf den vollen Pfadnamen angeben.

8.2.3 Die Umgebung beim Ablauf eines C-Programms

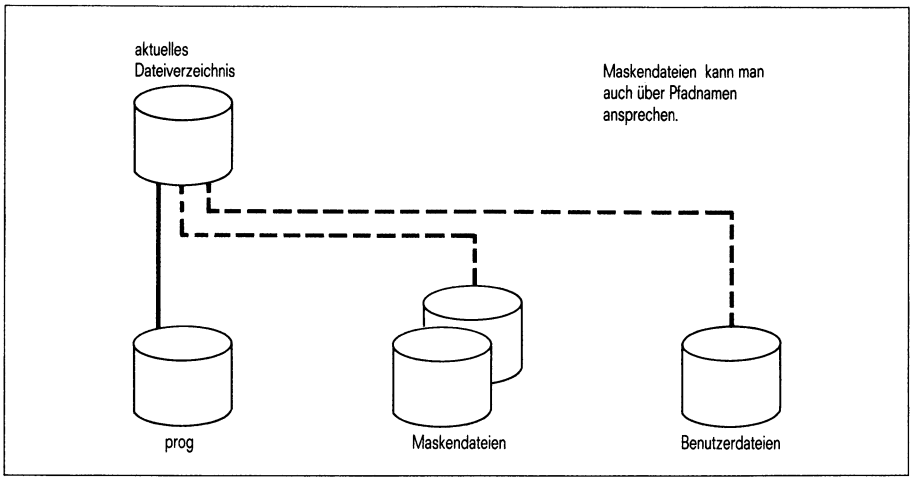


Bild 8-2: Die Umgebung beim Ablauf eines C-Programms.

8.3 FORMANT installieren

FORMANT wird auf drei Disketten ausgeliefert:

- Diskette 1 enthält die allgemeinen und die für C-Anwendungen nötigen Dateien.
- Diskette 2 enthält die englische Version von FORMANTGEN.
- Diskette 3 enthält die für COBOL-Anwendungen nötigen Teile.

FORMANT installieren Sie wie jedes andere Softwareprodukt mit der Installationsprozedur /etc/superinstall (unter Kennung root) oder im Menüsystem mit der Funktion 'Installieren von Softwareprodukten' (unter Kennung admin).

Während der Installation werden Sie gefragt:

Wollen Sie die deutsche Variante installieren ?	Eingabe: d
Do you want to install the English Version ?	Enter : e

Ihre Wahl/your choice <d/e> d
FORMANT1 - Diskette wird gelesen .....

Wenn Sie mit 'e' antworten, erhalten Sie die englische Version von FORMANT. Sie unterscheidet sich von der deutschen Version durch englischsprachige Masken bei FORMANTGEN. Sie brauchen Diskette 2 nicht einzulesen, wenn Sie englische Version nicht wollen.

Diskette 3 (COBOL) muß nur eingelesen werden, wenn mit COBOL gearbeitet werden soll.

Beachten Sie bitte: Bevor Diskette 3 eingelesen wird, muß LEVEL II COBOL installiert worden sein. Wenn später eine neue LEVEL II COBOL Version installiert wird, muß FORMANT erneut installiert werden. Sie können FORMANT nur komplett installieren, Diskette3 später einspielen geht nicht.

Was ist nach der Installation zu tun?

Für C-Anwendungen ist FORMANT fertig bereitgestellt.

Für COBOL-Anwendungen ist das Laufzeitsystem rts2 auszuwechseln wie folgt:

1. Arbeiten Sie unter root oder admin.
2. Sichern Sie das vorhandene COBOL-Laufzeitsystem in /usr/lib/cobol, z.B.

```
cd /usr/lib/cobol
mv rts2 rts2.orig
```

3. Bringen Sie das FORMANT-COBOL-Laufzeitsystem nach /usr/lib/cobol und ergänzen Sie die Zugriffsrechte:

```
ln formant/rts/rts2 .
chmod a+x rts2
```

Falls Sie mit ANIMATOR testen wollen, verfahren Sie ebenso mit dem Laufzeitsystem rts2a.

Laufzeitroutinen in rts2 und rts2a einbinden

Wenn Sie eigene Laufzeitroutinen verwenden, müssen Sie diese ins Laufzeitsystem mit einbinden, wie das im Manual 'LEVEL II COBOL Bedienungsanleitung' beschrieben ist. Wechseln Sie dazu in das Dateiverzeichnis

/usr/lib/cobol/formant/rts

und verwenden Sie die Dateien:

usercall.c und makefile.

Die Routinen zur Funktionstastenbehandlung dürfen Sie nicht einbinden.

