robotron

Programmtechnische
Beschreibung

FOR77

Benutzungshinweise

C 3015-0200-1 M 3030

Arbeitsplatzcomputer A 7150 Betriebssystem DCP 1700

ANWENDER-DOKUMENTATION	FORTRAN77 Benutzungshinweise	MOS
11/87		DCP

Programmtechnische Beschreibung

Benutzungshinweise FORTRAN77 2.0

AC A7150, EC 1834

VEB Robotron-Projekt Dresden

C 3015-0200-1 M3030

Die vorliegende Anwenderdokumentation, FORTRAN77 DCP, Benutzungshinweise, entspricht dem Stand von 11/87 und ist gültig für FORTRAN77 DCP, ab Ausgabe 2.0.

Die Ausarbeitung erfolgte durch ein Kollektiv des VEB Robotron-Projekt Dresden

Herausgeber:

VEB Robotron-Projekt Dresden
Leningrader Straße 9
Dresden DDR
8010

Kurzreferat

Die vorliegende programmtechnische Beschreibung enthält die erforderlichen Informationen für die Nutzung des FORTRAN77-Compilers und der zugehörigen Laufzeitbibliotheken im DCP. Als Kurzbezeichnung für die Gesamtheit des FORTRAN77-Compilers und der dazugehörigen Laufzeitbibliotheken im DCP wird FOR77 verwendet. Es werden die notwendigen Kommandos zum Übersetzen, Verbinden und Ausführen von FORTRAN77-Programmen beschrieben und alle implementationsabhängigen Elemente von FOR77 aufgeführt. Weiterhin liefert die Dokumentation Informationen zum Einbinden von externen Assemblerroutinen und zur Fehlerdiagnose.
Die in dieser Dokumentation beschriebene Optimierung von FORTRAN77-Programmen ist vorbereitet, wird aber erst in einer späteren Version von FOR77 realisiert.
Der FORTRAN77-Compiler des DCP ist kompatibel zu dem entsprechenden Compiler des SCP1700.

Inhaltsverzeichnis

Tabellenverzeichnis

Bildverzeichnis

1. Notation und Abkürzungen

Zur übersichtlichen Darstellung von einzugebenden Kommandozeilen oder von Ausschriften der Komponenten des DCP auf dem Terminal werden in der vorliegenden Schrift folgende Notationen und Abkürzungen verwendet:

Großbuchstabe(n)	- für Kommandos
Kleinbuchstabe(n)	- für Ausschriften
'Kleinbuchstabe(n)'	- variable Teile in Ausschriften und Kommandozeilen
[ ]	- Kennzeichnung von wahlfreien Angaben in Kommandozeilen und Ausschriften
|	- Formulierung von Alternativen (z. B. a|b)
{ }	- Klammerung
::=	- Definition und Untersetzung von variablen Teilen in Ausschriften und Kommandozeilen
CR	- Wagenrücklauf (Eingabetaste, hexadezimal 0D)
LF	- Zeilenvorschub (hexadezimal 0A)
CTRL-?	- gleichzeitiges Betätigen der Tasten CTRL und der anstelle des Zeichens '?' aufgeführten Taste
FF	- Vorschub auf Anfang der nächsten Seite (hexadezimal 0C)

2. FOR77 im Betriebssystem DCP

FOR77 besteht aus dem Compiler und einem generierbaren Laufzeitsystem. Der FOR77-Compiler und die Bestandteile des generierbaren Laufzeitsystems stehen dem Anwender auf 5 1/4"-Disketten zur Verfügung.

Die Diskette des Laufzeitsystems enthält die Datei F77INFO.TXT. Diese Datei liefert Informationen zur jeweiligen Ausgabe des Compilers (erfolgte Fehlerkorrekturen, Erweiterungen von FOR77 und der zugehörigen Dokumentation).

Für die Übersetzung eines FORTRAN77-Programmes mit dem FOR77-Compiler wird ein Hauptspeicherbereich von mindestens 200K Bytes benötigt. Größere Hauptspeicherbereiche verringern die Übersetzungszeit.

Neben den Bestandteilen von FOR77 werden zur Herstellung eines ausführbaren FORTRAN77-Programmes noch andere Komponenten des Betriebssystems DCP benötigt.
Zur Erzeugung und Wartung von FORTRAN77-Quelltexten wird ein Editor des DCP verwendet.
Der Bibliothekar LIB des DCP kann benutzt werden, um vom Compiler erzeugte Objektmoduln sowie weitere Objektmoduln zu einer Bibliothek zusammenzufassen.
Der Linker LINK des DCP wird benötigt, um die vom Compiler erzeugten Objektmoduln mit Moduln des Laufzeitsystems zu einem ausführbaren Programm zu verbinden.

Im Ergebnis der Übersetzung liefert der FOR77-Compiler einen Rückkehrcode, dessen Wert über ERRORLEVEL in einem Stapelverarbeitungsjob abgefragt werden kann (siehe Abschnitt 9.1.). Dasselbe gilt für die Ausführung eines FORTRAN77-Programmes, wobei in diesem Fall der Wert des Rückkehrcodes vom Anwender beeinflußbar ist (siehe Abschnitt 6.).

3. Übersetzen von FORTRAN77-Programmen

Beim Übersetzen eines FORTRAN77-Programmes entsteht ein Objektmodul mit dem Dateityp OBJ. Als Dateiname für den Objektmodul wird entweder der Name der Quelldatei gewählt oder, falls die Option "M" (siehe Abschnitt 3.4.1.) angegeben ist, der Name der FORTRAN77-Programmeinheit.

3.1. Struktur des FOR77-Compilers

Der FOR77-Compiler übersetzt einen FORTRAN77-Quelltext in mehreren Schritten in einen Objektmodul. Diese Übersetzungsschritte werden im weiteren Pässe genannt. Ein Paß kann aus mehreren Phasen bestehen, die nacheinander geladen werden. In Tabelle 1 sind die Pässe und ihre Funktion dargestellt.

Tabelle 1: Pässe des FOR77-Compilers

Pass	Phase	Funktion
FOR77	FOR77	Wurzel, Serviceroutinen, Paßwechsel
IPASS	FIPASS	Compilerinitialisierung
WPASS	W1PASS	Quelltext-Eingabe, Auflistung, Vorbereitung der lexikalischen Analyse
	W2PASS	Lexikalische Analyse
SPASS	SYPASS	Syntaxanalyse
GPASS	GXPASS	Globale Prüfung, Erzeugung von Symboltabellen und Symbolnachweistabellen, sowie Vorbereitung der Optimierung
OPASS	O1PASS	Optimierung 1
	O2PASS	Optimierung 2
RPASS	R1PASS	Registerzuordnung
CPASS	CCPASS	Objektcode-Erzeugung
EPASS	FEPASS	Erzeugung einer Fehlernachrichten-Liste

Der Compiler ist als Überlagerungsstruktur aufgebaut, in der die einzelnen Phasen dynamisch nachgeladen werden. Einige Phasen sind ebenfalls Überlagerungsstrukturen. Der Compiler muß sich nicht im aktuellen Verzeichnis befinden. Er kann in einem Verzeichnis enthalten sein, das über das Kommando PATH angeschlossen wurde oder über den Pfad geladen werden, der beim Compileraufruf von FOR77 angegeben wird.

Alle Phasen des Compilers müssen sich in dem Verzeichnis befinden, in dem der Modul FOR77.EXE enthalten ist. Soll der Compiler von mehreren Disketten geladen werden, ist ein Diskettenwechsel während der Übersetzung notwendig.

Die Compilerphasen, die sich auf der Folgediskette befinden, müssen in einem Verzeichnis dieser Diskette enthalten sein, das über den gleichen Pfadnamen erreichbar ist, wie das der Wurzel FOR77.EXE. Der Gerätename kann gewechselt werden.

Eine Phase muß vollständig in einem Verzeichnis enthalten sein.

Befindet sich eine Phase nicht in dem Verzeichnis, aus der die vorhergehende Phase geladen wurde (siehe Abschnitt 3.2.), so erscheint auf dem Terminal die Meldung:

```
unable to open file: 'phase'
change disk and/or type device name
```

Dabei ist 'phase' eine Phase aus Tabelle 1.

Der Anwender hat an dieser Stelle die Möglichkeit, eine Diskette zu wechseln oder ein Laufwerk (A - P) anzugeben, in dem sich die Folgediskette befindet. Wird das Laufwerk nicht gewechselt, sondern nur die Diskette, genügt CR. Wurde eine Laufwerksangabe eingegeben, so wird aus Gründen der Sicherheit eine Bestätigung der Angabe durch die Aufforderung:

```
Confirm device 'laufwerksangabe'
```

verlangt. Die Aufforderung muß mit neuer Laufwerksangabe oder mit CR beantwortet werden. Die Eingabe von CTRL-C beendet die Übersetzung.

Auf Disketten, die während der Übersetzung gewechselt werden, dürfen sich keine Dateien befinden, die der Compiler eröffnet hat (Hilfsdatei, Ein- und Ausgabedateien).

Die Compilerdateien müssen sich im aktuellen Verzeichnis für das angegebene Gerät befinden, bzw. werden dort angelegt.

3.2. Aufruf des FOR77-Compilers

Der FOR77-Compiler wird mittels folgendem Kommando aufgerufen:

```
['pfad'] FOR77 'qdatei' [{ $|#} 'options'] CR
```

Dabei ist

'pfad' Pfadname zum Verzeichnis, das den Compiler enthält,

'qdatei' die Dateispezifikation des zu übersetzenden FORTRAN77-Quelltextes

'options' eine Liste von Optionen der Kommandozeile (siehe Tabelle 2), mit deren Hilfe der Übersetzungsablauf und Compilerlisten/-dateien gesteuert werden.

Eine Dateispezifikation hat das Format:

```
[geräteangabe:] dateiname [.dateityp]
```

Fehlt der Dateityp, so ergänzt der FOR77-Compiler den Dateityp F77.

Bei fehlender Geräteangabe wird das Gerät aus der S-Option (siehe Abschnitt 3.4.1.) ergänzt oder das Standardgerät verwendet. Eine Pfadangabe ist nicht möglich.

Kommt es durch fehlerhafte oder fehlende Angaben in der Kommandozeile zu Initialierungsfehlern des FOR77-Compilers, so wird die Übersetzung mit einer selbsterklärenden Meldung abgebrochen.

3.3. Dateien des FOR77-Compilers

3.3.1. Eingabedatei des FOR77-Compilers

Die Eingabedatei muß sich im aktuellen Verzeichnis auf einer Diskette oder Festplatte befinden. Die Dateispezifikation ist in der Kommandozeile anzugeben.
Um dem Anwender die Eingabe im FORTRAN77-Standardeingabeformat (siehe [1]) zu erleichtern, wurde die Möglichkeit der Formatierung mittels Tabulator eingeführt. Ein Tabulator in den Positionen 1 bis 5 bewirkt den Übergang zu Spalte 6, wenn eine Ziffer 1 bis 9 folgt (Fortsetzungszeichen), ansonsten zu Spalte 7 (Programmtext). Die Zeichen vor dem Tabulator werden dem Markenfeld zugeordnet. Tabulatoren im Fortsetzungsfeld und im Anweisungsfeld haben die gleiche Bedeutung wie das Leerzeichen.
Eine Eingabezeile kann weniger als 72 Zeichen enthalten. Das Anweisungsfeld wird in diesem Fall mit Leerzeichen aufgefüllt.
Hat ein Eingabesatz mehr als 72 Zeichen, wird eine Warnung ausgegeben. Die Zeichen ab Position 73 werden als Kommentar behandelt.
Tritt im Anweisungsfeld das Zeichen "!" auf, so wird der Rest der Zeile ab diesem Zeichen als Kommentar gewertet. Damit wurde in Erweiterung des Standards die übliche Art der Kommentierung ermöglicht.
Die Eingabedatei kann mehrere FORTRAN77-Programmeinheiten enthalten.

Der FORTRAN77-Quelltext kann in Kleinbuchstaben eingegeben werden. Der Compiler wandelt alle Kleinbuchstaben in Namen und Schlüsselworten, außerhalb von Zeichenketten und Hollerithkonstanten, in Großbuchstaben um.
Zum Beispiel ist die Bezeichnung "NaMe" identisch mit "NAME".

Vor und zwischen FORTRAN77-Anweisungen können sich Steuerzeilen befinden. Diese werden durch das Zeichen "$" in der ersten Position gekennzeichnet. Die Wirkung der Steuerzeile ist im Abschnitt 3.4.2. beschrieben.

3.3.2. Ausgabedateien des FOR77-Compilers

Der FOR77-Compiler erzeugt in Abhängigkeit von Optionen (siehe Abschnitt 3.4.1.) im aktuellen Verzeichnis verschiedene Dateien. Diese Dateien erhalten eine Dateispezifikation gemäß Betriebssystemkonventionen (siehe Abschnitt 3.2.), ohne Pfadangaben.
Die Geräteangabe wird entweder einer Option der Kommandozeile entnommen oder es wird, falls die Geräteangabe fehlt, das Standardgerät der Übersetzung (siehe Abschnitt 3.4.1.) angenommen.
Der Dateiname ist i.a. der Dateiname der Eingabedatei, die in der Kommandozeile angegeben wurde. Der Dateityp der erzeugten Dateien wird wie folgt gebildet:

PRN - für Quelltextauflistungen und Tabellen

OBJ - für den Objektcode, der als Resultat der Übersetzung entsteht

ERR - Protokoll der Übersetzungsfehler

Ist die Option M in der Kommandozeile angegeben (siehe Abschnitt 3.4.1.), so wird für jede FORTRAN77-Programmeinheit ein separater Objektmodul erzeugt. Als Dateiname eines solchen Objektmoduls wird der Name der Programmeinheit verwendet. Hauptprogramme ohne PROGRAM-Anweisung erhalten den Namen der Eingabedatei aus der Kommandozeile. Unbenannte BLOCK DATA-Unterprogramme erhalten den Dateinamen der Eingabedatei, erweitert mit "#" auf acht Zeichen.

Sind in einer Eingabedatei mehrere unbenannte BLOCK DATA-Unterprogramme enthalten, führt das zum Überschreiben der Objektmoduln.

Der FOR77-Compiler benötigt eine Arbeitsdatei. Sie wird im aktuellen Verzeichnis unter dem Namen SPILLFIL.TMP angelegt. Das Gerät wird entsprechend der Angabe in der T-Option (siehe Abschnitt 3.4.1.) gewählt oder es wird das Standardgerät der Übersetzung verwendet. Die Arbeitsdatei wird am Ende der Übersetzung wieder gelöscht.

3.4. Optionen des FOR77-Compilers

Die Optionen des FOR77-Compilers dienen der Steuerung des Übersetzungsprozesses und der Definition der Ein- und Ausgabedateien des Compilers.
Einige Optionen sind nur in der Kommandozeile angebbar, andere nur im Quellprogramm (siehe Abschnitte 3.4.1. und 3.4.2.). Im folgenden sind die Optionen (außer denen zur Angabe von Geräten) unabhängig von ihrer Position beschrieben. Die hier verwendeten Bezeichnungen der Optionen sind in der Kommandozeile bzw. im Quellprogramm nicht angebbar und haben nur erläuternde Bedeutung. Die Optionen sind nur in der in den Tabellen 2 und 3 dargelegten Form angebbar.

OBJECT/NOOBJECT
Als Ergebnis der Übersetzung soll ein/kein Objektmodul erzeugt werden.

GOSTATEMENT/NOGOSTATEMENT
In der OBJ-Datei werden bei Angabe von GOSTATEMENT Anweisungen erzeugt, die die Anweisungsnummer in einen Arbeitsspeicherplatz zur Laufzeit eintragen. Bei Fehlern zur Laufzeit wird die Anweisungsnummer mit ausgegeben. Pro Anweisung werden sieben Byte zusätzlich erzeugt.

DEBUG/NODEBUG
Ein in Position 1 einer FORTRAN77-Quelltextzeile stehendes "D" wird durch Leerzeichen/"C" ersetzt,d.h., ist DEBUG angegeben, so wird diese Zeile als FORTRAN77-Anweisung übersetzt, ansonsten als Kommentar übergangen. DEBUG impliziert GOSTATEMENT.

SYMBOLS/NOSYMBOLS
In der PRN-Datei wird eine Symboltabelle erzeugt, wenn SYMBOLS angegeben wurde. Die Tabelle enthält die Namen und Attribute der im Programm vereinbarten Objekte.

XREF/NOXREF
In der PRN-Datei wird eine Symbolnachweistabelle erzeugt, wenn XREF angegeben wurde. Ist XREF und SYMBOLS wirksam, so entsteht eine gemeinsame Tabelle.

PAGELENGTH(n)
Bei der Erzeugung der PRN-Datei wird nach n Zeilen ein Seitenvorschub erzeugt. Dabei gilt 16 < n <= 32760. Standard ist bei Ausgaben auf Terminal 32760 (kein Seitenvorschub), ansonsten 65.

PAGEWIDTH(n)
Die Zeilenlänge einer Druckzeile der PRN-Datei beträgt maximal n Zeichen. Dabei gilt 60 <= n <= 132. Standard bei Ausgabe auf das Terminal ist 79, ansonsten 120.

STORAGE(INTEGER*x,REAL*y,LOGICAL*z)
Mit dieser Option wird die Länge von INTEGER-, REAL- und/oder LOGICAL-Variablen in Bytes festgelegt, wenn sie ohne Längenangabe vereinbart werden. Für x, y und z sind folgende Werte möglich:

```
x = 2 oder 4
y = 4 oder 8
z = 1, 2 oder 4
```

Wird die STORAGE-Option nicht angegeben, gelten folgende Standardannahmen:
x=4, y=4, z=4.

COMPILE/NOCOMPILE(n)
Ist die Option COMPILE wirksam, wird die Übersetzung bis zur Codeerzeugung fortgesetzt, auch wenn Fehler der Schwere "S" (siehe Abschnitt 9.1.) im FORTRAN77-Quelltext auftraten.
Ist die Option NOCOMPILE wirksam, wird die Übersetzung nach der globalen Prüfung (GPASS) unbedingt beendet. Durch die Angabe von NOCOMPILE(n) ist die Fortsetzung der Übersetzung abhängig von der Fehlerschwere n. Wenn Fehler der Schwere n oder schwerere Fehler auftraten, wird die Übersetzung nach der globalen Prüfung beendet. Dabei kann n das Zeichen E oder S sein.

TITLE('text')
Die Zeichenkette 'text' wird in die erste Zeile einer Seitenüberschrift in der PRN-Datei eingefügt.