8.4 Tips aus der Trickkiste

FORMANT allgemein:

Nach Beenden der Anwendung ist keine normale Eingabe möglich.
close session wurde nicht durchlaufen (siehe Abschnitt 2.1.3)

Keine Reaktion mehr auf Eingaben.
Sie haben vielleicht versehentlich [CTRL] [s] gedrückt. Das blockiert die Ausgabe. Drücken Sie [CTRL] [q].

Maske rutscht um eine Zeile nach oben.
25-Zeilen-Modus nicht eingeschaltet.

Speicherabzug geschrieben.
Hat eine FORMANT-Funktion einen negativen Wert geliefert? Haben Sie alle Ergebniswerte abgefragt?

Sie können die Maske nicht abspeichern.
Wahrscheinlich ist kein Speicher mehr frei oder Sie haben kein einziges Feld in der Maske definiert.

FORMANT beendet sich mit Ende-Status 3.
Fehler in der Datei /etc/termcap, z.B. Doppelpunkt vergessen.

Triggerfeld löst die zugeordnete Funktion nicht aus.
- Der Aufruf 'define user exits' wurde nicht erfolgreich durchlaufen.
- Die Maske wurde abgespeichert. In diesem Fall müssen sie Benutzerroutinen erneut zuordnen (define user exits).

Beim Einschalten des Gegentastprüfens werden die variablen Felder nicht gelöscht.
Normales Verhalten. Siehe Beispiele zum Gegentastprüfen.

In einem numerischen Feld, das mit 001 vorbelegt ist, steht nach dem Aufbereiten 100.

Das Feld ist linksbündig definiert und das Feldfüllzeichen ist Null. Dafür ist das Verhalten normal. Definieren Sie das Feld rechtsbündig.

ced zeigt keine Linien mehr, wenn eine FORMANT-Anwendung gelaufen ist.
FORMANT stellt den alternativen Zeichensatz nicht zurück. Abhilfe: [ESC] [)] [w] eingeben.

C-Programme:

Speicherabzug geschrieben.
- Parameter falsch versorgt, in dem eine Adresse erwartet wird?
- Alle Bereiche bereitgestellt, in die FORMANT Daten liefert, z.B. für den Feldnamen bei n_wir?

ld:/usr/lib/formant/libbat.a(__.SYMDEF): veraltet (Warnung)
Die genannte Bibliothek wurde vermutlich einmal kopiert. Dadurch stimmt das Datum nicht mit dem intern gespeicherten Datum überein. Das ist für die Anwendung ohne Bedeutung.

"Falscher Feldname" bei n-due.
Sind Ihre Feldnamen länger als 8 Zeichen (exkl. NULL)?

COBOL-Programme:

FORMANT-Aufrufe werden nicht ausgeführt.
CALL-Aufrufe, die nicht mehr in den Speicher passen, führt COBOL nicht aus. Wenn Sie nicht 'on overflow' verwenden, wird das Programm ohne Fehlermeldung fortgesetzt.

Laufzeitfehler 173 (Zwischencodedatei nicht gefunden).
Programm mit cbrun aufrufen (nicht mit rts2) oder CALL "name.INT". Bei Animator: CALL "name.INT" und Zwischencodedateien ins aktuelle Dateiverzeichnis legen.

cbrun meldet: can't exec.
Zugriffsrecht 'x' bei rts2 nicht gesetzt.

ANIMATOR:

Laufzeitfehler 173.
siehe unter COBOL-Programme.

Keine Reaktion mehr.
beenden mit [CTRL] [\]

FORMANTGEN:

Ihr vorhandenes Format ist nicht ladbar.
Sind Sie im richtigen Dateiverzeichnis?
Haben Sie das richtige Dateiverzeichnis angegeben?

Ihre Angabe für das Maskenverzeichnis oder das Formatverzeichnis wird abgelehnt.
Sie müssen diese Verzeichnisse vorher mit mkdir einrichten.

Keine Reaktion mehr auf Eingaben.
Sie haben vielleicht versehentlich [CTRL] [s] gedrückt. Das blockiert die Ausgabe. Drücken Sie [CTRL] [q].

Maske kann nicht generiert werden. Mögliche Ursachen sind:

- Zu viele Felder
- Feldnamen doppelt vorhanden
- Feldname gelöscht
- Attributangaben: Standardvorgabe gelöscht

t0file oder t1file.
Diese Dateien können im aktuellen Dateiverzeichnis eingetragen sein, wenn bei FORMANTGEN ein interner Fehler aufgetreten ist (z.B. Maske konnte nicht generiert werden). Aktion wiederholen. Die Dateien können Sie löschen.

Ein Feld konnte nicht eingefügt werden.
Beim Generieren einer Druckermaske müssen alle Felder unterschiedliche Namen haben (siehe Abschnitt 3.3.8).

Literatur

Betriebssystem SINIX
Buch 1
U1901-J-Z95-3

Betriebssystem SINIX
CES
vorläufige Ausgabe

Betriebssystem SINIX
LEVEL II COBOL Sprachbeschreibung
U2053-J-Z95-1

Betriebssystem SINIX
LEVEL II COBOL Bedienungsanleitung
U2051-J-Z95-1

Betriebssystem SINIX
ANIMATOR
U2222-J-Z95-1

Stichwörter

Anhang

Tastaturbelegung FORMANT

F13 F1 F1	F14 F2 F2	F15 F3 F3	F16 F4 F4	F17 F5 F5	F18 F6 F6	F19 F7 F7	F20 F8 F8	F21 F9 F9	F22 F10 F10	F23 F11 F11	F24 F12 F12	...

F25 F13 DUP	F26 F14 SEND

FKOR F15 LOEF	KOR F16 LVD	ANZ F17 -

VALID F18 F27	- F19 -	- F20 -	PRINT	

mit Shift

ohne Shift

BACK- SPACE	DEL
...	FE+
...	
...	MENU

HELP	START	END
	- WORD ERS	
	LAST	
CUL	HOME	CUR
BACK- WARDS	NEXT	FOR- WARDS

FE-				
				FE+

Tastaturbelegung FORMANTGEN

Dateneingabe

[→], [←]	Positionieren der Schreibmarke im Feld
[⌫]	ein Zeichen löschen (rückwärts)
[R], [L]	nächstes bzw. vorhergehendes Feld
[↵]	nächstes Feld, löschen ab Schreibmarkenposition
[↖]	erstes Feld
[SHIFT] [WORD]	Feld löschen
[F16]	alle Felder löschen
[CHAR]	ein Leerzeichen einfügen
[SHIFT] [CHAR]	ein Zeichen ausfügen.

Beim 'Attribute setzen' zusätzlich:

[↑]	Darüberliegendes Feld.
[↓]	Darunterliegendes Feld.

Fehler rücksetzen

[q] oder [Q]	entsperren die Tastatur wieder.

Positioniermodus

[F1]	Neues Feld öffnen und in den Schreibmodus gehen.
[CTRL] [a]	Wie [F1] (auf).
[F3]	Feld zum Ändern öffnen und in den Schreibmodus gehen.
[CTRL] [u]	Wie [F3] (update).
[→]	Schreibmarke nach rechts.
[←]	Schreibmarke nach links.
[⇥]	Tabulator rechts.
[⇤]	Tabulator links (Spalten 1,9,17, usw.).
[↵]	Anfang der nächsten Zeile.
[CHAR]	Nächstes Feld nach rechts verschieben.
[SHIFT] [CHAR]	Nächstes Feld nach links verschieben.
[LINE]	Zeile einfügen
[SHIFT] [LINE]	Zeile ausfügen
[F16]	Aktuelles Format im Arbeitspeicher löschen.
[F5]	Feld teilen in zwei Felder.
[CTRL] [c]	Wie [F5] (cut).