SUBTITLE('text')
Die Zeichenkette 'text' wird in eine zusätzlich ausgegebene zweite Überschriftszeile der PRN-Datei eingefügt. Tritt eine Steuerzeile mit SUBTITLE-Option zwischen FORTRAN77-Anweisungen auf, bewirkt sie gleichzeitig den Übergang auf eine neue Seite der PRN-Datei.

EJECT
Übergang auf eine neue Seite der PRN-Datei.

LIST/NOLIST
Der FORTRAN77-Quelltext soll/soll nicht nach der PRN-Datei ausgegeben werden.

INCLUDE(dateiname)
Die INCLUDE-Option liefert die Möglichkeit, Quelltext aus einer weiteren Datei in das Programm einzufügen. Im eingefügten Quelltext können wiederum INCLUDE-Steuerzeilen enthalten sein. Die Schachtelungstiefe ist auf drei Stufen beschränkt.
Ist 'dateiname' nicht vollständig angegeben, d.h.
"gerät: dateiname . erweiterung",
so werden folgende Standardannahmen gültig:
"gerät" : Geräteangabe aus der S-Option der Kommandozeile, bzw. Gerät der primären Eingabe
"erweiterung" : F77
Leerzeichen sind in der Option nicht erlaubt.

OFFSET
Auf die PRN-Datei wird eine Statement-Offset-Tabelle ausgegeben. Diese Tabelle enthält für alle Anweisungen die Anweisungsnummer und das zugehörige Offset im CODE-Segment (hexadezimal).

VALUE_CONTROL/NOVALUE_CONTROL
Bei Angabe von VALUE_CONTROL wird Code für die Überwachung von Werten der Indexgrößen und von Längenwerten erzeugt.

QUIET
Fehlernachrichten zur Übersetzungszeit werden nicht auf dem Terminal ausgegeben. Sie werden nur in der ERR-Datei gesammelt. Ist QUIET nicht wirksam, wird jede Fehlermeldung (außer Warnungen) bei Erkennung des Fehlers auf dem Terminal in einer Kurzform ausgegeben (siehe Abschnitt 9.1.).

OPT[(n)]/NOOPT
Bei Angabe von NOOPT wird nur eine nicht aufwendige, lokale Optimierung durchgeführt.
Ist OPT wirksam, wird eine Optimierung des zu erzeugenden Codes vorgenommen. Es sind zwei Stufen der Optimierung möglich. N=1 bewirkt eine Optimierung auf lokalem Niveau, n=2 eine Optimierung auf globalem Niveau. Wird (n) nicht angegeben, gilt OPT(1).

3.4.1. Optionen der Kommandozeile

Die Optionen der Kommandozeile beginnen mit dem Zeichen "$" oder "#" und müssen vom Dateinamen durch mindestens ein Leerzeichen getrennt sein. Die Optionen untereinander können durch Leerzeichen getrennt sein. Optionen der Kommandozeile sind für alle zu übersetzenden Programmeinheiten wirksam. Die Tabelle 2 enthält eine Übersicht aller Optionen der Kommandozeile.

Tabelle 2: Optionen der Kommandozeile

Angabe der Option in der Kommandozeile		Standard, falls nicht angegeben	Bedeutung
1.	2. Zeichen 1)		
S	A..P, @	Geräteangabe der Quelle oder @	Standardgerät für die Übersetzung, auf der die Compilerdateien liegen.
O	Leerzeichen Z A..P, @	Standard-gerät der Übersetzung	Gerät für OBJ-Datei OZ impliziert NOOBJECT
P	Leerzeichen X,Y,Z A..P, @	PZ	Gerät für PRN-Datei PZ impliziert NOLIST; sonst gilt LIST
T	A..P, @	wie O	Gerät für Arbeitsdatei SPILLFIL.TMP
E	X,Y, A..P, @	wie O	Gerät der ERR-Datei
B	-	-	OFFSET 2)
D	-	NODEBUG	DEBUG
G	-	NOGOSTATEMENT	GOSTATEMENT
X	-	NOXREF und NOSYMBOLS	XREF und SYMBOLS 2)
M	-	-	Modulseparierung
Q	-	-	QUIET
C	-	NOCOMPILE(S)	COMPILE
V	-	NOVALUE_CRTL	VALUE_CONTROL
@	1 oder Leerz. 2	NOOPT	OPT(1) OPT(2)
N	C E	NOCOMPILE(S)	NOCOMPILE NOCOMPILE(E)
	H		3)
	L	LIST 4)	NOLIST

Erläuterungen:

1) Geräteangabe:

A..P	Gerät A: bis P:
@	Standardgerät (DCP)
Leerzeichen	Standardgerät der Übersetzung
Z	Unterdrückung der Ausgabe
X	Terminal
Y	Drucker

2) Existiert keine PRN-Datei (PZ), so erfolgt die Ausgabe auf das Terminal.

3) Formularvorschubzeichen werden nicht auf die PRN-Datei ausgegeben. Die Seitenlänge wird auf den Maximalwert gestellt.

4) LIST wird Standard, wenn eine PRN-Datei erzeugt werden soll.

Die Beschreibung der hier nicht erläuterten Optionen ist dem Abschnitt 3.4. zu entnehmen.

3.4.2. Optionen der Steuerzeile

Steuerzeilen eines FORTRAN77-Programmes sind durch das Zeichen "$" in der ersten Position einer Quelltextzeile gekennzeichnet. Beginnt die Steuerzeile mit der Zeichenfolge "$$", wird die Steuerzeile nicht mit aufgelistet.
Diesem ersten Zeichen folgt eine durch Kommas getrennte Liste von Optionen, wie sie laut Tabelle 3 in der Steuerzeile angegeben werden dürfen.

Tabelle 3: Optionen der Steuerzeile

Angabe der Option in der Steuerzeile	Option	Art der Option 1)
[NO]GS	[NO]GOSTATEMENT	P
[NO]VC	[NO]VALUE CONTROL	P
STG(I*x,R*y,L*z) 2)	STORAGE	G
[NO]SB	[NO]SYMBOL	P
[NO]XR	[NO]XREF	P
[NO]L	[NO]LIST	A
NOOPT	NOOPT	P
OPT(n)	OPT(n)	P
[NO]DB	[NO]DEBUG	P
NOCOM(n)	NOCOMPILE(n)	P
COM	COMPILE	P
TT('text')	TITLE	P
STT('text')	SUBTITLE	A
EJ	EJECT	A 3)
PW(n)	PAGEWIDTH	G
PL(n)	PAGELENGTH	G
INC('dateiname')	INCLUDE	A

Die Optionen sind im Abschnitt 3.4. beschrieben.

Erläuterungen zu Tabelle 3:

1) Die Steuerzeile mit der Option darf im Quelltext an folgenden Positionen stehen:
   - P Vor der ersten Quelltextanweisung einer Programmeinheit; nur für diese Programmeinheit wirksam.
   - G Vor der ersten Quelltextanweisung einer Programmeinheit; ist für alle folgenden Programmeinheiten wirksam.
   - A Vor und zwischen FORTRAN77-Quellanweisungen angebbar.

2) Es müssen nicht alle Argumente angegeben werden. Die Reihenfolge ist beliebig.
   Die Werte für x,y und z sind im Abschnitt 3.4. beschrieben.
   Beispiel: STG(R*8)

3) Die Steuerzeile mit dieser Option darf nicht vor der ersten FORTRAN77-Quellanweisung auftreten.

3.5. Auflistungen des FOR77-Compilers

3.5.1. Auflistungen auf dem Terminal

Zu Beginn der Übersetzung gibt der Compiler folgende Informationen auf dem Terminal aus:

- Kopfzeile
  FORTRAN77 DCP V.M n.m
- Teil der Kommandozeile ab Dateispezifikation
- Dateispezifikationen aller vom Compiler benutzten Dateien in folgender Form:

  File assignments:

  Source-file: 'dateispezifikation'
  Print-file: 'dateispezifikation'
  Spill-file: 'dateispezifikation'
  OBJ-file: 'dateispezifikation'
  ERR-file: 'dateispezifikation'

Wird während der Übersetzung eine weitere Phase des Compilers geladen, so erscheint auf dem Terminal die Ausschrift:

Phase: 'name der phase'
(z.B. Phase: GXPASS)

Wird der Compiler von mehreren Disketten geladen, wird der Anwender zum Diskettenwechsel aufgefordert. Wird die zu ladende Phase auf der Diskette im Laufwerk g, von der die vorherige Phase geladen wurde, nicht gefunden, so erscheint auf dem Terminal folgende Mitteilung und Aufforderung:

Unable to open g:'name der phase'
Change disk and/or type device name

(siehe Abschnitt 3.1.)
Bei Eingabe einer neuen Laufwerksangabe wird aus Sicherheitsgründen eine Bestätigung gefordert:

Confirm device 'laufwerksangabe':

Mit dem Drücken der ENTER-Taste wird bestätigt, daß das angegebene Laufwerk angesprochen werden soll, bzw. eine neue Laufwerksangabe ist einzugeben.

Nach der Verarbeitung führender Steuerzeilen im Quelltext wird eine Liste der aktuellen Optionen ausgegeben. Als Bezeichnung wurde die im Abschnitt 3.4. angegebene Schreibweise benutzt.

Als Abschluß der Übersetzung einer Programmeinheit wird auf dem Terminal eine Tabelle ausgegeben, die die Namen der vom Compiler erzeugten Segmente, deren Längen und Speicherklasse enthält.

Weitere Auflistungen auf dem Terminal erscheinen, wenn die PRN-Datei oder die ERR-Datei auf das Terminal gelegt wurden.

Wurde die Option QUIET nicht angegeben, so wird beim Erkennen eines Fehlers eine verkürzte Fehlermeldung auf dem Terminal ausgegeben. Dabei besteht die Möglichkeit, die Übersetzung abzubrechen (siehe Abschnitt 9.1.).

Durch die Eingabe von CTRL-C kann an beliebiger Stelle die Übersetzung abgebrochen werden.

3.5.2. Auflistungen in der PRN-Datei

Wird in der Kommandozeile die P-Option mit einem von Z verschiedenen Gerätenamen angegeben, so wird die PRN-Datei auf dem entsprechenden Gerät erzeugt.
Die erste Zeile der PRN-Datei enthält Versionsangaben.

FORTRAN77 DCP V.M n.m Module:dateispez. PAGE 1

Die folgenden Zeilen enthalten eine Auflistung der Steuerzeilen bis zur ersten FORTRAN77-Anweisung und die Auflistung der wirksamen Optionen.

Ist LIST wirksam, so wird anschließend der FORTRAN77-Quelltext aufgelistet.

Die Quelltextauflistung hat folgenden Aufbau:

1. Kopfzeile

   FOR77 n.m MODULE: dateispezifikation PAGE: p
   wenn keine TITLE-Option vorhanden ist.

   FOR77 n.m 'text' PAGE: p
   wenn TITLE('text') angegeben ist.

2. Kopfzeile
   Sie ist nur vorhanden, wenn eine SUBTITLE-Option angegeben wurde und enthält die Zeichenkette aus der Option.

3. Kopfzeile (Listenüberschrift)

   LINE STMT S O U R C E L I S T I N G

In den folgenden Zeilen wird jede Quelltextzeile mit Zeilen- und Anweisungsnummer ausgegeben. Überschreitet die Länge der Quelltextzeile den Wert der PAGEWIDTH-Option, so wird eine weitere Zeile mit dem Rest der Quelltextzeile ausgegeben.

Die Optionen EJECT und SUBTITLE bewirken den Übergang auf eine neue Seite.

Die Kopfzeilen 1 und 2 werden nicht ausgegeben, wenn sich die PRN-Datei auf dem Terminal befindet.

Sind die Optionen XREF und/oder SYMBOLS wirksam, so wird eine Symbol- und/oder Symbolnachweis-Tabelle ausgegeben.
Die Tabelle hat folgenden Aufbau:

Spalten 1 bis 10 Name des im Programm vereinbarten Objektes

Spalten 11 bis n (n=Angabe in PAGEWIDTH)
Falls die Option
- SYMBOLS angegeben wurde, werden Attribute des Objektes ausgegeben.
- XREF angegeben wurde, werden Referenzen des Objektes (mehrere Zeilen sind möglich) ausgegeben.

Mit "*" gekennzeichnete Referenzen bedeuten:

- für Variablen: Variable kann in dieser Anweisung verändert werden.
- Marken: Definitionsstelle, DO-Ende-Marke

Die Option OFFSET bewirkt die Ausgabe der Statement-Offset-Tabelle.
Diese Tabelle enthält zu jeder Anweisungsnummer das Offset (hexadezimal) der entsprechenden FORTRAN77-Anweisung im CODE-Segment. Für ein Programm, bestehend aus drei Anweisungen, kann sie z.B. folgende Form haben:

STMT	OFFSET	STMT	OFFSET	STMT	OFFSET
1	0	2	1A	3	22

3.6. Aufbau des erzeugten Objektmoduls

Der FOR77-Compiler erzeugt für eine FORTRAN77-Programmeinheit in der Datei mit dem Dateityp OBJ einen Objektmodul mit folgendem Inhalt:

Modulkopf: Enthält den Namen der FORTRAN77-Programmeinheit oder FOR@@@ für ein unbenanntes Hauptprogramm.

Externe Vereinbarungen: Externe Vereinbarungen für Namen von Subroutinen, Funktionen, COMMON-Bereichen und für Eintrittspunktnamen von Bibliotheksroutinen (siehe Anlage 4).

PUBLIC-Vereinbarungen: Vereinbarung von Namen der Programmeinheit und der Eintrittspunktnamen. Für eine PROGRAM-Programmeinheit wird zusätzlich der PUBLIC-Name @_MAIN erzeugt.

Segmente:

Name	Attribute	Inhalt
D@nnn *)	PARA PUBLIC 'CODE'	Konstante, Programmvariable, Feld- und Formatbeschreiber
C@nnn *)	BYTE PUBLIC 'CODE'	Code für FORTRAN77-Anweisungen und compilerinterne Unterprogramme
COMMON-Name oder @COMMON	PARA COMMON 'EXTRA'	COMMON-Bereiche des Programmes
STACK	WORD STACK 'STACK'	

*) nnn ist der Name der Programmeinheit;
@COMMON ist der Name des unbenannten COMMON-Bereiches

Das CODE-Segment eines FORTRAN77-Hauptprogramms beginnt immer mit dem Aufruf der Laufzeit-Initialisierungsroutine FQ_PGM2. Diese ruft weitere Initialisierungsroutinen auf (siehe Tabelle 4).

Tabelle 4: Laufzeit-Initialisierungsroutinen

Name	Funktionen
F77_INI	Initialisierung des E/A-Steuersystems Initialisierung des Fehlerbehandlungssystems
INITFP	Initialisierungen für Gleitkomma-Arithmetik

Das STACK-Segment einer FORTRAN77-Programmeinheit hat genau die Größe, die zur Ausführung dieser Programmeinheit notwendig ist. Während des Verbindens eines FORTRAN77-Programms (siehe Abschnitt 5.) ergibt sich die Länge des STACK-Segments aus der Summe der Längen aller STACK-Segmente der zu verbindenden FORTRAN77-Programmeinheiten. Diese STACK-Länge ist aber zu groß, wenn sich aus der FORTRAN77-Programmstruktur ergibt, daß sich die bestimmten Programmeinheiten zugeordneten STACK-Segmente überlagern.

Bild 1 soll ein Beispiel einer FORTRAN77-Programmstruktur liefern. Bei der in diesem Bild gezeigten Programmstruktur ist nur eine Gesamtlänge des Stacksegments von 5436 Bytes notwendig, was der Summe aus den Längen der Stacksegmente des Hauptprogramms, des Unterprogramms A und des Unterprogramms C entspricht. Die Verkürzung des Stacks kann beim Verbinden durch Angabe der gewünschten Größe im STACK-Parameter erreicht werden (siehe Abschnitt 5.2.).
Dabei ist zu beachten, daß die FORTRAN77-Laufzeitroutinen eine bestimmte STACK-Größe benötigen; z.B. benutzen die E/A-Routinen etwa 4K Bytes STACK. Diese Größe ist bei der Angabe im STACK-Parameter mit zu berücksichtigen.

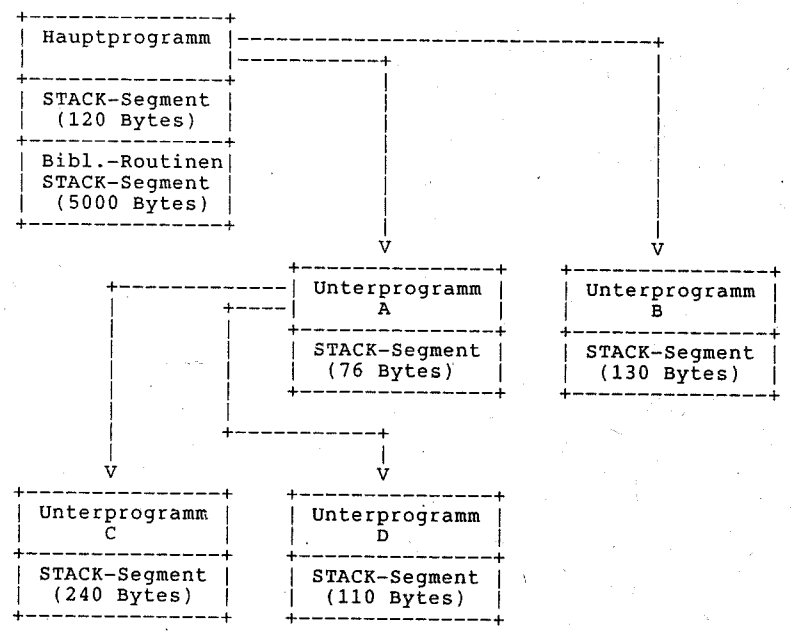

Bild 1: STACK-Größe in FORTRAN77-Programmen

4. Generierung des Laufzeitsystems für FOR77

An den Anwender wird eine Verteilungsdiskette ausgeliefert, die die in Tabelle 5 aufgeführten Bestandteile des Laufzeitsystems enthält.
Der auf der Verteilungsdiskette bereitgestellte Generierungsjob F77RUN.BAT gestattet es dem Anwender, ein für seine Bedingungen optimales Laufzeitsystem zu erzeugen. Der Generierungsjob wird wie folgt aufgerufen:

```
x:F77RUN x:
```

Dabei ist x das Laufwerk mit der Verteilungsdiskette.
Der Job führt den Anwender im Dialog und ist selbsterklärend aufgebaut. Deshalb sollen an dieser Stelle nur die prinzipiellen Möglichkeiten zur Zusammenstellung des Laufzeitsystems erläutert werden.
Der Generierungsjob führt den Anwender bei der Erzeugung einer Laufzeitbibliothek F77LIB.LIB, die seinen Anwendungsbedingungen genügt. Das hat den Vorteil, daß beim Linken nur die vom Anwender erzeugten Objektmoduln angegeben werden müssen. Die Bibliothek F77LIB.LIB wird automatisch zur Auflösung externer Bezugnahmen durchsucht.
Nur in Ausnahmefällen sollten die Bestandteile des Laufzeitsystems separat genutzt werden, z. B. Erzeugen von ausführbaren Programmen für andere Rechnerkonfigurationen. Dann ist es erforderlich, diese Bestandteile beim Linken separat anzugeben.
Ein funktionsfähiges FORTRAN77-Laufzeitsystem muß folgende Bestandteile enthalten:

```
F77BAS.LIB

Einen der E/A-Objektmoduln
      F77EAS.OBJ (Entsteht aus den Moduln F77EAB.OBJ und
                  F77EAS.OBJ auf der Verteilungsdiskette)
      F77EAR.OBJ (Die Anwendung dieses Moduls setzt das
                  Vorhandensein des residenten E/A-Moduls
                  F77EAS.RES voraus (siehe unten).

      F77EAN.OBJ
sowie eine der Gleitkommabibliotheken
      F77P87.LIB
      F77E87.LIB
      F77N87.LIB
```

Eine vom Anwender wählbare Kombination dieser Bestandteile wird bei der Generierung des Laufzeitsystems zu der Bibliothek F77LIB.LIB zusammengefaßt. Der Anwender kann entscheiden, ob er im Zielverzeichnis der Generierung weitere, in die F77LIB.LIB nicht eingehende E/A-Objektmoduln und Gleitkommabibliotheken aufnehmen möchte.
Bei der Abarbeitung des Generierungsjobs F77RUN.BAT hat der Anwender folgende wesentliche Entscheidungen für die Erzeugung der Bibliothek F77LIB.LIB zu treffen:

- Arbeit mit Gleitkomma-Prozessor oder -Emulator
- Arbeit mit/ohne residenten E/A-Modul

Die erste Entscheidung sollte im Normalfall entsprechend den konkreten Hardware-Bedingungen getroffen werden. Sinnvollerweise wird F77P87.LIB in F77LIB.LIB eingebunden, wenn ein Gleitkomma-

Prozessor vorhanden ist, sonst F77E87.LIB.
Der residente E/A-Modul entsteht während der Generierung als Bestandteil des Ladeprogrammes F77RSL.EXE und wird damit bei dessen Ausführung geladen.
Die Arbeit mit dem residenten E/A-Modul bringt den Vorteil, daß in das ausführbare FORTRAN77-Programm beim Verbinden nur die wenig Platz benötigende Interface-Routine F77EAR.OBJ eingeht. Bei der Abarbeitung ist jedoch zu sichern, daß der bei der Generierung entstehende E/A-Modul F77EAS.RES hauptspeicherresident ist. Dazu ist es notwendig, nur einmal während der Betriebszeit des DCP durch Aufruf der Laderoutine F77RSL.EXE den E/A-Modul resident zu machen. Das Laden der für die Arbeit mit dem residenten E/A-Modul vorbereiteten ausführbaren FORTRAN77-Programme erfolgt dann wesentlich schneller, da der E/A-Modul nicht bei jedem Programmstart erneut geladen werden muß.

Tabelle 5: Inhalt der Verteilungsdiskette des Laufzeitsystems

Dateiname	Bedeutung
F77BAS.LIB	Initialisierungsroutinen, arithmetische Routinen sowie Standardfunktionen; notwendiger Bestandteil der zu generierenden Laufzeitbibliothek F77LIB.LIB
F77EAB.OBJ	Basisroutinen der FORTRAN77-E/A; notwendiger Bestandteil des zu erzeugenden E/A-Objektmoduls F77EAS.OBJ und des residenten E/A-Moduls
F77EAS.OBJ	Interface-Routine für F77EAB.OBJ; bildet zusammen mit F77EAB.OBJ den E/A-Objekt-Modul F77EAS.OBJ
F77EAN.OBJ	Pseudo-E/A-Routinen; kann anstelle der FORTRAN77-E/A verwendet werden, wenn die zu bearbeitenden Programme keine FORTRAN77-E/A enthalten.
F77EAR.OBJ	Interfaceroutine für residenten E/A-Modul; notwendig, wenn residenter E/A-Modul genutzt werden soll.
F77P87.LIB	Gleitkomma-Initialisierung, wenn ein Gleitkomma-Prozessor vorhanden ist.
F77E87.LIB	Gleitkomma-Initialisierung und Gleitkomma-Emulator, wenn kein Gleitkomma-Prozessor vorhanden ist.
F77N87.LIB	Pseudo-Gleitkomma-Routinen; kann anstelle von F77P87.LIB oder F77E87.LIB genutzt werden, wenn in den zu verarbeitenden Programmen keine Gleitkommadaten verarbeitet werden.
F77TBP.OBJ	Hilfsroutine zur Erzeugung des residenten E/A-Moduls, falls ein Gleitkomma-Prozessor vorhanden ist.
F77TBE.OBJ	Hilfsroutine zur Erzeugung des residenten E/A-Moduls, falls kein Gleitkomma-Prozessor vorhanden ist.
F77RSL.OBJ	Laderoutine für residenten E/A-Modul
F77RUN.BAT	Generierungsjob für das Laufzeitsystem
F77RUN.MSO	Generierungsmitteilungen
F77RN1.EXE	Programm für Generierung
F77SERV.LIB	Service-Routinen (Beschreibung siehe F77INFO.TXT)
F77INFO.TXT	Informationen zur aktuellen Ausgabe von FOR77

5. Verbinden von FORTRAN77-Programmen

Nach dem Übersetzen aller zu einem FORTRAN77-Programm gehörenden Programmeinheiten besteht der nächste Arbeitsschritt aus dem Verbinden von Objektmoduln zu einem ausführbaren Programm. Das Ergebnis des Linkprozesses ist ein verschiebbarer Lademodul, der in eine Datei mit dem Dateityp EXE ausgegeben wird.
Die zu verbindenden Objektmoduln setzen sich wie folgt zusammen:
- Objektmoduln, die als Ergebnis der Übersetzung von FORTRAN77-Programmeinheiten entstanden sind.
- Objektmoduln, die durch Assemblierung von in Assembler geschriebenen Unterprogrammen entstanden sind.
- Objektmoduln aus der Laufzeitbibliothek F77LIB.LIB (siehe Abschnitt 4.) und aus privaten Bibliotheken. Auf diese Objektmoduln existieren noch nicht aufgelöste externe Bezugnahmen. Die Auflösung dieser Bezugnahmen erfolgt vom Programmverbinder automatisch.

Im Betriebssystem DCP steht der Programmverbinder LINK zur Verfügung. Diese Komponente bietet eine Vielzahl von Möglichkeiten, die an dieser Stelle nicht beschrieben werden. Dazu muß auf [2] verwiesen werden. In diesem Abschnitt sollen nur die prinzipiellen Möglichkeiten gezeigt werden, die zum Aufbau eines lauffähigen FORTRAN77-Programmes notwendig sind.

5.1. Aufruf des Linkers LINK

Der Linker kann gestartet werden, indem über die Tastatur das LINK-Kommando eingegeben wird. Wird das Kommando LINK ohne zusätzliche Angaben eingegeben, so erscheinen auf dem Terminal vier Anforderungen, die wie folgt beantwortet werden müssen:

Anforderung	Antworten
Objektmoduln [.OBJ]:	[d:][Pfad][Dateiname[.erw]] [+[d:][Pfad][Dateiname[.erw]]]...
Ausführungsdatei [Dateiname.EXE]:	[d:][Pfad][Dateiname[.erw]]
Listdatei [NUL.MAP]:	[d:][Pfad][Dateiname[.erw]]
Bibliotheken [.LIB]:	[d:][Pfad][Dateiname][.erw]] [+[d:][Pfad][Dateiname[.erw]]]...

Die Standardannahmen werden in eckigen Klammern am Ende des Anforderungstextes dargestellt. Nur für die Anforderung 'Objektmoduln' ist eine Antwort erforderlich.
Für die Bibliotheken können entweder die Dateispezifikationen aufgelistet oder einfach die Eingabetaste betätigt werden. Wird die Eingabetaste betätigt, so durchsucht das LINK-Programm standardmäßig die Hauptbibliothek F77LIB.LIB nach ungelösten externen Bezugnahmen.
Nach jeder der Anforderungen kann die Eingabe für den Linker durch ein Semikolon abgeschlossen werden, so daß keine weiteren Anforderungen erscheinen.
Am Ende jeder der Antworten auf die vier Anforderungen können ein oder mehrere Parameter spezifiziert werden. Dazu sind lediglich der Schrägstrich (/) und der erste Buchstabe des Parameters anzugeben.

Der Parameter /HIGH darf beim Verbinden von FOR77-Programmen nicht angegeben werden.
Mit dem Parameter /STACK:Größe kann die Größe des Stapels überschrieben werden, die von FOR77 bzw. vom Assembler bereitgestellt wurde (siehe Abschnitt 3.6.).
Soll das automatische Durchsuchen der vom FOR77-Compiler spezifizierten Bibliothek F77LIB.LIB entfallen, muß der Parameter /NOD angegeben werden.
Der Linker kann auch durch die Eingabe einer kompletten Befehlszeile oder durch die Angabe einer automatischen Antwortdatei gestartet werden (siehe [2]).

5.2. Benutzung des FOR77-Laufzeitsystems

Der FOR77-Compiler erzeugt externe Bezugnahmen auf Routinen, die zur Ausführungszeit benötigt werden. Diese Routinen sind in der generierten Laufzeitbibliothek F77LIB.LIB enthalten. Diese Bibliothek wird vom Linker automatisch zur Auflösung der externen Bezugnahmen durchsucht. Folgendes Beispiel demonstriert den Linker-Aufruf, wie er im Normalfall zum Verbinden eines FOR77-Objektmoduls notwendig ist:

```
C > LINK
Objektmoduln[.OBJ]: objektmodulname CR
Ausführungsdatei[objektmodulname.EXE]: CR
Listdatei [NUL.MAP]: CR
Bibliotheken[.LIB]: CR
oder einfacher:
C > LINK objektmodulname; CR
```

Auf der Verteilungsdiskette für das Laufzeitsystem sind zwei Bestandteile enthalten, die beim Linken genutzt werden sollten, um nicht benötigte Laufzeitroutinen durch Pseudoroutinen zu ersetzen. Die Bibliothek F77N87.LIB sollte verwendet werden, wenn keine Gleitkommadatentypen im Programm auftreten.
Der Objektmodul F77EAN.OBJ sollte verwendet werden, wenn keine FORTRAN77-E/A im Programm genutzt wird.
Da diese Bestandteile im Normalfall nicht in die Laufzeitbibliothek F77LIB.LIB eingebunden sind, sind sie beim Aufruf des Linkers separat anzugeben.

Beispiel:

Soll der Objektmodul TEST.OBJ, in dem keine Gleitkommadatentypen benutzt werden, verbunden werden, so kann das mit folgendem Kommando erfolgen:

```
C > LINK  TEST,,,F77N87; CR
```

Generell ist zu beachten, wie bereits in Abschnitt 4 erwähnt, daß der E/A-Objektmodul oder eine der Gleitkommabibliotheken im LINK-Kommando explizit anzugeben sind, wenn sie bei der Generierung nicht in die F77LIB.LIB eingebunden wurden.

Beispiel:

Der Objektmodul TEST.OBJ soll zu einem ausführbaren Modul TEST.EXE verbunden werden, der auf einem Rechner mit Gleitkomma-Prozessor arbeiten und den residenten E/A-Modul nicht nutzen soll. Außerdem soll der Stack auf eine Länge von 5436 Bytes

festgelegt werden. Es wird vorausgesetzt, daß die F77LIB.LIB für die Arbeit mit dem Gleitkomma-Emulator und dem residenten E/A-Modul generiert wurde. Dann muß folgendes LINK-Kommando eingegeben werden:

```
C > LINK TEST+F77EAS,,,F77P87/STACK:5436; CR
```

5.3. Erzeugung einer Überlagerungsstruktur

Der Linker LINK bietet die Möglichkeit, einen Lademodul (EXE-Datei) zu erzeugen, der eine Hauptspeicherüberlagerung realisiert. Dieser Lademodul besteht aus der Wurzel und den einzelnen Überlagerungsteilen. Beim Aufruf des Lademoduls wird der Wurzelbestandteil geladen und ein Ladebereich bereitgestellt, der so groß ist, daß der größte Überlagerungsteil darin Platz findet. Alle Überlagerungsteile nutzen diesen Ladebereich gemeinsam. Die Spezifizierung der Überlagerungsteile erfolgt im LINK-Kommando bei der Angabe der Objektmoduln. Jede in runde Klammern eingeschlossene Liste von Objektmoduln repräsentiert einen Überlagerungsteil.

Beispiel:

```
Objektmoduln[.OBJ]: a + (b+c) + d + (e)
```

Die Moduln (b+c) und (e) bilden je einen Überlagerungsteil. Die Moduln (a,d) und alle Routinen, die beim Linken aus der Bibliothek hinzugefügt werden, bilden die Wurzel.
Das Laden der Überlagerungsteile geschieht bei Bedarf automatisch. Nur ein Überlagerungsteil kann zur gleichen Zeit resident sein. Gleiche Eingangspunktnamen in verschiedenen Überlagerungsteilen sind nicht erlaubt.
Zur Realisierung der Überlagerung wird ein Interrupt benutzt. Standardmäßig ist das der Interrupt 205. Wird ein anderer Interrupt gewünscht, kann mit dem LINK-Parameter

```
/OVERLAYINTERRUPT : dezimale Interruptnummer
```

ein anderer Interrupt spezifiziert werden.

5.4. Anschluß von Assembler-Programmen

FOR77 bietet die Möglichkeit, vom Anwender geschriebene Assemblerprogramme nach der Übersetzung in ein FORTRAN77-Programm einzubinden.
Solche Assembler-Unterprogramme müssen den Konventionen der Objektmoduln entsprechen, die vom FOR77-Compiler erzeugt werden (siehe Abschnitt 3.6.).

An dieser Stelle sollen besonders die Anschlußbedingungen beim Betreten und Verlassen von Assembler-Unterprogrammen betrachtet werden.
Der Name eines externen Unterprogramms wird vom FOR77-Compiler wie folgt vereinbart:

```
EXTRN  up-name: FAR
```

Daraus folgt, daß ein eigenes Assembler-Unterprogramm stets eine Prozedur vom Typ FAR sein muß. Beim Betreten eines Unterprogramms enthalten die Register SS und DI die Adresse der Argumentliste. Diese Liste hat folgenden Aufbau:

Offset	Länge	Inhalt
0	4	[A(Funktionswert)]
4	4	A(1.Argument)
.	.	.
.	.	.
.	.	.
x*4	4	A(x.Argument)

Am Offset 0 ist immer Platz für die Rückgabe eines Funktionswertes reserviert. Hat das Unterprogramm keinen Funktionswert, bleiben die ersten 4 Byte der Argumentliste unbenutzt. Ist ein Argument vom Datentyp CHARACTER, zeigt A(Argument) auf einen Beschreiber mit folgendem Aufbau:

Offset	Länge	Inhalt
0	4	A(Zeichenkettenargument)
4	2	Länge des Zeichenketten-arguments

Vor dem Rücksprung in das aufrufende Programm müssen folgende Zustände hergestellt werden:
- Alle Register (außer DI) müssen den Inhalt zum Zeitpunkt des Aufrufes haben.
- Die Gleitkomma-Umgebung muß den Zustand zum Zeitpunkt des Aufrufes haben.
- Der Gleitkomma-Stapel muß leer sein.

6. Ausführen von FORTRAN77-Programmen

Ein FORTRAN77-Programm wird ausgeführt, wenn es unter dem vom Linker erzeugten Dateinamen aufgerufen wird. Dazu wird die Dateispezifikation des Programms als Kommando auf dem Terminal eingegeben.
FOR77-Programme übergeben beim Beenden einen Rückkehrcode, der über ERRORLEVEL in einem Stapelverarbeitungsjob abgefragt werden kann. Das FOR77-Laufzeitsystem übergibt als Rückkehrcode den binären Wert des niederwertigsten Bytes des COMMON-Bereiches mit dem Namen F77RET. Dieser Name ist reserviert. Standardmäßig übergibt ein normal beendetes Programm den Rückkehrcode 0 und ein Programm, das nach der Ausgabe einer FOR77-Laufzeitfehlernachrichten beendet wird, den Rückkehrcode 255. Durch die Definition eines COMMON-Bereiches mit dem Namen F77RET im FORTRAN77-Programm, der eine Variable beliebigen Datentypes enthält und der Zuweisung eines Wertes zu dieser Variablen, kann der Rückkehrcode für normal beendete Programme vom FORTRAN77-Programmierer selbst festgelegt werden. Der Wert des Rückkehrcodes ist in diesem Fall der binäre Wert des niederwertigsten Bytes der im COMMON-Bereich enthaltenen Variablen.

6.1. Aufruf eines FORTRAN77-Programms

Zusätzlich können beim Aufruf Vorverbindungen zwischen Einheiten und FORTRAN77-Dateien hergestellt werden. Die vollständige Kommandozeile für den Aufruf ist:

```
'pname'    ['datspez'  |  'liste']
```

Dabei ist:

'pname'	die Dateispezifikation des FORTRAN77-Programms. Der Dateityp .EXE muß nicht angegeben werden.
'datspez'	die Dateispezifikation einer Datei. Diese Datei enthält in jeder Zeile eine Dateivorverbindung 'datver'.
'liste'	eine in die Zeichen "[" und "]" eingeschloßene Liste von Dateivorverbindungen der Form 'datver'[,'datver']...
'datver'	eine Dateivorverbindung der Form: UNIT'datnr' = 'fspez', z.B. UNIT01=CON:.
'datnr'	eine maximal dreistellige Einheitennummer (siehe [1], Abschnitt 6.1.2.).
'fspez'	eine Dateispezifikation.