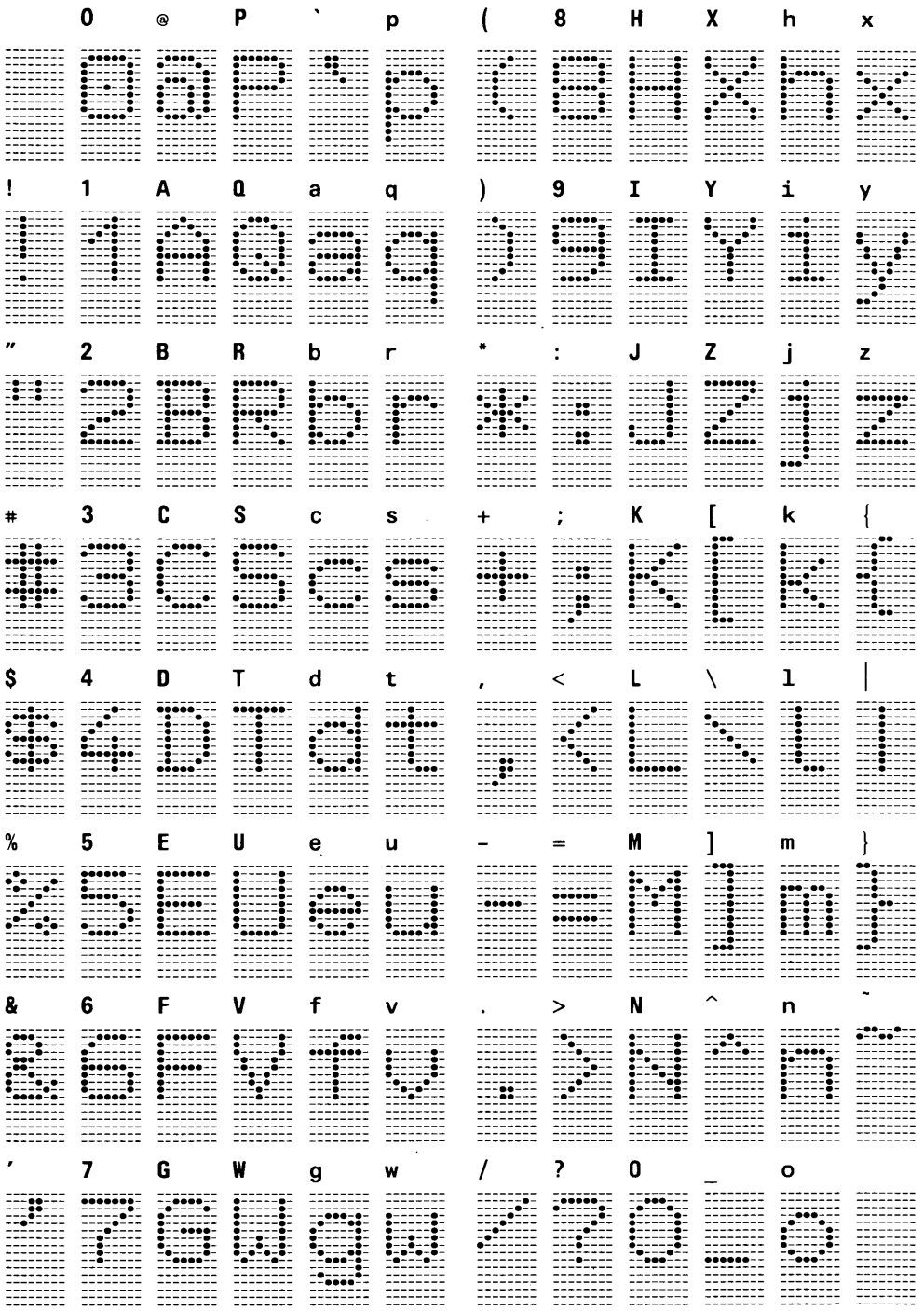
F6	Zwei nebeneinanderliegende Felder verbinden.
CTRL [	Wie F6 (link).
SHIFT WORD	Feld löschen.
F16	Aktuelles Format im Arbeitspeicher löschen.

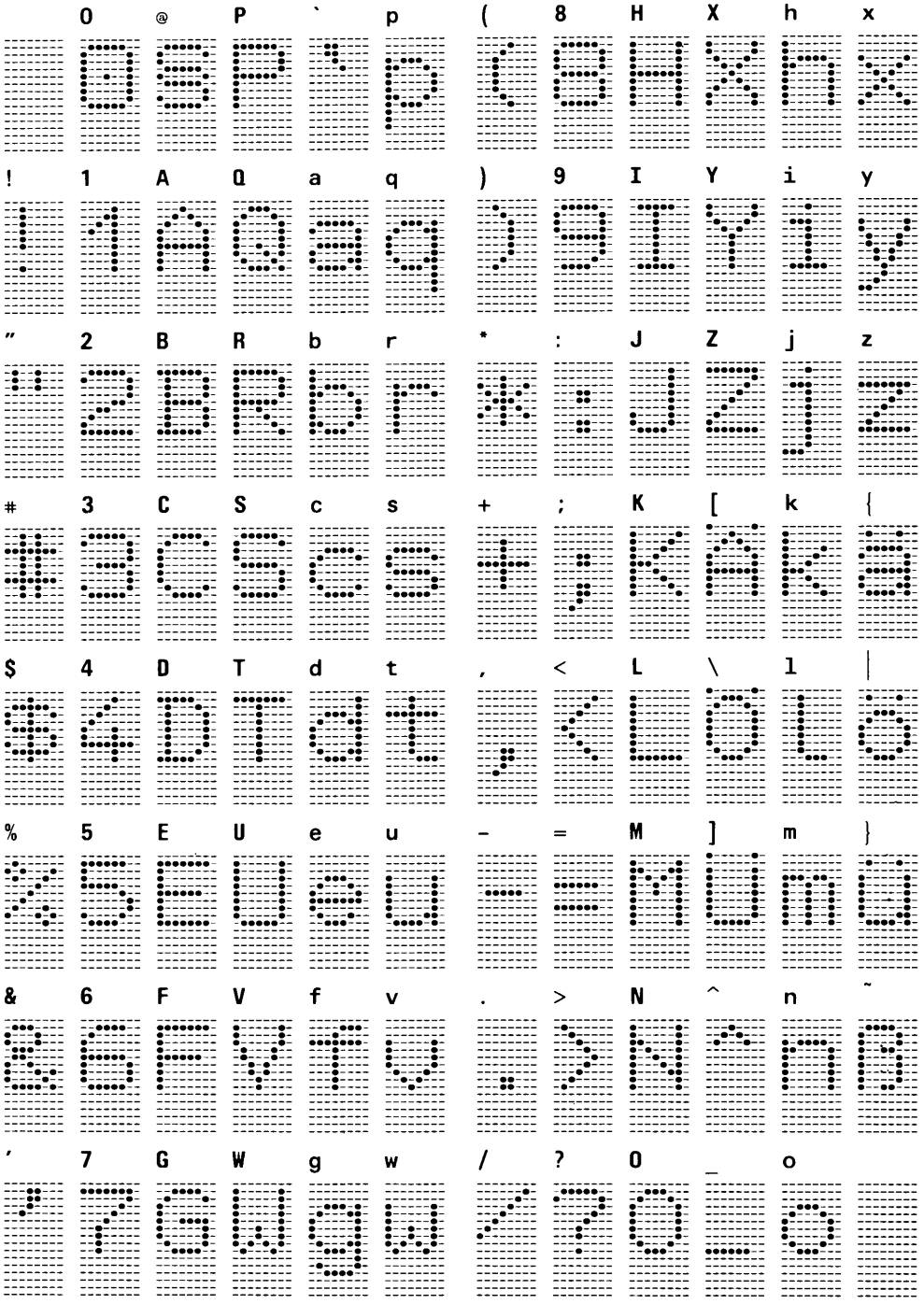
Schreibmodus

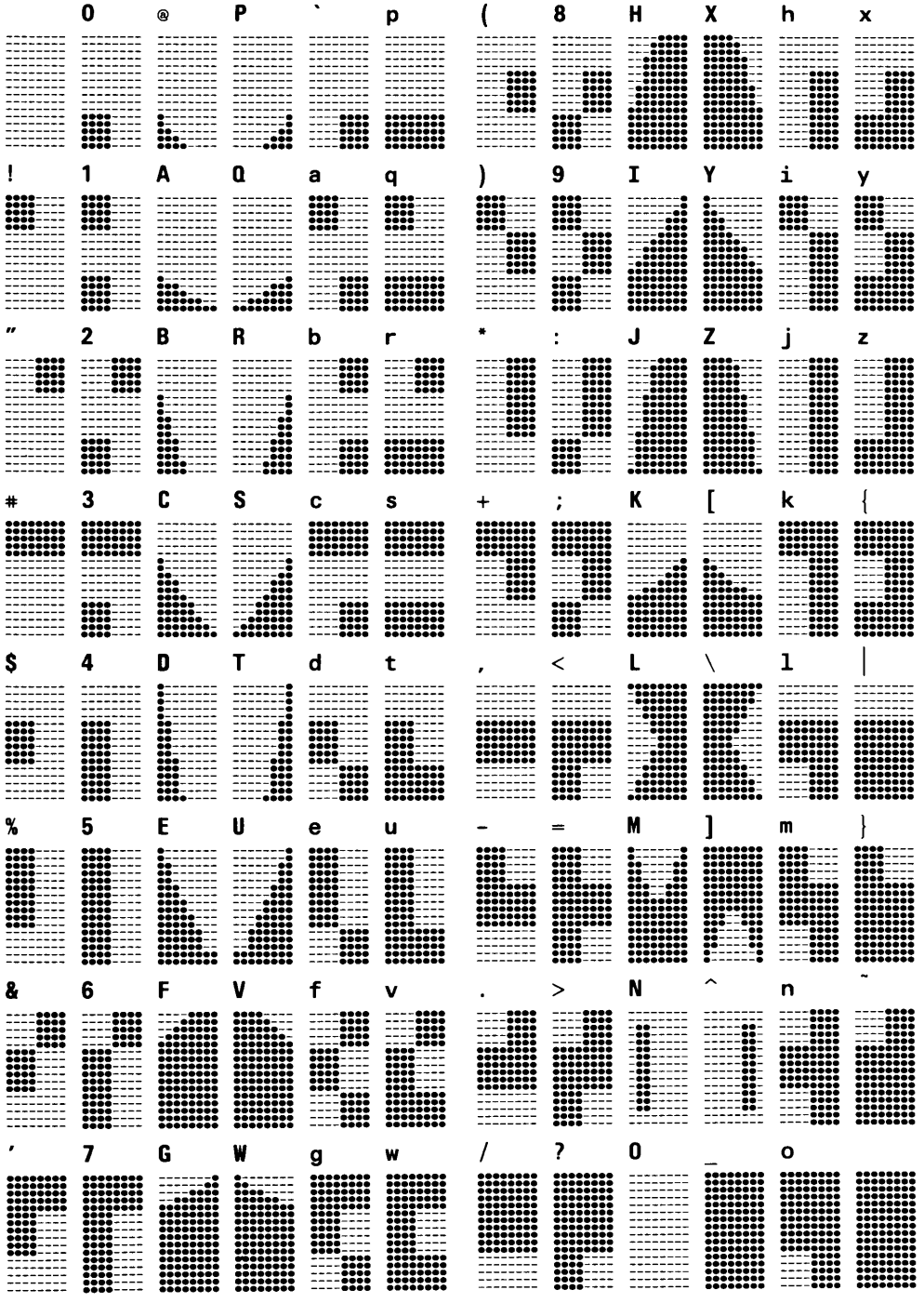
F2	Neues Feld schließen.
CTRL z	Wie F2 (zu).
F4	Feld nach dem Ändern schließen.
CTRL e	Wie F4 (exit).
→	Schreibmarke nach rechts.
←	Schreibmarke nach links.
⇥	Schreibmarke an das aktuelle Feldende.
⇤	Schreibmarke an den aktuellen Feldanfang.
⌫	Ein Zeichen löschen (rückwärts).
CHAR	Ein Leerzeichen einfügen.
SHIFT CHAR	Ein Zeichen ausfügen.
SHIFT WORD	Feld löschen.
F16	Aktuelles Format im Arbeitspeicher löschen.