Eine Dateispezifikation, die von der FORTRAN77-E/A verwendet wird, hat für Disketten- oder Festplattendateien die im DCP übliche Form.

```
[gerät:][pfad-name] dateiname[.erweiterung]
```

oder ist einer der folgenden Namen zur Beschreibung von speziellen Dateien oder Geräten.

X:, CON:, TTB:, TTU:	Terminal
Y:, LST:, PRN:, LPT1:	1. Drucker
LPT2:, LPT3:	weitere Drucker
Z:, NUL:	Pseudo-Datei
WRK:	temporäre Datei auf Standardgerät
WKA:, WKB:, ..., WKG:	temporäre Datei auf Gerät A,B,...,G
AUX:, COM1:, COM2:	durch das DCP festgelegte Geräte

Diese Namen sind reserviert. Die Namen CON, PRN, LPTx, NUL, AUX und COMx können auch ohne den abschließenden Doppelpunkt angegeben werden. Die verschiedenen Namen für das Terminal ermöglichen die Umleitung von Ein- und Ausgaben und eine gepufferte oder ungepufferte Eingabe von der Tastatur. Die besonderen Bedingungen der Anwendung all dieser Dateispezifikationen werden im Abschnitt 7. erläutert.
Die Möglichkeit, die Dateivorverbindungen in einer Datei zeilenweise zu fixieren, besteht deshalb, weil pro Kommandozeile nur 128 Zeichen zur Verfügung stehen. Dateivorverbindungen, die sich in einer solchen Datei oder in der Kommandozeile befinden, können für eine Einheitennummer mehrfach auftreten. Es ist aber zu beachten, daß die erste aufgetretene Dateivorverbindung wirksam wird.

6.2. Standard-Vorverbindungen von Dateien

Auch wenn keine Vorverbindungen beim Programmaufruf angegeben werden, existieren zwei Standardvorverbindungen, eine für UNIT005 als Eingabedatei und eine für UNIT006 als Ausgabedatei. Diese Vorverbindungen haben die Form:

```
UNIT005=CON:,   UNIT006=CON:
```

Sie sichern die Verwendbarkeit von PRINT-Anweisungen und von READ- und WRITE-Anweisungen, in denen der UNIT-Parameter nicht oder in denen UNIT=* angegeben wurde. Die Standard-Vorverbindungen können durch die Angabe eigener Vorverbindungen für UNIT005 und UNIT006 beim Programmaufruf überschrieben werden.

Als Beispiel soll ein FORTRAN77-Programm TSTBSP ausgeführt werden. Es benötigt Dateien mit den Einheitennummern 1 und 6:

```
A>B:TSTBSP [UNIT001=C:DAT1.TXT,UNIT006=LST] CR
```


7. FORTRAN77-Ein/Ausgabe im Betriebssystem DCP

7.1. E/A-Steuerwerte

Einige Werte für die Steuerung der E/A eines FORTRAN77-Programmes sind in einer Tabelle zusammengefaßt. Die Tabelle bildet einen Modul der Bibliothek F77LIB.LIB des generierten Laufzeitsystems. Durch Einschluß eines Objektmoduls, der eine derartige Tabelle enthält, können die angegebenen Standardwerte den Anforderungen des Programms angepaßt werden. Dieser Standardmodul hat folgende Struktur:

```
        PUBLIC    CONTROL_DATA
CONTROL_DATA:
        DW        64        ; maximale Einheitennummer
        DW        16        ; maximale Anzahl von gleich-
                              zeitig offenen Dateien
        DW       512        ; Maximalwert für variable
                              Satzlänge von formatierten
                              Dateien
        DW        80        ; Satzlänge für list-formatierte
                              Ausgabe
        DW         0        ; Größe der E/A-Region
        DW        11 DUP(0); reserviert
```

Mit der Angabe der Größe der E/A-Region wird festgelegt, wieviel Speicherplatz bei der Initialisierung der E/A reserviert wird. Der Platz in dieser Region wird dann von der E/A selbst verwaltet.
Dabei gilt für diesen Wert:

- > 0, Größe der Region in Byte
- = 0, Region belegt die Hälfte des freien Speicherplatzes
- < 0, der Absolutwert dieser Größe legt die Größe des freizulassenden Speicherplatzes fest.

Seitens der E/A entstehen folgende Speicherplatzanforderungen für diese Region:

- 2 Bytes für jede existierende Einheitennummer
- 160 Bytes für jede der möglichen gleichzeitig existierenden Dateiverbindungen
- 320 Bytes für andere Steuerbereiche
- 512 Bytes für list-formatierte Eingabe

Außerdem wird für die Verarbeitung von formatierten Dateien ein Ein- und Ausgabebereich benötigt, dessen Größe sich nach dem Maximalwert für variable Satzlänge oder der festen Satzlänge von formatierten Dateien richtet. Wird die Datei mit dem größten dieser Werte als erste eröffnet, wird nur ein solcher Ein-/Ausgabebereich benötigt.
Der Maximalwert für die Anzahl von offenen Dateien beträgt 95. Wird ein größerer Wert angegeben, so wird dieser Maximalwert angenommen. Die maximale Einheitennummer kann nicht größer als 255 festgelegt werden.

7.2. Dateiverbindungen

Im Betriebssystem DCP wird eine Datei durch eine Dateispezifikation (siehe Abschnitt 6.) beschrieben. Das Mittel der Sprache FORTRAN77 für den Zugriff auf eine Datei ist die Einheit. Für den Zugriff auf eine Datei muß eine aktive Verbindung zwischen dieser Datei und der zum Zugriff benutzten Einheit hergestellt werden. Diese Verbindung wird über die Einheitennummer und die Dateispezifikation erzeugt.
Aktive Dateiverbindungen entstehen
- durch OPEN-Anweisungen oder
- durch eine READ-, WRITE- oder PRINT-Anweisung, wenn für die angegebene Einheit eine Dateivorverbindung existiert.

Eine aktive Dateiverbindung muß eindeutig sein, d. h., es darf zur gleichen Zeit eine Einheit höchstens mit einer Datei aktiv verbunden sein. Eine aktive Dateiverbindung wird durch eine CLOSE-Anweisung oder am Programmende inaktiv.

Dateivorverbindungen können beim Aufruf des FORTRAN77-Programms erzeugt werden (siehe Abschnitt 6.). Alle Dateivorverbindungen sind inaktiv. Sie erlauben die Benutzung von Einheiten, ohne daß zuvor für diese Einheiten eine OPEN-Anweisung ausgeführt wurde. Eine Dateivorverbindung für eine Einheit bleibt unberücksichtigt, falls in einer OPEN-Anweisung für diese Einheit der FILE-Parameter mit einer Dateispezifikation angegeben wird. Existiert für eine Einheit keine Vorverbindung und fehlt in der OPEN-Anweisung der FILE-Parameter, wird eine aktive Dateiverbindung mit einer temporären Datei erzeugt (siehe Abschnitt 7.4.).
Einheiten und Dateien, die noch nicht oder nicht mehr Bestandteile von aktiven Dateiverbindungen sind, können jederzeit neue aktive Verbindungen eingehen.
Die maximale Einheitennummer und die maximale Anzahl von gleichzeitig aktiven Dateiverbindungen, d. h. von eröffneten Dateien, kann durch Konstanten in einer Tabelle von Steuerwerten festgelegt werden (siehe Abschnitt 7.1).

7.3. Dateistruktur und Dateizugriff

Dateien haben im Betriebssystem DCP keine durch irgendein Aufzeichnungsverfahren bedingte Satzstruktur. Eine Datei ist also nur eine Folge von Zeichen (Bytes).
Durch Satztrennzeichen CR LF oder durch die Verwendung des RECL-Parameters in einer OPEN-Anweisung kann einer Datei eine Satzstruktur aufgeprägt werden. Durch den RECL-Parameter wird die Satzlänge nur für die Dauer der aktiven Dateiverbindung festgelegt.
Wenn die Satzlänge nicht durch den RECL-Parameter festgelegt ist, hat jeder Satz eine durch seinen Inhalt festgelegte (variable) Länge. Der Maximalwert für diese Länge kann über die Tabelle von Steuerwerten (siehe Abschnitt 7.1.) geändert werden.
Die Aufprägung einer Satzstruktur durch Satztrennzeichen ist nur für Dateien mit formatierten Sätzen möglich. Die Satzstruktur, die durch Verwendung des RECL-Parameters aufgeprägt wird, hat jedoch Vorrang vor der durch Satztrennzeichen aufgeprägten Satzstruktur. Die Satzlänge für formatierte Sätze kann auch für sequentiellen Zugriff festgelegt werden. Als Satztrennzeichen werden folgende Zeichen oder Zeichenfolgen von einem FORTRAN77-

Programm erzeugt oder erkannt:

CR LF, FF, FF CR, CR FF, CR und CR CR.

Diese Zeichen dürfen deshalb in den formatierten Sätzen selbst nicht auftreten. Sind sie in auszugebenden Zeichenketten enthalten, wird die Satzstruktur verfälscht.
Die Struktur einer Datei, die von einem FORTRAN77-Programm mittels PRINT- oder WRITE-Anweisungen (mit Angabe des FMT-Parameters) erzeugt wurde, ist von der Eigenschaft PRINT der Datei abhängig.
Eine Datei hat die Eigenschaft PRINT, wenn ihre Sätze auf das Terminal oder den Drucker ausgegeben werden sollen (gültige Dateispezifikationen X:, CON:, TTB:, TTU:, LST:, PRN:, LPT1:, LPT2:, LPT3: und Y:) oder wenn die Datei in ihrer Dateispezifikation einen der folgenden Dateitypen hat:

LS?, MAP, MP?, PRT, PUN.

Statt des Zeichens ? kann ein beliebiges Zeichen (jedoch kein Leerzeichen) stehen.
Bei der Ausgabe von formatierten Sätzen in eine Datei mit der Eigenschaft PRINT wird das erste Zeichen als Steuerzeichen für einen Drucker realisiert. Eine Datei mit der Eigenschaft PRINT beginnt also immer mit einem Satztrennzeichen.
Durch Angabe von Benutzertabellen können für Bildschirm und Drucker jeweils 255 eigene Steuerzeichen und die stattdessen auszugebenden Zeichenfolgen definiert werden.
Eine derartige Benutzertabelle kann durch den Assembler erzeugt und durch den Linker mit dem Programm verbunden werden. Sie hat folgendes Format:

```
        PUBLIC  CONTROL_CHARACTER_xxx
CONTROL_CHARACTER_xxx :
        DB      stz,lg,folge
    [           ...                 ]
        DB      stz,lg,folge
        DB      -1                  ; Endekennzeichen (=0FFH)
        DB      fkz
```

wobei gilt:

xxx	CON für Bildschirm LST für Drucker
stz	umzusetzendes Steuerzeichen (0H-0FEH)
lg	Länge der Zeichenfolge (maximal:33=21H)
folge	statt des Steuerzeichens auszugebende Zeichenfolge
fkz	Fortsetzungskennzeichen (0 = keine Fortsetzung, 1 = Ankettung der Standards)

Standardmäßig sind folgende Zeichenfolgen definiert:

Steuerzeichen	Zeichenfolge	
	Bildschirm	Drucker
1	CLEAR HOME	FF CR
+	CR CR	CR CR
0	CR LF CR LF	CR LF CR LF
:	kein Zeichen	kein Zeichen
blank	CR LF	CR LF

Die Zeichen sind durch die symbolischen Namen des ASCII-Codes dargestellt oder repräsentieren eine Escapefolge.

CLEAR	ESC [2 J	
HOME	ESC [H	
LF	Line feed	0AH
FF	Form feed	0CH
CR	Carriage return	0DH
ESC	Escape	1BH

Bemerkung:
Die Escape-Folgen CLEAR und HOME sind eventuell bei Verwendung eines anderen Gerätedrivers für den Bildschirm nicht verfügbar.

Existiert ein Steuerzeichen in der Benutzertabelle nicht, und ist die Standardtabelle nicht angeschlossen, wird das letzte Zeichen aus der Benutzertabelle verwendet.
Existiert ein Steuerzeichen weder in der Benutzertabelle noch in der Standardtabelle, wird CR LF als Zeichenfolge verwendet.
Enthält die Benutzertabelle eine Zeichenfolge, die länger als 33 Zeichen ist, wird diese Tabelle nicht weiter durchsucht, aber es wird sofort die Standardtabelle angeschlossen, unabhängig davon, ob dieses durch das nicht mehr erkennbare Fortsetzungskennzeichen gewünscht wird oder nicht.
Bei der Ausgabe von formatierten Sätzen in eine Datei, die nicht die Eigenschaft PRINT hat, wird das erste Zeichen des Satzes nicht verändert. Als Satztrennzeichen wird die Zeichenfolge CR LF zu jedem Satz hinzugefügt.
Bei der Eingabe einer Datei mit der Eigenschaft PRINT wird das erste Zeichen jedes gelesenen Satzes aus dem Satztrennzeichen erzeugt. Hat die Datei nicht die Eigenschaft PRINT, werden alle Zeichen eingelesen, die zwischen den Satztrennzeichen stehen oder vor der Betätigung der CR-Taste eingegeben werden.
In unformatierten Sätzen sind alle Zeichen möglich. Deshalb können keine Zeichen als Satztrennzeichen verwendet werden. Eine Satzstruktur ist nur durch Festlegung einer Satzlänge zu erreichen. Deshalb ist die Verwendung des RECL-Parameters auch für sequentiellen Zugriff erlaubt.
Besteht eine Datei aus unformatierten Sätzen und wurde die Dateiverbindung ohne den RECL-Parameter erzeugt, so hat die Datei keinerlei Satzstruktur. Die durch eine E/A-Anweisung einzulesende oder auszugebende Datenmenge wird nur durch Anzahl und Typ der Datenlistenelemente bestimmt.
Beim Ausführen einer BACKSPACE-Anweisung für eine unformatierte Datei ohne festgelegte Satzlänge wird die aktuelle Dateiposition um 1 Byte zurückgestellt. Wurde zum Beispiel zuletzt eine REAL-Variable gelesen, können nach Ausführung von 4 BACKSPACE-Anwei-

sungen dieselben Bytes der Datei nochmals gelesen werden.
Auf Dateien mit formatierten als auch auf Dateien mit unformatierten Sätzen kann der Zugriff sequentiell und direkt erfolgen. Tabelle 6 enthält eine Übersicht, welche Parameterangaben zu welcher Zugriffsart führen.

Tabelle 6: Dateizugriffsarten

RECL-Parameter	ACCESS-Parameter	Zugriffsart
angegeben	nicht angegeben DIREKT	direkt
angegeben	SEQUENTIAL	sequentiell
nicht angegeben	nicht angegeben	sequentiell

Die Angabe der Satzlänge durch den RECL-Parameter ist also für den Direktzugriff notwendig und für den sequentiellen Zugriff zur Aufprägung einer Satzstruktur möglich. Der Direktzugriff zu einer Datei erfolgt über die relative Adresse eines Satzes in der Datei. Sie berechnet sich aus Satznummer (REC-Parameter) und Satzlänge (RECL-Parameter) wie folgt:

Relative Adresse des Satzes = (Satznummer-1) * (Satzlänge+n)

Durch den Wert n wird bei der Berechnung der relativen Adresse des Satzes die reale Länge der Sätze berücksichtigt.
Dabei gilt:
n = 0 - unformatierte Datei
n = 1 - PRINT-Datei
n = 2 - nicht PRINT-Datei
Der direkte Zugriff auf eine Datei ist nur möglich, wenn sich die Datei auf einer Diskette oder Festplatte befindet. Beim Direktzugriff auf eine Datei mit formatierten Sätzen mit einer READ- oder WRITE-Anweisung kann durch "/"-Formatelemente die Verarbeitung mehrerer Sätze gefordert werden.

7.4. Implementierungsbedingte Besonderheiten der FORTRAN77-E/A

7.4.1. Allgemeines

Abschließende Leerzeichen eines Satzes einer formatierten Datei werden nicht ausgegeben, wenn für die Datei kein RECL-Parameter angegeben wurde und wenn die Ausgabe in eine Disketten- oder Festplattendatei erfolgt, die die Eigenschaft PRINT hat oder wenn das Ausgabegerät ein Drucker ist.
Wird ein derart verkürzter Satz mit einer READ-Anweisung und einem oder mehreren A-Formatelementen gelesen, wird er immer als mit Leerzeichen aufgefüllt betrachtet.
Eine Datei auf einer Diskette oder Festplatte kann nicht mehrere aktive Dateiverbindungen gleichzeitig haben, das heißt, der parallele Zugriff über mehrere Einheitennummern auf eine Diskettendatei ist nicht möglich. Dagegen können mit den Geräten Terminal (CON:, TTB:, TTU:, X:) und Drucker (LST:, PRN:, LPTx:, Y:) mehrere FORTRAN77-Dateien über ihre Einheitennummer gleichzeitig aktiv verbunden sein.

Der UNIT-Parameter der Form UNIT=* ist in einer READ-Anweisung gleichbedeutend mit UNIT=5, in einer WRITE-Anweisung mit UNIT=6.

Temporäre Dateien erhalten einen Dateinamen der Form:

WORKn.TMP

n ist ein Wert zwischen 0 und 255. Eine temporäre Datei existiert physisch zwischen der Ausführung der zugehörigen OPEN- und CLOSE-Anweisung.
Wird eine WRITE-Anweisung für eine sequentielle Diskettendatei ausgeführt und die aktuelle Dateiposition ist nicht das Dateiende, so erfolgt das Verkürzen der Datei durch folgende Schritte:
- Umspeichern der verbleibenden Sätze in eine Hilfsdatei,
- Erzeugen einer Datei mit dem alten Namen,
- Umspeichern der Hilfsdatei in diese Datei,
- Löschen der Hilfsdatei.

Die Hilfsdatei wird auf dem gleichen Gerät angelegt, auf dem sich die zu verkürzende Datei befindet. Damit ist der Schreibzugriff bei ausreichendem Platz für die Hilfsdatei gesichert.
Bei der Ausführung einer E/A-Anweisung mit IOSTAT-Parameter erhält die im IOSTAT-Parameter angegebene Variable die in Tabelle 7 zusammengestellten Werte.