Reihenfolge festlegen

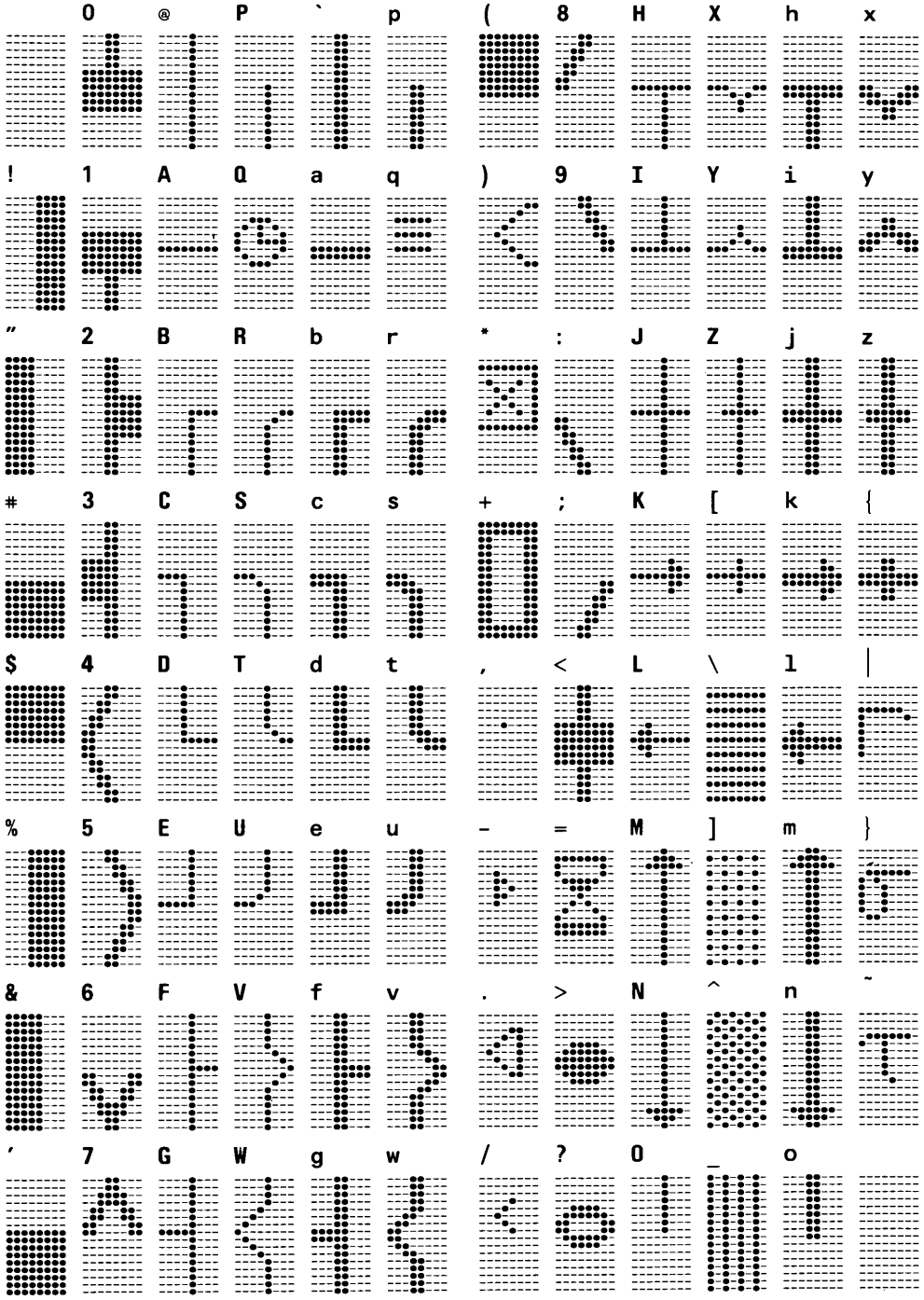
⇥	Nächstes Feld.
⇤	Vorhergehendes Feld.
↖	Erstes Feld.
MENU	Markieren beenden.

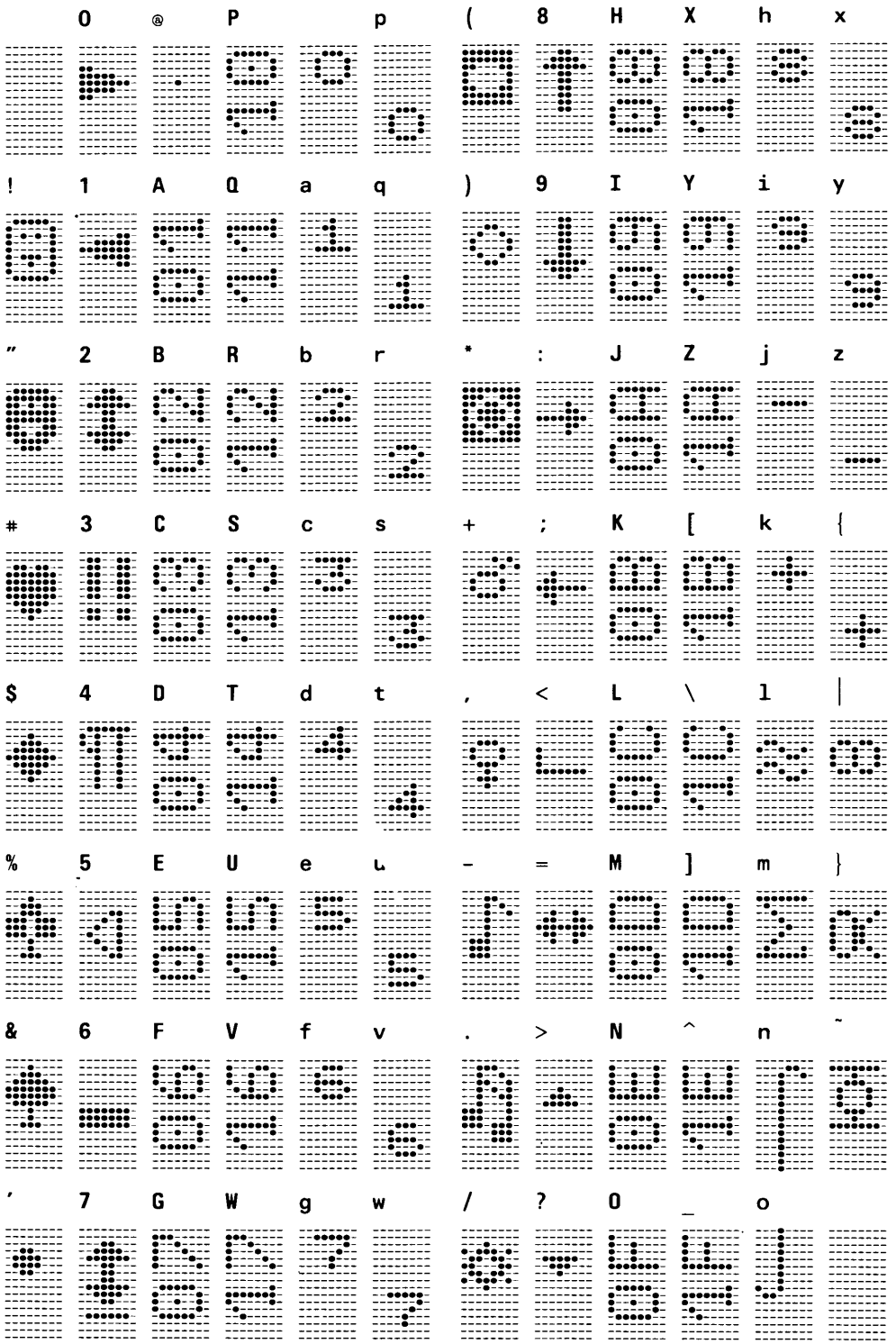

0 @ P ` p (8 H X h x
! 1 A Q a q) 9 I Y i y
" 2 B R b r * : J Z j z
3 C S c s + ; K [k {
$ 4 D T d t , < L \ l |
% 5 E U e u - = M] m }
& 6 F V f v . > N ^ n ~
' 7 G W g w / ? O _ o