Tabelle 7: IOSTAT-Werte

IOSTAT-Wert	Bedeutung
-2	Leereingabe auf Tastatur
-1	EOF ist aufgetreten
0	Anweisung wurde fehlerfrei ausgeführt
Wert positiv	Fehlercode (siehe Anlage 3)

7.4.2. Arbeiten mit dem Terminal

Die Arbeit mit dem Terminal ist sowohl mit formatierten als auch mit unformatierten E/A-Anweisungen möglich. Dabei erfolgen die physischen E/A-Operationen bei Verbindung mit den Dateispezifikationen CON: und X: über die erweiterten Funktionen des Betriebssystems DCP. Derartige Ein- und Ausgaben können dabei zu Disketten- oder Festplattendateien oder zu Hilfsprogrammen (z. B. SORT) umgeleitet werden (siehe [3]).
Bei Verbindung mit den Dateispezifikationen TTB: oder TTU: werden Ausgaben immer auf den Bildschirm geleitet, und die Eingabe erfolgt immer von der Tastatur. Eine Umleitung ist nicht möglich.

Formatierte E/A

Bei der formatierten Ausgabe wird ein Satz erst ausgegeben, nachdem er im Hauptspeicher komplett aufgebaut wurde. Durch eine Eingabeoperation über die Tastatur wird bei der formatierten Eingabe ein kompletter Satz eingelesen.
Eine Eingabeoperation wird bei Verbindung mit den Dateispezifikationen CON:, X: oder TTB: durch die ENTER-Taste oder durch die Eingabe von CR oder CTRL-M bewirkt. Bei Verbindung mit der Dateispezifikation TTU: bewirkt jeder Tastendruck eine Eingabeoperation.

Ist der Satz kürzer als durch Formatelemente beschrieben, können die fehlenden Teile des Satzes durch die Formatelemente A oder Aw verarbeitet werden, so als ob der Satz bis zur vollen Länge mit Leerzeichen aufgefüllt wäre.
Ist die Dateiverbindung mit fester Satzlänge erfolgt, wird der Eingabebereich bis zur Satzlänge mit Leerzeichen aufgefüllt.
Bei der formatierten Eingabe wird für die Aufforderung zur Eingabe ein Doppelpunkt, gefolgt von einem Leerzeichen, auf die nächste Zeile ausgegeben. Weitere notwendige Eingaben werden bei Verwendung der listengesteuerten Aufbereitung durch ein Pluszeichen angefordert.
Endet die vorhergehende Ausgabe auf das Terminal mit einem Doppelpunkt, wird zur Eingabeanforderung nur ein Leerzeichen an den ausgegebenen Satz angefügt. Weitere Eingaben können auf der gleichen Zeile erfolgen, solange das Steuerzeichen aus einer WRITE- oder PRINT-Anweisung keinen Vorschub auf eine neue Zeile bewirkt.

Unformatierte E/A

Bei der unformatierten Ein- und Ausgabe erfolgt für jedes Element der Datenliste (für Variable, Teilketten und alle Feldelemente eines Feldes, dessen Name ohne Indexliste verwendet wird) eine physische E/A-Operation. Für jedes Element der Datenliste ist also einmal die ENTER-Taste zu betätigen, wenn die Verbindung mit einer der Dateispezifikation CON:, X: oder TTB: erfolgte. Bei Verbindung mit TTU: wird jedes Element der Datenliste durch einen einzigen Tastendruck definiert.
Der Inhalt der eigentlich einzulesenden Variablen wird bei einer Leereingabe nicht geändert.
Ist die Variable länger als die eingegebene Zeichenfolge, wird der Rest durch Leerzeichen aufgefüllt.

Bei der Arbeit mit der Dateispezifikation TTB: können durch eine Eingabeoperation maximal 255 Zeichen eingelesen werden. Es ist zu beachten, daß durch das Betätigen von Tasten oder Tastenkombinationen Zeichen oder Zeichenfolgen eingelesen werden. Die den Tasten zugeordneten Zeichen oder Zeichenfolgen sind abhängig vom für die Arbeit mit der Tastatur verwendeten Driver. Die meisten verwendeten Driver ordnen nur den Buchstaben- und Zifferntasten sowie den gebräuchlichsten Sonderzeichen ein einziges Zeichen entsprechend dem ASCII-Code zu. Bei Betätigen von anderen Tasten werden definierte Zeichenfolgen eingelesen (z.B. erweiterter ASCII-Code, siehe [3]). Bei der Arbeit mit der Dateispezifikation TTU: wird jeweils ein solches Zeichen oder eine solche Zeichenfolge bei einem Tastendruck bereitgestellt.

Das Dateiende kann bei der Eingabe über die Tastatur
- durch Eingabe von CTRL-Z und CR (ENTER-Taste)
- durch eine Leereingabe (ENTER-Taste ohne vorheriges Betätigen einer anderen Taste)

dem FORTRAN77-Programm angezeigt werden.

Ist in einer READ-Anweisung weder der END-Parameter noch der IOSTAT-Parameter angegeben, und es erfolgt eine Leereingabe, wird das Programm mit dem Fehlercode 1285H abgebrochen. Die Ausführung des Programms kann nach einer Leereingabe ohne Einschränkungen fortgesetzt werden, wenn die READ-Anweisung den END- oder den IOSTAT-Parameter enthält. Bei dem anderen EOF-Zustand kann die Arbeit mit dieser UNIT erst nach einem CLOSE fortgesetzt werden.

7.5. Benutzung des residenten E/A-Moduls

Wenn der residente E/A-Modul bei der Generierung des Laufzeitsystems erzeugt wurde, kann er zur Ausführung von FORTRAN77-Programmen verwendet werden. Beim Verbinden des Programmes muß dazu der Modul F77EAR.OBJ angefordert werden, falls dieser nicht Bestandteil der Bibliothek F77LIB.LIB geworden ist. Dieser Modul stellt über den reservierten Interrupt 144 die Verbindung zu dem residenten E/A-Modul her.
Der residente E/A-Modul wird durch einfachen Aufruf des Programmes F77RSL.EXE geladen.

8. Gleitkomma-Arithmetik in FORTRAN77

FOR77 für DCP benutzt für die Realisierung der Gleitkomma-Arithmetik den in [2] beschriebenen Prozessor oder einen mit FOR77 gelieferten Emulator mit gleichem Funktionsumfang.
In diesem Abschnitt werden nur die Informationen geliefert, die für das Verständnis der FORTRAN77-spezifischen Anwendung notwendig sind. Weitere Details sind [2] zu entnehmen. Die Semantik der durch den FORTRAN77-Compiler erzeugten Gleitkomma-Befehle ist auch in [2] beschrieben. Die im folgenden gegebenen Aussagen zum Gleitkomma-Prozessor gelten in gleicher Weise für den Gleitkomma-Emulator.

8.1. Gleitkomma-Prozessor

Für die internen Operationen verwendet der Prozessor einen speziellen Datentyp (im folgenden TEMPREAL genannt) mit folgendem Gültigkeitsbereich:

```
3.4*10**-4932 < |x| < 1.2*10**4932  oder |x| = 0
```

Ein TEMPREAL hat folgenden internen Aufbau:


```
S     : Vorzeichenbit
|     : Position des impliziten Kommas
I     : erstes Bit der Mantisse
Bias  : 16383 (3FFFH)
```

Die Befehle des Prozessors gestatten es, auch explizit mit Operanden vom Datentyp TEMPREAL zu arbeiten. Bei der Realisierung der Standardfunktionen von FORTRAN77 wird dieser Datentyp verwendet, um möglichst wenig Genauigkeitsverluste zu erhalten. Zu diesem Zweck werden die Argumente vor der Verarbeitung in TEMPREAL-Argumente konvertiert. Ebenso wird das TEMPREAL-Ergebnis in den Zieldatentyp umgewandelt.

Die wesentlichen Informationen zur Steuerung des Prozessors und zu seinem gegenwärtigen Zustand werden in einem Steuerwort (Control Word=CW) bzw. einem Statuswort (SW) gehalten.
Das CW wird benutzt, um den Prozessor an bestimmte Anwendungsbedingungen anzupassen. Es hat folgenden Aufbau:

Masken für Ausnahmebedingungen:		Bit
INVALID OPERATION (1)	-+	0
DENORMALIZED OPERAND (1)	|	1
ZERODIVIDE (1)	> (5)	2
OVERFLOW (1)	|	3
UNDERFLOW (1)	|	4
PRECISION (1)	-+	5
Interruptmaske (1)		7
Precision-Modus (2)		8-9
Rundungs-Modus (3)		10-11
Infinity-Modus (4)		12

Die Bits 6 und 13-15 sind nicht signifikant.

(1) Bedeutung der Bitbelegung:
0 = nicht maskiert oder Interrupt aktiv
1 = maskiert oder Interrupt nicht aktiv

(2) Bedeutung der Bitbelegung:
00 = 24 Bits Mantisse
01 = reserviert
10 = 53 Bits Mantisse
11 = 64 Bits Mantisse

(3) Sei Z das mathematisch exakte Ergebnis einer arithmetischen Operation und Z1,Z2 die im Zieldatentyp exakt darstellbaren Nachbarn von Z mit Z1 <= Z <= Z2, so gilt:

00 = die Zahl von Z1 und Z2, die Z am nächsten liegt; falls |Z1-Z| = |Z2-Z|, so wird von Z1 und Z2 die Zahl gewählt, deren letztes Bit 0 ist.
01 = Z1 (Abrunden gegen - unendlich)
10 = Z2 (Aufrunden gegen + unendlich)
11 = die Zahl von Z1 und Z2, die näher zu Null liegt (Abschneiden)

(4) Bedeutung der Bitbelegung:
0 = projektiv (+ unendlich und - unendlich werden gleich behandelt)
1 = affin (+ unendlich und - unendlich werden unterschieden)

(5) Wenn eines dieser Bits auf 0 gesetzt ist, muß auch Bit 7 (Interruptmaske) auf 0 gesetzt sein, damit beim Auftreten einer maskierten Ausnahmebedingung eine Interruptbehandlung angefordert wird.

Das Statuswort (SW) enthält Informationen über den aktuellen Zustand des Prozessors. Es hat folgenden Aufbau:

	Bit
INVALID OPERATION (1)	0
DENORMALIZED OPERAND (1)	1
ZERODIVIDE (1)	2
OVERFLOW (1)	3
UNDERFLOW (1)	4
PRECISION (1)	5
Unterbrechungsanforderung (2)(6)	7
Rückkehrcode (3)	8-10, 14
Stackpointer (4)	11-13
(5)(6)	15

(1) Ist das entsprechende Bit mit 1 belegt, so ist die Ausnahmebedingung aufgetreten.
(2) Ist dieses Bit mit 1 belegt, so ist eine Unterbrechung aufgetreten (bei FORTRAN77 nur im Falle "INVALID OPERATION" möglich).
(3) Der Rückkehrcode wird für bedingtes Verzweigen benötigt.
(4) Gibt das aktuelle erste Element des Gleitkomma-Stacks des Prozessors an.
(5) Eine Belegung mit 1 gibt die Abarbeitung eines Gleitkomma-Befehls an.
(6) Werden vom Gleitkomma-Emulator nicht gewartet.

Das Statuswort wird in die FORTRAN77-Laufzeitfehlermeldungen einbezogen. Für die FORTRAN77-Anwender sind nur die Bits 0 bis 5 wesentlich, die die aufgetretenen Ausnahmebedingungen anzeigen. Während der Laufzeitinitialisierung von FORTRAN77 wird das CW wie folgt belegt:

0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	0

Damit werden folgende Festlegungen getroffen:

- Der Infinity-Modus ist projektiv, d. h. + unendlich und - unendlich werden nicht unterschieden (+ unendlich und - unendlich werden im Prozessor durch Spezialwerte dargestellt, die als Standardergebnis bei einer OVERFLOW-Ausnahmebedingung entstehen können).
- Für alle impliziten Rundungen (außer bei der Konvertierung von REAL zu INTEGER) wird der Rundungsmodus mit der Bitbelegung 00 (siehe oben) gewählt. Eine Rundung wird durchgeführt, wenn ein Resultat nicht im gewünschten Zieldatentyp exakt darstellbar ist.
- Alle Gleitkomma-Zwischenresultate werden mit einer Genauigkeit von 64 Bits berechnet.
- Bei allen Ausnahmebedingungen, außer "INVALID OPERATION", werden die Standardmaßnahmen des Prozessors wirksam. Bei "INVALID OPERATION" erfolgt ein Abbruch der Abarbeitung des Programms mit einer Fehlermeldung.
  Die Standardmaßnahmen werden im folgenden Abschnitt erläutert.
- Die Unterbrechungsmaske steht auf 0, so daß beim Auftreten von "INVALID OPERATION" stets die im vorangehenden Anstrich beschriebene Aktion wirksam wird.

8.2. Ausnahmebedingungen des Gleitkomma-Prozessors

Im folgenden werden die Maßnahmen beschrieben, die beim Auftreten einer Ausnahmebedingung während der Ausführung eines FORTRAN77-Programms gewählt werden, und es werden die Bedingungen angegeben, unter denen diese Ausnahmebedingungen auftreten können.
Für die Beschreibung der Maßnahmen sind folgende Begriffserläuterungen notwendig:

normalisierter Wert:
Eine normale Null oder ein Wert, dessen führendes Mantissen-Bit 1 und dessen Exponent größer als Null ist.

denormalisierter Wert:
Ein Wert, dessen Exponent gleich 0 ist und dessen führendes explizites oder implizites Mantissen-Bit 0 ist, aber keine normale Null darstellt.

unnormalisierter Wert:
Ein solcher Wert tritt nur im TEMPREAL-Format auf und entsteht aus einem denormalisierten Wert. Er ist aufgebaut wie ein normaler TEMPREAL-Wert, nur daß das führende Bit der Mantisse 0 ist. Mit unnormalisierten Werten kann eine normale TEMPREAL-Arithmetik durchgeführt werden. Es wird aber stets durch die 0 im führenden Mantissen-Bit angezeigt, ob bei der Berechnung Genauigkeitsverluste aufgetreten sind, die sich auf die darstellbaren Mantissenstellen auswirken.

INVALID OPERATION

Beim Auftreten dieser Ausnahmebedingung kommt es zu einer Unterbrechung, die nach Ausgabe einer Fehlermeldung zum Abbruch des Programms führt.
Eine solche Ausnahmebedingung tritt auf, wenn entweder ein Operand für eine bestimmte Operation nicht erlaubt ist oder die Operation selbst ungültig ist.
Diese Ausnahmebedingung weist im allgemeinen auf einen Programmierfehler hin, wie z. B. die Bezugnahme auf eine nicht initialisierte Variable.

Unter folgenden Bedingungen kann diese Ausnahmebedingung bei der Abarbeitung eines FORTRAN77-Programms auftreten:

- Ein oder mehrere Operanden einer Folge von Operationen sind unnormalisiert oder denormalisiert, und ein gültiges Resultat kann nicht garantiert werden, da nicht vertretbare Genauigkeitsverluste entstehen würden.

- Eine der folgenden Operationen wird versucht:

```
          unendlich + unendlich
          unendlich - unendlich
          0.0       * unendlich
          unendlich * 0.0
          unendlich / unendlich
          0.0       / 0.0
  normalisierte Zahl / unnormalisierte Zahl
  normalisierte Zahl / denormalisierte Zahl
```

- Einer der folgenden Vergleiche wird versucht:

```
          unendlich mit   0.0
          unendlich mit   normalisierter Zahl
          unendlich mit   unendlich
```

- Bei INT oder NINT ist das Argument zu groß, um in das INTEGER-Format zu passen.

- SQRT(x), wenn x eine negative Zahl, eine denormalisierte Zahl, eine unnormalisierte Zahl oder +- unendlich ist.

- SIN(x), COS(x), TAN(x), EXP(x), wenn x = +- unendlich oder |x| eine unnormalisierte Zahl >= 2 ** -63 ist.

- ASIN(x), ACOS(x), wenn x = +- unendlich oder |x| eine unnormalisierte Zahl >= 2 ** -63 oder |x| >= 1 ist.

- ATAN(x), wenn |x| eine unnormalisierte Zahl >= 2 ** -63 ist.

- LOG(x), LOG10(x), wenn 0 > x oder x eine unnormalisierte oder denormalisierte Zahl oder +- unendlich ist.

- SINH(x), COSH(x), TANH(x), wenn |x| eine unnormalisierte Zahl >= 2 ** -63 ist.

- ATAN2(y,x), wenn x und y unnormalisierte Zahlen sind und |y/x| >= 2 ** -63 gilt oder wenn |x| = |y| = 0 oder
  |x| = |y| = unendlich.

- MIN(x1,x2,...,xn), MAX(x1,x2,...,xn), wenn eines der Argumente unendlich ist.

- AMOD(x,y), DMOD(x,y), wenn y = +- unendlich ist oder x ist unnormalisiert oder denormalisiert.

- DIM(x,y), wenn x = y = +- unendlich ist.

- y ** x, wenn eine der folgenden Bedingungen erfüllt ist:

```
  y = x = +- 0
  y     normalisiert, x = +- unendlich
  y = - unendlich, x unnormalisierte Zahl oder
      +- unendlich oder +- 0
  y = + unendlich, x = +- 0
  y = + unendlich, x = + unendlich
  y = +- 0,        x = +- unendlich
  y = + unendlich, x = - unendlich
  y = + unendlich, x normalisiert (nicht ganzzahlig)
  y = - unendlich, x normalisiert (nicht ganzzahlig)
  y < 0 normalisiert, x normalisiert (nicht ganzzahlig)
```

- y ** i, wenn i < 0, i sich nicht in eine ganze Zahl von 32 Bit Länge umwandeln läßt und y eine unnormalisierte Zahl ist.

DENORMALIZED OPERAND

Beim Auftreten dieser Ausnahmebedingung kommt es zu keiner Programmunterbrechung, da sie für FORTRAN77 maskiert ist.
Diese Ausnahmebedingung tritt auf, wenn einer oder mehrere der Operanden denormalisierte Zahlen sind. Eine denormalisierte Zahl wird standardmäßig als Reaktion auf eine "UNDERFLOW"-Ausnahmebedingung geliefert.
Die denormalisierten Operanden werden in entsprechende unnormalisierte Operanden umgewandelt, und es wird anschließend die korrekte Arithmetik für unnormalisierte Operanden ausgeführt.