0 @ P ` p (8 H X h x
! 1 A Q a q) 9 I Y i y
" 2 B R b r * : J Z j z
3 C S c s + ; K [k {
$ 4 D T d t , < L \ l |
% 5 E U e u - = M] m }
& 6 F V f v . > N ^ n ~
' 7 G W g w / ? O _ o


0 @ P ` p (8 H X h x
! 1 A Q a q) 9 I Y i y
" 2 B R b r * : J Z j z
3 C S c s + ; K [k {
$ 4 D T d t , < L \ l |
% 5 E U e u - = M] m }
& 6 F V f v . > N ^ n ~
' 7 G W g w / ? O _ o

	0	@	P	`	p	(	8	H	X	h	x
!	1	A	Q	a	q	)	9	I	Y	i	y
"	2	B	R	b	r	*	:	J	Z	j	z
#	3	C	S	c	s	+	;	K	[	k	{
$	4	D	T	d	t	,	<	L	\	l	|
%	5	E	U	e	u	-	=	M	]	m	}
&	6	F	V	f	v	.	>	N	^	n	~
'	7	G	W	g	w	/	?	O	_	o	


0 @ P ` p (8 H X h x
! 1 A Q a q) 9 I Y i y
" 2 B R b r * : J Z j z
3 C S c s + ; K [k {
$ 4 D T d t , < L \ l |
% 5 E U e u - = M] m }
& 6 F V f v . > N ^ n ~
' 7 G W g w / ? O _ o


0 @ P p (8 H X h x
! 1 A Q a q) 9 I Y i y
" 2 B R b r * : J Z j z
3 C S c s + ; K [k {
$ 4 D T d t , < L \ l |
% 5 E U e u - = M] m }
& 6 F V f v . > N ^ n ~
' 7 G W g w / ? O _ o

Aufrufnamen, alphabetisch

change error message	Fehlermeldung ausgeben	N-CEM	n_cem
change field	Feldinhalt ändern	N-CHF	n_chf
change field attribute	Feldeigenschaften ändern	N-CFA	n_cfa
change field type	Feldtyp und Reihenfolge ändern	N-CFT	n_cft
change record	Datensatz in die Maske übertragen	N-CHR	n_chr
change status message	Statusmeldung ausgeben	N-CSM	n_csm
clear field	Feldinhalt löschen	N-CLF	n_clf
clear record	Alle Felder der Maske löschen	N-CLR	n_clr
close session	Sitzung schließen	N-CLS	n_cls
create presentation image	Maske anlegen	N-CPI	n_cpi
define user exits	Benutzerroutinen für Triggerfelder	N-DUE	n_due
delete field	Feld ausfügen	N-DEF	n_def
destroy presentation image	Maske schließen	N-DPI	n_dpi
get cursor	Schreibmarkenposition abfragen	N-GEC	n_gec
get field	Datenfeld übernehmen	N-GEF	n_gef
get field attribute	Feldeigenschaften abfragen	N-GFA	n_gfa
get record	Datensatz aufbauen	N-GER	n_ger
insert field	Feld einfügen	N-INF	n_inf
modify environment parameters	Umgebungsparameter ändern	N-MEP	n_mep
modify input reaction table	Reaktion auf Systemsteuertasten	N-MIR	n_mir
modify operating mode	Eingabeparameter ändern	N-MOM	n_mom
modify session parameters	Sitzungsparameter ändern	N-MSP	n_msp
open session	Sitzung eröffnen	N-OPS	n_ops
print presentation image	Maske ausdrucken	N-PPI	n_ppi
read	Daten einlesen	N-RED	n_red
request environment parameters	Umgebungsparameter abfragen	N-REP	n_rep
request operating mode	Eingabeparameter abfragen	N-ROM	n_rom
request session parameters	Sitzungsparameter abfragen	N-RSP	n_rsp
save presentation image	Maske abspeichern	N-SPI	n_spi
set beeper	Signalton erzeugen	N-SBE	n_sbe
set cursor	Schreibmarke positionieren	N-SEC	n_sec
set dup record	Dupliziersatz setzen	N-SDR	n_sdr
write	Maske ausgeben	N-WRT	n_wrt
write read	Maske ausgeben und Daten einlesen	N-WIR	n_wir