Tritt ein denormalisierter Operand als Argument von Standardfunktionen auf, so werden die in der folgenden Liste angegebenen Resultate geliefert oder Aktionen ausgeführt:

Funktion	Resultat
ACOS(x)	pi/2
ASIN(x)	x
SINH(x)	x
COSH(x)	1
SIN(x)	x
COS(x)	1
EXP(x)	1
ATAN(x)	x
LOG(x)	INVALID OPERATION
LOG10(x)	INVALID OPERATION
SORT(x)	INVALID OPERATION
TAN(x)	x
TANH(x)	x
ATAN2(y,x)	Denormalisierte Argumente werden in normalisierte umgewandelt und die Funktionsberechnung wird fortgesetzt.
DIM(y,x)	Reaktion wie bei ATAN2
MAX(x1,x2,...,xn)	Reaktion wie bei ATAN2
MIN(x1,x2,...,xn)	Reaktion wie bei ATAN2
SIGN(y,x)	
x denormalisiert	Vorzeichen wird von x übernommen
MOD(y,x)	
x denormalisiert	INVALID OPERATION

ZERODIVIDE

Die Ausnahmebedingung "ZERODIVIDE" tritt auf, wenn der Divisor einer Divisionsoperation gleich Null und der Dividend eine endliche Zahl ungleich Null ist.
Als Resultat wird "unendlich" geliefert, wobei sich das Vorzeichen aus den Operandenvorzeichen ergibt. Eine Programmunterbrechung erfolgt nicht.
"ZERODIVIDE" wird durch die FORTRAN77-Bibliothek auch gesetzt, wenn folgende Fälle auftreten:

- y ** x, y ** i, wenn y=0 und x < 0 bzw. i < 0
  (x, y reell, i ganzzahlig)
- TAN(x), wenn x = ungeradzahliges Vielfaches von pi/2
- LOG(x), LOG10(x), wenn x=0

OVERFLOW

Die Ausnahmebedingung "OVERFLOW" tritt auf, wenn ein gerundetes Resultat endlich ist, aber sein Exponent für den Resultatdatentyp zu groß ist.
Diese Ausnahmebedingung führt zu keiner Programmunterbrechung.
Das Resultat wird gleich "unendlich" gesetzt und die PRECISION-Ausnahmebedingung im Steuerwort angezeigt.
Für EXP(x), SINH(x), COSH(x), y ** x und y ** i wird die "OVERFLOW"-Ausnahmebedingung durch die FORTRAN77-Bibliothek nach Prüfung der Argumente bereits vor Ausführung der Funktionsberechnung gesetzt.

UNDERFLOW

Die Ausnahmebedingung "UNDERFLOW" tritt auf, wenn eine der folgenden Bedingungen erfüllt ist:

- Ein gerundetes Resultat einer Operation hat einen für den Resultattyp zu kleinen Exponenten.

- Das Ergebnis einer Multiplikation oder Division, deren Operanden verschieden von Null sind, ist nicht von Null unterscheidbar.

Das Resultat nach Auftreten der Ausnahmebedingung "UNDERFLOW" ist eine korrekt gerundete denormalisierte Zahl oder Null.
Eine Programmunterbrechung tritt nicht auf.

PRECISION

Die Ausnahmebedingung "PRECISION" tritt auf, wenn das korrekt gerundete Resultat einer Operation vom ungerundeten Resultat verschieden ist oder in dieser Operation vorher eine "OVERFLOW"-Ausnahmebedingung aufgetreten ist.
Als Ergebnis wird stets das korrekt gerundete Resultat geliefert.
Eine Programmunterbrechung tritt nicht auf.

9. Fehlernachrichten

9.1. Fehlernachrichten zur Übersetzungszeit

Zur Übersetzungszeit werden die lexikalischen, syntaktischen und semantischen Fehler innerhalb einer FORTRAN77-Programmeinheit durch die Pässe des Compilers erkannt und in der Fehlerdatei name.ERR aufgelistet. Eine vollständige Liste aller Fehlernachrichten zur Übersetzungszeit ist in Anlage 2 enthalten.
Ist die Option Q (siehe Abschnitt 3.4.1.) unwirksam, erfolgt zusätzlich und unmittelbar bei Erkennung des Fehlers die Ausgabe des Fehlers auf das Terminal in folgender Kurzform:

```
Error: nr, Sev: schw, Stmt: anr, Line:znr, Insert: einfügung
```

Dabei entsprechen die Angaben nr, schw, anr und znr genau den zugehörigen Spalten in der Fehlerdatei. Unter Insert ist die Einfügung angegeben, die im Fehlertext der Anlage 2 mit spezifiziert ist. Nach der Kurzform einer Fehlernachricht erscheint die Ausschrift:

```
Enter E or CTRL-C to abort or any other to continue
```

E als Antwort bewirkt ein sofortiges Verzweigen in den EPASS des Compilers zur Ausgabe aller bisher erzeugten Fehlernachrichten und anschließende Beendigung der Übersetzung. Bei CTRL-C wird die Übersetzung sofort abgebrochen.
In Abhängigkeit vom Ergebnis der Übersetzung wird vom Compiler ein Rückkehrcode (RC) gesetzt (siehe nachfolgende Übersicht).
Nach erfolgter Übersetzung kann zum Beispiel mit dem Kommando TYPE die Fehlerdatei ausgegeben werden. Sie hat folgenden Inhalt:

```
Error diagnostics

Error     Sev     Stmt     Message

'nr'      'schw'  'anr'    'znr'     'meldungstext mit einfügung'
 .         .       .        .                    .
 .         .       .        .                    .
 .         .       .        .                    .
```

Dabei ist 'nr' die Fehlernummer, 'anr' die Nummer und 'znr' die Anfangszeilennummer der fehlerhaften Anweisung.
Kann einem Fehler keine Anweisung zugeordnet werden, wird als Anweisungsnummer Null ausgegeben.
In der Spalte 'Sev' enthält die Fehlerdatei wie auch die Kurzform einer Fehlernachricht die Fehlerschwere des aufgetretenen Fehlers. Sie hat folgende Bedeutung:

Sev	RC	Bedeutung
T	16	Abbruchfehler; Der Fehler ist so schwerwiegend, daß eine Fortsetzung der Übersetzung unmöglich ist.
S	12	Schwerer Fehler; Der Fehler ist so schwer, daß eine Codeerzeugung (CPASS) nicht sinnvoll ist. Die Übersetzung bricht nach der globalen Stufe (GXPASS) ab. Die Codeerzeugung kann aber mit der Option COMPILE erzwungen werden. Die fehlerhafte Anweisung wird gestrichen und durch einen Aufruf des Laufzeitfehlerbehandlers ersetzt.
E	8	Fehler; Der aufgetretene Fehler kann noch zu einer sinnvollen Codeerzeugung führen. Falls das nicht gewünscht wird, kann durch NOCOMPILE(E) auch in diesem Fall der Abbruch nach der globalen Stufe erreicht werden.
W	4	Warnung; Der Fehler ist von geringer Bedeutung. Der Übersetzungsprozess wird nicht beeinträchtigt.
I	0	Information.

Während der Übersetzungszeit kann auch eine weitere Art von Fehlernachrichten entstehen, die auf Fehler im Compiler selbst zurückzuführen sind. Sie erscheinen auf dem Terminal und haben die Form:

```
Compiler error: 'nr'   stmt: 'anr'  line: 'lnr'  id: 'kennz'
oder
Compiler error: 'nr'
```

Bei Auftreten dieser Fehlerart ist über die zuständige Betreuungsgruppe Kontakt mit dem Entwickler aufzunehmen. In diesem Fall wird RC=20 gesetzt. Kommt es während einer Übersetzung zu Fehlerzuständen bei der Speicherplatzinitialisierung oder bei der Arbeit mit den Compilerdateien, so erscheint auf dem Terminal eine selbsterklärende Nachricht.

9.2. Fehlernachrichten zur Ausführungszeit

Alle Fehlernachrichten zur Ausführungszeit werden auf dem Terminal ausgegeben. Dabei werden folgende Fehlergruppen unterschieden:

- E/A-Fehler
- FORTRAN77-E/A-Fehler
- Festkomma- und Überlauffehler
- Fehler bei der Abarbeitung von Gleitkomma-Funktionen
- Gleitkommafehler *)
- Stop durch Übersetzungsfehler
- allgemeine Fehler

*) Es ist zu beachten, daß INTEGER*4-Operationen zu Gleitkommafehlern führen können, da einige dieser Operationen bei Verwendung des Gleitkommaprozessors über diesen realsiert werden.

In den folgenden Abschnitten werden die Nachrichten dieser Fehlerarten beschrieben. Dabei bedeuten:

'hexnr' - hexadeximale Fehlernummer. In Anlage 3 ist eine vollständige Liste aller Ausführungszeit-Fehlernummern und ihre Bedeutung zusammengefaßt.

'hexbasis':
'hexoffs' - hexadezimale Adresse, bei der der Fehler aufgetreten ist. Dabei entsteht die Adresse aus 'hexbasis'*16+'hexoffs'.

'pname' - Name der Programmeinheit, in der der Fehler auftrat.

'deznr' - dezimale Nummer einer Anweisung oder eines Übersetzungsfehlers.

'statwort' - Statuswort des Gleitkomma-Prozessors (siehe Abschnitt 8.).

'fktname' - Name einer FORTRAN77-Standardfunktion oder einer mathematischen Operation (siehe Tabelle 8).

'cd' - hexadezimaler Operationscode eines Gleitkomma-Befehls.

'addr' - hexadezimale Adresse.

Tabelle 8: Einfügungen in Fehlernachrichten zu Gleitkomma-Funktionen

Einfügung	Bedeutung
MAX	FORTRAN77-Standardfunktion
MIN	FORTRAN77-Standardfunktion
Y**I	Potenzierung; Exponent INTEGER
INT	FORTRAN77-Standardfunktion
ARCTAN(Y,X)	FORTRAN77-Standardfunktion
ARCTAN	FORTRAN77-Standardfunktion
ARCCOS	FORTRAN77-Standardfunktion
ARCSIN	FORTRAN77-Standardfunktion
TAN	FORTRAN77-Standardfunktion
COS	FORTRAN77-Standardfunktion
SIN	FORTRAN77-Standardfunktion
TANH	FORTRAN77-Standardfunktion
COSH	FORTRAN77-Standardfunktion
SINH	FORTRAN77-Standardfunktion
LOG10	FORTRAN77-Standardfunktion
LN	FORTRAN77-Standardfunktion
EXP	FORTRAN77-Standardfunktion
Y**X	Potenzierung; Exponent REAL
MOD	FORTRAN77-Standardfunktion
NINT	FORTRAN77-Standardfunktion
ANINT	FORTRAN77-Standardfunktion
AINT	FORTRAN77-Standardfunktion
DIM	FORTRAN77-Standardfunktion
SIGN	FORTRAN77-Standardfunktion
CSQRT	FORTRAN77-Standardfunktion
CEXP	FORTRAN77-Standardfunktion
CLOG	FORTRAN77-Standardfunktion
CSIN	FORTRAN77-Standardfunktion
CCOS	FORTRAN77-Standardfunktion
C**C	Potenzierung; Exponent COMPLEX

9.2.1. E/A-Fehlernachrichten

Auf dem Terminal erscheint folgende Nachricht:

```
*** Run-time I/O exception 'hexnr'
*** Near location 'hexbasis':'hexoffs'
*** In program 'pname' statement 'deznr'      1)
*** Job aborted.
```

9.2.2. FORTRAN77-E/A-Fehlernachrichten

Die Nachricht auf dem Terminal hat folgendes Aussehen:

```
*** Run-time FORTRAN77 I/O exception 'hexnr'
*** Near location 'hexbasis':'hexoffs'
*** In program 'pname' statement 'deznr'      1)
*** Job aborted.
```

Vor einer Fehlernachricht mit dem Fehlerwert 12FF kann ein Text ausgegeben werden, der den Fehler erläutert.

9.2.3. Festkomma- und Überlauffehler

Folgende Nachricht erscheint auf dem Terminal:

```
*** Run-time 'text' exception
*** Near location 'hexbasis':'hexoffs'
*** In program 'pname' statement 'deznr'      1)
*** Job aborted.
```

Dabei steht für 'text' eine der folgenden Zeichenketten:

INTEGER-OVERFLOW	Festkomma-Überlauf
INTEGER-ZERO DIVIDE	Festkomma-Nulldivision
RANGE	Überlauf von Feldgrenzen oder Teilkettengrenzen

9.2.4. Fehler bei der Abarbeitung von Gleitkomma-Funktionen

Es erscheint folgende Fehlernachricht auf dem Terminal:

```
*** Run-time 'fktname' exception 'statwort'
*** Near location 'addr'
*** In program 'pname' statement 'deznr'      1)
*** Job aborted.
```

9.2.5. Gleitkomma-Fehler

Fehler bei der Abarbeitung von Gleitkomma-Befehlen, die vom Compiler generiert wurden, führen zu folgender Form der Fehlernachricht auf dem Terminal:

```
*** Run-time floating-point exception 'statwort'
*** Instr opcode 'cd'
*** Memop address 'addr'
*** Near location 'addr'
*** In program 'pname' statement 'deznr'  1)
*** Job aborted.
```

9.2.6. Allgemeine Fehler

Alle anderen Fehler, die nicht näher klassifiziert werden können, erscheinen in folgender Form auf dem Terminal:

```
*** Run-time exception 'hexnr'
*** Near location 'hexbasis':'hexoffs'
*** In program 'pname' statement 'deznr'      1)
*** Job aborted.
```

9.2.7. Stop durch Übersetzungszeitfehler

Dieser Abbruch der Ausführung wird wie folgt auf dem Terminal protokolliert:

```
*** Run-time exception
*** Compilation error 'dexnr'
*** In program 'pname' statement 'deznr'      1)
*** Job aborted.
```

1) *** In program 'pname', wenn Option NOGOSTATEMENT für die Übersetzung angegeben ist.

10. Optimierung von FORTRAN77-Programmen

Die logische Stufe Optimierung erzeugt effektiven Objektcode, wenn eine der Compileroptionen OPT=1 oder OPT=2 wirksam ist und kein vorzeitiger Abbruch der Übersetzung durch einen Fehler oder eine Abbruchbedingung erfolgt. Nach dem Verbinden führt der optimierte Objektcode zu kleineren und schnelleren Programmen.
Folgende Optimierungstechniken können in Abhängigkeit von der Programmstruktur zur Anwendung kommen:

- Ausdrucksoptimierung
- Senkung des Berechnungsaufwandes bei der Feldadreßrechnung
- Verlagerung schleifeninvarianter Ausdrücke vor die Schleife

10.1. Optimierungsstufe 1 (OPT=1)

Ein wesentlicher Teil der Optimierung ist die Ausdrucksoptimierung.
Die Ausdrucksoptimierung erfaßt arithmetische und logische Ausdrücke, auch solche, die in Feldelement-Adreßberechnungen und in Argumentlisten von Unterprogrammaufrufen auftreten.
Treten im Programm formal gleiche Ausdrücke auf, und wird der Wert der in den Ausdrücken enthaltenen Variablen zwischen den Berechnungen der Ausdrücke nicht geändert, so wird nur der im Ablauf erste Ausdruck berechnet und dessen Wert anstelle der folgenden gleichen Ausdrücke verwendet.
Ausdrücke, deren Operanden Konstanten oder Variablen sind, denen zuvor ein fester Wert zugeordnet wurde, werden sofort berechnet und das Ergebnis dem Zieloperanden zugeordnet.
Wird eine Variable im Programm nur einmal mit einem Wert belegt, wird sie bei der Verwendung in Ausdrücken wie eine Konstante behandelt.
Die Feldelement-Adreßberechnung wird nicht nur für gleiche Feldelemente, sondern auch für Elemente gleich aufgebauter Felder und für unterschiedlich strukturierte Felder mit gleichen Indexausdrücken optimiert.
Argumentlisten werden optimiert, wenn mehrere in mindestens einem Argument übereinstimmen.

Arithmetische Operatoren werden auf möglichst niedriges Niveau zurückgeführt, um eine Verkürzung der Ausführungszeit zu erreichen (z. B. A ** 2 --> A * A).
Operationen, die unabhängig vom Wert der Variablen nur zu einem Ergebnis führen können, werden von der Optimierungsstufe ausgeführt.
Außer der oben beschriebenen Ausdrucksoptimierung wird innerhalb von DO-Schleifen eine Codeverbesserung durch Verlagerung schleifeninvarianter Teilausdrücke innerhalb arithmetischer Ergibtanweisungen in den Schleifenkopf vorgenommen. Dadurch werden solche Ausdrücke einer unnötigen wiederholten Berechnung entzogen. Dieser Optimierungseffekt ist insbesondere bei Feldelement-Adreßberechnungen wirkungsvoll.
Darüber hinaus wird für Ausdrücke, die hinsichtlich der Laufvariablen der DO-Schleife Polynome 1. Grades und ansonsten schleifeninvariant sind, eine inkrementale Funktion gebildet. Dadurch wird bei jedem Schleifendurchlauf nicht der vollständige Ausdruck neu berechnet, sondern sein (gespeicherter) Wert lediglich um den durch die Schrittweite der Laufvariablen hervorgerufenen Änderungsbetrag korrigiert.

10.2. Optimierungsstufe 2 (OPT=2)

Außer dem in Abschnitt 10.1. genannten Optimierungsumfang wird der Code in impliziten Schleifen verbessert. Solche Schleifen sind die durch den Programmierer mittels Sprunganweisungen gebildeten Zyklen. In ihnen werden ebenfalls Verlagerungen von schleifeninvarianten Codeteilen, ähnlich wie in Abschnitt 10.1. beschrieben, vorgenommen.

10.3. Allgemeine Bemerkungen zur effektiven Programmierung

Alle genannten Optimierungen verlängern deutlich die Übersetzungszeit, da dabei mitunter sehr große Programmabschnitte (Schleifen) global zu erfassen sind und eine Vervielfachung des Aufwandes im Falle von mehreren geschachtelten DO- oder impliziten Schleifen entsteht.

Der Programmierer kann bis zu einem gewissen Grade auf die Übersetzungszeiten Einfluß nehmen. Dazu gehört beispielsweise die Vermeidung der Berechnung konstanter Ausdrücke und der mehrfachen Berechnung gleicher Ausdrücke. Darüber hinaus sollte der Programmierer bei Verwendung von Schleifen jeder Art versuchen, schleifeninvariante Teile zu erkennen und durch deren Umordnung vor die Schleifen wiederholter Berechnung zu entziehen. Dadurch wird eine Schleifenoptimierung erleichtert, wenn auch in der Regel nicht überflüssig, da infolge Feldelement-Adreßberechnungen Optimierungsmöglichkeiten entstehen können.
Die für die Optimierungsstufen 1 und 2 erwähnten inkrementalen Funktionen bringen sicher erst Laufzeiteinsparungen, wenn die DO-Schleifen genügend häufig durchlaufen werden.

Anlage 1 Bereichsgrenzen und interne Darstellung der FORTRAN77-Datentypen

Datentyp	Länge(in Bytes)	Bereich
INTEGER*2	2	-32768<=x<=+32767
INTEGER*4	4	-2147483648<=x<=+2147483647
REAL*4 1)	4	10**(-37)<=|x|<=10**37
REAL*8 1)	8	10**(-307)<=|x|<=10**307
DOUBLE PRECISION	entspricht REAL*8	
COMPLEX 2)	2*4	Wertbereich für Real- und Imaginärteil entspricht dem REAL*4-Wertebereich
LOGICAL*1	1	FALSE, TRUE
LOGICAL*2	2	FALSE, TRUE
LOGICAL*4	4	FALSE, TRUE
CHARACTER*n	1<=n<=32767	

Erläuterungen:

1) Die interne Darstellung der Real-Datentypen entspricht der gültigen Darstellung der SHORT- bzw. LONG REAL-Datentypen des Gleitkomma-Prozessors.

2) Real- und Imaginär-Teil des komplexen Datentyps stehen im Speicher hintereinander und entsprechen der REAL*4-Darstellung.

Die interne Darstellung der LOGICAL-Daten hat folgendes Format:

00H steht für FALSE
FFH steht für TRUE

CHARACTER-Daten werden intern im ASCII-Code gespeichert.

Anlage 2 Fehlernachrichten zur Übersetzungszeit

22 I Line(s) shorter than 72 characters.
Programmeinheit enthält eine oder mehrere Zeilen, die kürzer als das Standardformat von 72 Zeichen sind.

42 E Error in control line.
Fehler in Steuerkarten.

43 T No FORTRAN text.
Keine FORTRAN-Anweisung gefunden.

44 W Text after column 72.
Der Text der FORTRAN-Anweisung geht über die Spalte 72 hinaus. Dieser Text wird als Kommentar betrachtet.

90 S Primary and secondary ENTRY incompatible.
Datenattribut und Länge der Eintrittspunkte widersprechen sich.

91 E Line @ is not an initial line.
Zeile "@" ist keine Anfangszeile.

92 E Too many digits for label in line @.
Die Marke in Zeile "@" enthält zu viele Ziffern.

93 E Label not complete in line @.
Marke nicht vollständig in Zeile "@".

94 E Line @ expects a continuation line.
Nach Zeile "@" wird eine Fortsetzungszeile erwartet.

95 S Statement starts with incorrect character in line @.
Anweisung in Zeile "@" beginnt mit ungültigem Zeichen.

96 E Invalid label in line @.
Ungültige Marke in Zeile "@".

97 E Invalid format in line @.
Ungültiges Format in Zeile "@".

98 E Unknown statement type in line @.
In Zeile "@" beginnt nicht erkennbarer Anweisungstyp.

99 S Analysis of statement impossible in line @.
In Zeile "@" beginnt nicht analysierbare Anweisung.

100 S Unallowed statement starts in line @.
In Zeile "@" beginnt unerlaubte Anweisung.

101 S Statement not permitted in this position.
Anweisung an dieser Stelle unzulässig. Sie wurde ignoriert.

102 S Invalid text after statement end ignored.
Unerlaubter oder nicht interpretierbarer Text nach Anweisungsende wurde ignoriert.

103 E Invalid character. Statement truncated.
Anweisung enthält ungültiges Zeichen. Nachfolgender Text wurde ignoriert.

104 E END statement not in last line.
END-Anweisung ist nicht die letzte Zeile.

105 S First operand LOGICAL.
Erster Operand im Vergleich ist vom Typ LOGICAL.

106 S Ending apostroph assumed at end of statement.
Fehlender Apostroph einer Zeichenkette wurde am Anweisungsende angenommen.

107 S Erroneous relational expression.
Zweiter Operand im Vergleich ist vom Typ LOGICAL oder CHARACTER- und arithmetischer Operand werden verglichen.

108 S ',' does not [immediately] follow an index expression near @.
Fehlerhafter Text im Indexausdruck für Feld "@" wurde ignoriert.

109 W @ can not be INTRINSIC because of actual arguments.
Die Datentypen oder die Anzahl der aktuellen Parameter erzwingen, daß "@" der Name einer externen Funktion statt einer INTRINSIC-Funktion ist.

110 S Erroneous type of @ operand.
Unzulässiger Typ des "@" Operanden in logischer, arithmetischer oder CHARACTER-Operation.

111 S ')' does not [immediately] follow last index expression near @.
Klammer nach Indexliste für Feld "@" fehlt, oder fehlerhafter Text vor dieser Klammer wurde ignoriert.

112 S Exponent not of type INTEGER.
Exponent im konstanten Ausdruck ist nicht vom Typ INTEGER.

113 S @ illegal in context.
Bezeichner "@" ist unpassend zum Kontext.

114 S Illegal concatenation with operand of *-length.
Verkettung außerhalb einer Ergibtanweisung enthält Operanden mit *-Länge.

115 E) inserted after arithmetic expression.
')' wurde nach arithmetischen Ausdruck eingefügt.

116 E Parenthesis or operator illegal.
Klammer oder Operator vor einem Element der Eingabeliste.

117 S Missing or wrong operand/expression.
Fehlender oder falscher Operand im Ausdruck oder fehlender Indexausdruck.

118 E Merging of COMPLEX and DOUBLE PRECISION.
Auftreten eines COMPLEX- und eines DOUBLE PRECISION-Operanden in einer arithmetischen Operation.

119 S Argument list missing.
Argumentliste fehlt in CALL-Anweisung.

120 S Unpaired parenthesis. Statement ignored.
Unpaarige Klammern. Anweisung wurde ignoriert.

129 E Invalid alternate return.
Alternativer Rücksprung ist nicht erlaubt.

130 S Implementation restriction. Too many labels in argument list.
Die Argumentliste enthält zu viele Marken.

131 E Character constant invalid. Blank assumed.
Zeichenkettenkonstante der Länge Null ist nicht erlaubt. Ein Leerzeichen wurde angenommen.

132 E Character constant truncated.
Zu lange Zeichenkettenkonstante wurde rechtsbündig abgeschnitten.

134 E Missing expression replaced by zero.
Fehlender Ausdruck wurde durch 0 ersetzt.

135 S Implementation restriction. Too many labels in label list.
Zu viele Marken in der Markenliste.

136 S Number @ exceeds value limit.
Zahl "@" außerhalb des Wertevorrates.

137 E Statement contains invalid exponent.
Anweisung enthält ungültigen Exponenten.

138 E Number too long.
Zahl hat zu viele Ziffernstellen.

139 E Missing END statement inserted.
Fehlende END-Anweisung wurde eingefügt.

140 S END statement exceeds initial line.
END-Anweisung nicht vollständig auf Anfangszeile.

141 E Statement too long.
Anweisung ist zu lang.

142 E Statement over too many lines.
Anweisung hat zu viele Zeilen.

143 S Label not permitted in this statement.
Anweisung enthält unerlaubte Marke.

144 S Erroneous symbolic name.
Anweisung enthält fehlerhafte Bezeichnung.

145 S Error in name list.
Fehler in der Namenliste.

146 E Error in label list.
Fehler in der Markenliste.

147 E Invalid label list ignored.
Ungültige Markenliste wurde ignoriert.

148 S () assumed.
In der Anweisung wurden () angenommen.

149 E Missing @ inserted.
Das fehlende Zeichen "@" wurde eingefügt.

150 S Implementation restriction. Too many elements.
Implementierungsbedingte Einschränkung. FORMAT-Anweisung enthält zu viele Elemente.

151 E Format element @ incomplete.
Unvollständiges FORMAT-Element "@".

152 S @ empty format element(s).
"@" leere(s) FORMAT-Element(e) wurde(n) ignoriert.

153 E Implementation restriction. Number in @ too large.
Implementierungsbedingte Einschränkung. Zahl im FORMAT-Element "@" ist zu groß.

154 E Incorrect length in H format element.
Falsche Längenangabe im H-FORMAT-Element.

155 E Missing digits before @. Format element ignored,
Ziffernfolge vor X-, P- oder H-FORMAT-Element fehlt. Element wurde ignoriert.

156 S Format element beginning with @ not recognizable.
FORMAT-Element beginnend mit "@" ist nicht deutbar.

157 E Repeat specification illegal in this context.
Einem Wiederholungsfaktor folgt ein unzulässiges FORMAT-Element.

158 S Minus not followed by P format element.
Nach '-' folgt kein P-FORMAT-Element.

159 S Format element @ incomplete or zero.
Das FORMAT-Element "@" enthält unzulässige Ziffernfolge(n).

160 E Duplicate @-spezifier ignored.
Der doppelte "@"-Parameter wird ignoriert.

161 E Keyword specifier @ ignored.
Der in der E/A-Anweisung angegebene Schlüsselwortparameter "@" wurde ignoriert.

162 E Wrong specifier @.
Fehler im Parameter "@".

163 S Specifier @ missing or erroneous.
Parameter "@" fehlt oder hat falschen Typ.

164 E Missing comma inserted in data list.
In die Datenliste wurde ein fehlendes Komma eingefügt.

167 E No label in ERR or end-of-file specifier.
Der ERR- oder END-Parameter enthält keine Marke.

168 S UNIT=* not allowed in this statement.
UNIT=* ist in dieser Anweisung nicht erlaubt.

169 S Incompatible FMT and UNIT specifier.
Widersprüchliche UNIT- und FMT-Parameter.

172 S Error in element of data list.
Fehler in einem Datenlistenelement.

173 S List of data elements contains assumed size array.
Feld mit angenommener Größe in der Datenliste.

179 S DO-variable of incorrect type.
Die implizite DO-Variable ist nicht von Typ INTEGER, REAL oder DOUBLE PRECISION.

180 S Expression in implied-do-control of incorrect type.
Ausdruck in der Steuerung für implizites DO ist nicht vom Typ INTEGER, REAL oder DOUBLE PRECISION.

181 S Terminal expression in implied-do missing.
In der Steuerung für implizites DO fehlt der zweite Ausdruck.

184 E Missing input/output list.
Auf Komma folgt keine Datenliste.

185 E Redundant commas ignored.
Überflüssige Kommas in der Datenliste wurden ignoriert.

186 E Unpermitted increment expression. 1 assumed.
Unzulässiger Ausdruck für Schrittweite.1 wurde angenommen.

187 S REC specifier not allowed.
Unerlaubter Parameter REC für interne Datei.

188 S Illegal format specifier for internal file.
Listengesteuerte oder unformatisierte E/A für interne Datei nicht erlaubt.

189 E Illegal positional specifier ignored.
Stellungsparameter steht nicht am Listenanfang und wurde ignoriert.

190 E @ illegal in this context.
"@" ist an dieser Stelle unzulässig.

201 S Symbolic name @ illegal in this context.
Bezeichner "@" ist ungültig in Kontext.

202 S Symbolic name @ neither in COMMON nor dummy argument.
Variable "@" muß außerhalb der Routine einen Wert erhalten.

203 S @ illegal in left hand side of assignment.
Bezeichner "@" darf nicht auf der linken Seite der Ergibtanweisung auftreten.

204 S Array @ must be a dummy argument.
Feld "@" muß ein formaler Parameter sein.

206 S Illegal use of dummy argument @.
Fehlerhafte Anwendung des formalen Parameters "@" der Anweisungsfunktion.

207 S Intrinsic name @ used as actual argument.
INTRINSIC-Name "@" darf nicht als aktuelles Argument übergeben werden.

208 S @ not allowed as actual argument.
Bezeichner "@" ist als aktuelles Argument ungültig.

209 S @ not a function or subroutine name.
Bezeichner "@" ist kein Funktions- oder Unterprogrammname.

210 S @ twice declared as dummy argument.
Der Bezeichner "@" wurde mehrfach als formaler Parameter einer Anweisungsfunktion vereinbart.

211 S @ not allowed as dummy argument.
Bezeichner "@" darf nicht als formaler Parameter einer Anweisungsfunktion verwendet werden. Standarddatentyp wurde angenommen.

212 S @ neither array nor statement function.
Bezeichner "@" auf der linken Seite der Ergibtanweisung ist kein Feld oder Bezeichner kann kein Anweisungsfunktionsname sein.

213 E @ not of type INTEGER.
Bezeichner "@" hat nicht den Datentyp INTEGER.

214 S @ not allowed in label assignment.
Bezeichner "@" darf nicht als Markenvariable verwendet werden. Der Standarddatentyp wurden angenommen.

215 S Multiple declared label.
Die Marke wurde mehrfach vereinbart.

216 S Substring @ not of type CHARACTER.
Der als Teilkette verwendete Bezeichner "@" ist nicht vom Typ CHARACTER.

217 S @ in illegal context.
Bezeichner "@" im Kontext nicht erlaubt.

218 S @ in illegal context. May be consequence of error.
Bezeichner "@" im Kontext nicht erlaubt; möglicherweise Folgefehler.

222 E Control line ends statement. Control line ignored.
Innerhalb einer Anweisung tritt eine Steuerzeile auf. Die Anweisung wurde beendet und die Steuerzeile ignoriert.

240 S @ may not be dummy argument.
Formaler Parameter "@" wurde bereits als SUBROUTINEN-Name oder formaler Parameter verwendet.

241 S Illegal dummy argument ignored.
Ungültiger formaler Parameter in der Liste der formalen Parameter wurde ignoriert.

242 E Incorrect length specification.
Ungültige Längenangabe in FUNCTION wurde ignoriert.

243 S @ conflicts with use before.
Eintrittspunkt "@" steht im Widerspruch zur vorherigen Verwendung. Die Anweisung wurde ignoriert.

244 E Alternate return specifier near @ ignored.
Alternative Rücksprünge in FUNCTION bei ENTRY "@" wurden ignoriert.

245 S @ used before in COMMON, SAVE, or EQUIVALENCE.
Formaler Parameter "@" wurde bereits in COMMON, EQUIVALENCE oder SAVE verwendet.

246 S Implementation restriction. More than @ dummy arguments.
Implementierungsbedingte Einschränkung. Die Parameterliste enthält mehr als "@" formale Parameter. Überzählige Argumente wurden ignoriert.

247 E Illegal prefix ignored.
Unerlaubtes Vorzeichen in der Längenangabe wurde ignoriert.

248 E Duplicate use of @ in SAVE.
Zweimalige Verwendung des Elementes "@" in der SAVE-Anweisung. Das Element wurde ignoriert.

249 E @ is INTRINSIC. Dummy argument ignored.
Formaler Parameter "@" wurde bereits als INTRINSIC-Funktion definiert. Er wurde ignoriert.

254 S Missing (near @.
Fehlende "(" vor Dimensionsliste bei "@".

255 S More than 7 dimensions.
Mehr als 7 Dimensionen angegeben.

256 E Invalid delimiter. Comma assumed.
Ungültiges Trennzeichen. Ein Komma wurde angenommen.

260 E Zero length specification. 1 assumed.
Längenangabe hat den Wert 0 oder ist nicht vom Typ INTEGER. 1 wurde angenommen.

262 E @ not a parameter. 1 assumed.
"@" ist keine Konstante. 1 wurde angenommen.

263 S First * ends dimension list near @.
Der erste '*' beendet die Dimensionsliste von "@".

266 E Symbolic name expected.
Erwarteter symbolischer Name fehlt.

267 E Invalid length specification. 1 assumed.
Ungültige Längenangabe. 1 wurde angenommen.

268 E Invalid length specification. Default assumed.
Ungültige Längenangabe. Standardlänge wurde angenommen.

270 S @ impossible as assumed size array.
Variable Dimensionsgrenzen bei "@", aber gleichzeitig in SAVE oder COMMON.

272 S @ used before as global entity or in PARAMETER.
"@" ist bereits globale Größe oder symbolischer Name einer Konstanten.

273 S Invalid common name.
Fehlerhafter COMMON-Name.

275 S Assumed size array @ impossible in COMMON.
Feldgrenzen für "@" enthalten Variablen. Rest des COMMON-Bereichs wurde ignoriert.

276 S Type conflict near @.
COMMON-Element "@" widerspricht dem Datentyp der anderen COMMON-Elemente.

277 S Duplicate array declaration for @.
COMMON-Element "@" wurde bereits als Feld vereinbart.

278 S Common element expected or duplicate.
COMMOM-Element fehlt oder doppelt verwendet. Der Rest des COMMON-Bereiches wurde ignoriert.

279 S Type conflict in COMMON near @
In einer COMMON-Anweisung hat "@" einen unverträglichen Datentyp.

280 S @ in EQUIVALENCE with conflicting type.
"@" steht zu einer Variablen mit unverträglichen Datentyp in EQUIVALENCE.

282 E @ redundant in EQUIVALENCE.
Redundantes Auftreten von "@" in EQUIVALENCE.

283 S Illegal substring or index specification for @.
Fehler in INDEX- oder Teilkettenangabe für "@".

284 S More than one element expected.
EQUIVALENCE-Gruppe enthält nur ein Element.

285 W Last index too large for @.
In der Indexliste von @ ist der letzte Index zu groß.

286 S Index exceeds dimension bounds for @.
Index nicht in Dimensionsgrenzen bei "@".

287 S @ not an array but used so.
"@" hat Indizes, ist aber kein Feld.

288 S Number of indexes conflicts with dimensions for @.
Indexanzahl für "@" stimmt nicht mit Dimensionsanzahl überein. Die Indexliste wurde ignoriert.

289 T Conflicting equivalence near @.
Unverträgliche Äquivalenz bei "@".

290 W Common @ has no element.
COMMON-Bereich "@" hat kein Element.

291 T Common extented to left near @ effected by EQUIVALENCE.
COMMON-Bereich wurde bei "@" durch eine EQUIVALENCE nach links erweitert.

292 T Common changed near @ effected by EQUIVALENCE.
Abstand oder Reihenfolge der COMMON-Elemente wurde bei "@" durch EQUIVALENCE verändert.

293 S Invalid element in EQUIVALENCE.
Ungültiges EQUIVALENCE-Element.

295 E @ can not be intrinsic.
"@" ist keine INTRINSIC-Funktion oder durch andere Verwendung zu Nicht-INTRINSIC geworden.

296 S @ used as common name before.
Element "@" ist bereits ein COMMON-Name.

297 S Lower bound exceeds upper bound near @.
Untergrenze ist größer als Obergrenze im Felderklärer von "@".

300 E Text following an expression skipped. May be consequence of error.
Text zwischen Ausdruck und nächstem :,), Komma oder Anweisungsende wurde übergangen; oft Folgefehler eines Fehlers im Ausdruck.

301 S Symbolic name expected in input list.
Element in Eingabeliste ist kein Bezeichner.

302 S GT/GE/LT/LE-comparision with COMPLEX operand.
Ausdruck enthält GT/GE/LT/LE-Vergleich mit COMPLEX-Operanden.

303 S Expression type conflicts with operand type.
Operand passt nicht zum verlangten Ausdruckstyp.

304 S @ in constant expression misplaced.
Variable "@" tritt in einem Ausdruck auf, der konstant sein muß.

305 S Parameter @ misplaced in input list.
Konstante "@" tritt in Eingabeliste auf.

306 S Text before) in substring list ignored.
Text vor ')' in Teilkettenliste wurde übergangen.

307 S Text before : in substring list ignored.
Text vor ':' in Teilkettenliste wurde übergangen.

308 S List of actual arguments for @ ignored.
Parameterliste für Prozedur "@" muß leer sein. Die Parameter wurden ignoriert.

309 S List of actual arguments for @ missing.
Parameterliste für Prozedur "@" darf nicht leer sein.

310 S Argument of @ of wrong type.
Falscher Datentyp im aktuellen Parameter der INTRINSIC-Funktion "@".

311 S Misplaced .NOT. .
Operator .NOT. ist hier nicht erlaubt.

312 W Actual argument for @ with variable length.
Für '(CHARACTER-OPERAND)' als aktuellen Parameter der Prozedur "@" kann keine temporäre Variable erzeugt werden, da Länge unbekannt ist. Der Operand wurde selbst als aktueller Parameter übergeben.

313 S Actual argument contains // with variable length operand.
Für einen CHARACTER-Ausdruck (mit //-Operator(en)) als aktuellen Parameter kann keine korrekte temporäre Variable erzeugt werden, da *-Längen aufgetreten sind. Der Parameter wurde wahrscheinlich mit falscher Länge übergeben.

314 S Text before ',' skipped in list of actual arguments near @.
In der Liste der aktuellen Parameter von Prozedur "@" wurde Text zwischen Parameter und ',' übergangen.

315 S Text before ')' skipped in list of actual arguments near @.
Schließende Klammer fehlt oder folgt nicht sofort auf Liste der aktuellen Parameter der Prozedur "@".

316 S Missing actual arguments in call of @.
Zu wenig aktuelle Parameter im Aufruf der SUBROUTINE "@".

317 S Too many actual arguments in call of @.
Zu viele aktuelle Parameter im Aufruf der SUBROUTINE "@". Restliche Parameter wurden ignoriert.

318 S Redundant text in CALL statement.
Überflüssiger Text in CALL-Anweisung nach SUBROUTINEN-Aufruf.

319 S Wrong number of alternate returns.
Falsche Anzahl alternativer Ausgänge in der CALL-Anweisung.

320 E Wrong actual argument deleted.
Fehlerhafter aktueller Parameter wurde ignoriert.

321 E Invalid text ignored.
Nicht interpretierbarer Text in der Anweisung wurde übergangen.

322 E Length specification * incorrect. 1 assumed.
Längenangabe '*' ist nicht erlaubt. Länge 1 wurde angenommen.

323 E Duplicate use of letter.
Buchstabe darf nur einmmal in der IMPLICIT-Anweisung auftreten.

324 E Wrong expression type converted.
Ausdruck hat falschen Typ und wurde konvertiert.

325 S @ invalid destination operand.
Unzulässiger Zieloperand "@" in der Ergibtanweisung.

326 S Missing = after destination operand.
'=' fehlt oder folgt nicht sofort auf Ziel.

327 S Conflict between source and destination. Assignment impossible.
Typ von Quelle und Ziel in der Ergibtanweisung unverträglich; Zuweisung nicht möglich.

328 S Conversion to LOGICAL impossible.
Keine Konvertierung zum Typ LOGICAL möglich.

329 S Redundant text at end of statement ignored.
Überflüssiger Text am Anweisungsende wurde ignoriert.

330 S Empty argument in list of actual arguments.
'(,' oder ',,' oder ',)' in der Liste der aktuellen Parameter.

331 E Intervall specification invalid.
Buchstabenbereich nicht in alphabetischer Folge.

332 S Expression not of type LOGICAL.
Ausdruck hinter IF/ELSEIF ist nicht vom Typ LOGICAL.

333 S ENDIF inserted.
ENDIF wurde vor END-Anweisung eingefügt, da ENDIF fehlt oder in DO-Gruppe auftritt.

334 S Invalid length of substring @. 1 assumed.
Fehler in Teilkettengrenzen für Variable "@".

335 S Invalid reference of function @.
Unzulässiger Aufruf der CHARACTER-Funktion "@". Bei Aufruf darf die Länge des Funktionswertes nicht als *-Länge vereinbart sein.

336 S Expression of wrong type. Conversion impossible.
Ausdruck hat falschen Typ, Konvertierung nicht möglich.

337 W Critical conversion in assignment.
Bedenkliche Konvertierung bei Zuweisung.

338 E Integer constant too large in assignment.
INTEGER-Konstante zu groß bei Zuweisung.

402 S Unable to evaluate a constant expression.
Die Berechnung eines Konstantenausdrucks führt zu einer Unterbrechungsbedingung.

406 S Error in conversion of constant.
Eine Konstante läßt sich nicht in den erforderlichen Datentyp konvertieren.

407 E Name list contains illegal entity.
Unzulässiges Element in der Namensliste.

408 E Initialize of @ prohibited.
Initialisierung von "@" ist nicht erlaubt.

413 S Implementation restriction. Implied-DO nesting too deep.
Implementierungsbedingte Einschränkung. Implizite DO-Liste zu tief geschachtelt.

414 E Implied-DO list contains illegal entity.
Ungültiges Element in der DO-Liste.

416 S Implementation restriction. INTEGER constant exceeds 32767.
Implementierungsbedingte Einschränkung. Indexliste enthält Wert > 32767.

419 E Illegal entity in index list for @.
Indexliste von "@" enthält ungültiges Element.

420 E Illegal use of @ in name list .
"@" ist ungültiges Element der Namensliste.

422 E @ not a parameter or invalid.
"@" ist kein Konstantenname.

423 E Type of @ conflicts with type of initial value.
Datentyp von "@" ist nicht mit dem Typ des zugehörigen Anfangswertes verträglich.

424 E Repeat specification not of type INTEGER.
Wiederholungsfaktor ist keine INTEGER-Konstante.

426 E Data list ended before name list.
Datenliste abgearbeitet, Namenliste enthält noch Elemente.

427 E @ not of type INTEGER.
"@" in Index-/Laufvariablenausdruck ist keine INTEGER-Konstante.

428 E Data list missing. May be consequence of error.
Datenliste fehlt.

432 S Prefix operator conflicts with type of operand.
Präfixoperator widerspricht dem Datentyp des Operanden.

433 E Erroneous repeat specification.
Fehlerhafte Wiederholungsspezifikation.

434 E Illegal text after data list.
Ungültiger Text nach Datenliste.

435 E Data list without closing /.
Datenliste nicht durch "/" beendet.

437 S Invalid type of DO variable @.
Falscher Datentyp für Laufvariable "@" der DO-Anweisung, Anweisung wurde ignoriert.

439 S Expression starts with two or more unary operators.
Ausdruck beginnt mit zwei oder mehr unären Operatoren.

440 S End of DO group missing.
Offene DO-Gruppen am Ende der Programmeinheit, DO-Gruppen wurden vor der END-Anweisung beendet.

441 S DO group exceeds end of IF/ELSEIF block.
DO-Gruppe erstreckt sich über das Ende eines IF/ELSEIF-Blocks, DO-Gruppe wurde vor der ELSEIF-, ELSE- oder ENDIF-Anweisung beendet.

442 S Statement not permitted. Replaced by CONTINUE.
Anweisung ist in einer DO-Gruppe nicht erlaubt und wurde durch CONTINUE ersetzt.

443 S Overlapping DO group.
Verbotene Überlappung von DO-Gruppen, für den Abschluß der inneren DO-Gruppe wurde Code eingefügt.

444 S Erroneous end of DO group. Replaced by CONTINUE.
Die Anweisung ist nicht als Ende einer DO-Gruppe erlaubt und wurde durch CONTINUE ersetzt.

445 S Comma missing after expression.
',' folgt nicht oder nicht sofort auf Ausdruck in der DO-Anweisung, oft Folge eines Fehlers im Ausdruck.

447 S Incrementation expression erroneous. 1 assumed.
Ausdruck für Schrittweite in DO-Anweisung unvollständig, 1 wurde angenommen.

448 T Implementation restriction. DO nesting too deep.
Implementierungsbedingte Einschränkung. Zu tiefe Schachtelung von DO-Gruppen.

450 S Invalid label reference.
Marke in DO-Anweisung bezieht sich auf frühere Anweisung, DO-Anweisung wurde ignoriert.

452 S Text ignored between expression and label.
Überflüssiger Text zwischen arithmetischen Ausdruck und Marken wurde ignoriert, oft Folge von Fehler im Ausdruck

453 S Erroneous label @.
Fehler in Marke "@" der arithmetischen IF-Anweisung.

454 S Expression not of type LOGICAL. .TRUE. assumed.
Ausdruck in logischer IF-Anweisung ist kein logischer Ausdruck, .TRUE. wurde angenommen.

455 S Statement missing after expression.
Bedingte Anweisung fehlt in logischer IF-Anweisung, CONTINUE wurde angenommen.

456 S Text ignored between IF expression and statement.
Überflüssiger Text zwischen logischem Ausdruck und Schlüsselwort wurde ignoriert.

457 S Statement invalid after IF expression.
Anweisung in logischer IF-Anweisung ist dort verboten, nicht erkennbar oder falsch und wurde durch CONTINUE ersetzt.

459 S Erroneous label after GOTO.
Fehler in Marke der unbedingten GOTO-Anweisung nach logischen IF, GOTO wurde durch CONTINUE ersetzt.

460 E Incrementation value of 0 replaced by 1.
Schrittweite 0 wurde durch 1 ersetzt.

461 S Implementation restriction. Length exceeds 32767 near @.
"@" hat Länge größer 32767. 1 wurde angenommen.

462 S Implementation restriction. Offset exceeds 32767 near @.
"@" hat Index- oder Teilkettenangabe größer 32767.

463 S Implementation restriction. Equivalence offset exceeds 32767 near @.
EQU-Offset für "@" größer 32767.

470 S Incorrect use of label @.
Sprung zu MARKE "@" ist unzulässig. Einsprung in einen DO- oder IF-Block.

471 S Incorrect use of label @.
Marke "@" ist an einer FORMAT-Anweisung definiert. Verzweigung ist nicht möglich.

472 S Use of label @ in I/O list prohibited.
Marke "@" ist nicht an einer FORMAT-Anweisung definiert. Verwendung in E/A-Liste ist unzulässig.

473 S Incorrect use of label @.
Marke "@" ist an einer ELSEIF-, ELSE- oder ENDIF-Anweisung definiert.

474 S Incorrect use of label @.
Marke "@" ist an einer nicht ausführbaren Anweisung definiert.

475 S Undefined label @.
Marke "@" ist nicht vereinbart.

476 E Label @ shortened to 5 digits.
Die Marke "@" hat einen unerlaubten Wert. Die Marke wurde gekürzt.

477 T Implementation restriction. More than 99 statement functions.
Implementierungsbedingte Einschränkung. Die Programmeinheit enthält mehr als 99 Anweisungsfunktionen.

480 T Implementation restriction. Data segment overflow.
Gesamtlänge aller Variablen, Konstanten und Hilfsvariablen im DATA-Segment ist größer als 65535.

481 T Implementation restriction. COMMON @ overflow.
Implementierungsbedingte Einschränkung. Gesamtlänge aller Variablen im COMMON-Segment "@" größer als 65535.

482 T Implementation restriction.
More than 1000 external references.
Die Programmeinheit enthält mehr als 1000 externe Bezugnahmen.

483 T Unable to open @ file.
Die "@"-Datei kann nicht erfolgreich eröffnet werden.

484 T Implementation restriction.
Overflow of an internal statement table.
Die Anweisung enthält mehr als 300 Namen und Konstanten.

485 T Implementation restriction. Temporary storage overflow.
Implementierungsbedingte Einschränkung. Länge oder Anzahl der temporären Variablen ist zu groß.

486 T Implementation restriction. Operand stack overflow.
Implementierungsbedingte Einschränkung.
Operandenstapel-Überlauf. Anweisung ist zu kompliziert.

Anlage 3 Fehlercodes der Ausführungszeitfehler

Die Fehlercodes des Bereiches 0AH-300H werden erzeugt, wenn bei der internen Nutzung von Betriebssystemfunktionen ein Fehler angezeigt wurde.

Fehlercode hex	dezimal	Beschreibung des Fehlers
000A	0010	Ungültiger Funktionsaufruf
0014	0020	Datei existiert nicht
001E	0030	Pfad nicht gefunden
0028	0040	zu viele Dateien eröffnet
0032	0050	Zugriff auf die Datei beim Eröffnen abgelehnt
003C	0060	Fehler bei der Dateibehandlung
0046	0070	Speichersteuerblöcke zerstört
0050	0080	nicht genügend Speicherplatz
005A	0090	ungültige Speicherblockadresse
0078	0120	falscher Zugriffscode beim Eröffnen der Datei
0082	0130	falsche Daten
0200	0512	Datei wird vom System nicht unterstützt
0220	0544	Versuch, R/O-Datei neu anzulegen
0230	0560	ungültiger Modus bei Nutzung von Betriebssystemfunktionen
0240	0576	Versuch, nicht vorhandene Region freizugeben
0250	0592	Verbindung für nicht existierende Datei
0260	0608	Eingabebereich zu kurz
0270	0624	nicht genügend Platz auf Diskette
0280	0640	ungültiger Dateiname
0290	0656	Fehler beim Lesen
0300	0768	Fehler beim Schreiben
11xx		Die mit der UNIT=xx verbundene Datei kann nicht abgeschlossen werden.
1200	4608	ungültige STATUS-Angabe in OPEN-Anweisung
1201	4609	ungültige ACCESS-Angabe in OPEN-Anweisung
1202	4610	ungültige FORM-Angabe in OPEN-Anweisung
1203	4611	ungültige BLANK-Angabe in OPEN-Anweisung
1204	4612	ungültige STATUS-Angabe in OPEN-Anweisung
1210	4624	OPEN-Anweisung mit STATUS='NEW' für existierende Datei nicht erlaubt.
1211	4625	OPEN-Anweisung mit FILE-Angabe verlangt STATUS='NEW' oder STATUS='OLD'
1212	4626	STATUS='KEEP' für SCRATCH-Datei nicht erlaubt
1215	4629	STATUS='OLD' für nicht existierende Datei nicht erlaubt
1220	4640	ACCESS-Angabe in OPEN-Anweisung für die Datei nicht erlaubt
1221	4641	FORM-Angabe in OPEN-Anweisung widerspricht dem Dateiformat
1222	4642	Bei der Angbe FORM='UNFORMATTED' ist die Angabe BLANK nicht gestattet.
1230	4658	Für eine Direktzugriffsdatei muß in der OPEN-Anweisung die Angabe RECL= vorhanden sein.
1232	4660	Auf die Datei kann nur sequentiell zugegriffen werden.
1233	4661	Die RECL-Angabe steht im Widerspruch zur Satzlänge der Datei.
1234	4662	Die geforderte Satzlänge ist größer als 32767.
1240	4674	E/A-Fehler beim Eröffnen der Datei.

Fehlercode hex	dezimal	Beschreibung des Fehlers
1250	4690	BACKSPACE, ENDFILE oder REWIND ist für die Datei nicht erlaubt (kein sequentieller Zugriff oder Standard-E/A-Datei).
1251	4691	BACKSPACE für nicht formatierte Datei ohne RECL-Angabe ist nicht möglich.
1252	4692	Kein Rückwärtslesen der Datei bei REWIND oder BACKSPACE möglich.
1253	4693	BACKSPACE für nichtexistierende Datei nicht erlaubt.
1281	4737	Es soll eine E/A-Anweisung innerhalb einer E/A-Anweisung ausgeführt werden (z. B. E/A in Funktionsaufruf in Datenliste).
1282	4738	Imlementierungsbedingte Einschränkung: - UNIT-Nr. zu groß.
1283	4739	Implementierungsbedingte Einschränkung: - zu viele Dateien eröffnet.
1284	4740	Implementierungsbedingte Einschränkung: - zu tiefe Schachtelung der Formatliste.
1285	4741	Dateiende aufgetreten.
1286	4742	Versuch des Schreibens hinter ENDFILE-Satz
1287	4743	Keine UNIT-Angabe.
1289	4745	Keine Datei-Vorverbindung vorhanden.
1290	4752	Keine Dateiverbindung für Direktzugriffsdatei.
1291	4753	Unverträglichkeit zwischen E/A-Anweisung und Dateiverbindung.
1295	4757	Satznummer ist negativ.
1296	4758	Satz existiert nicht.
12A0	4768	Satzlänge ist kleiner als die Datenlänge.
12A1	4769	Implementierungsbedingte Einschränkung: Satzlänge wird zu groß.
12A2	4770	Datenlänge überschreitet Satzende.
12A5	4771	Es sind keine weiteren Sätze einer internen Datei vorhanden.
12B0	4784	Falscher Datentyp übergeben; ggf. Speicherbereiche überschrieben.
12B1	4785	Datenlänge überschreitet Satzende.
12B2	4786	Satz- und Dateilänge unverträglich.
12C0	4800	Zum Datentyp nicht passende Datendarstellung.
12C1	4801	Zu Formatelement Iw nicht pass. Datenelement.
12C2	4802	Zu Formatelement Iwm nicht pass. Datenelement.
12C3	4803	Zu Formatelement Fwd nicht pass. Datenelement.
12C4	4804	Zu Formatelement Ewd nicht pass. Datenelement.
12C5	4805	Zu Formatelement EwdEe nicht pass. Datenelement.
12C6	4806	Zu Formatelement Dwd nicht pass. Datenelement.
12C7	4807	Zu Formatelement Gwd nicht pass. Datenelement.
12C8	4808	Zu Formatelement GwdEe nicht pass. Datenelement.
12C9	4809	Zu Formatelement Lw nicht pass. Datenelement.
12CA	4810	Zu Formatelement A nicht pass. Datenelement.
12CB	4811	Zu Formatelement Aw nicht pass. Datenelement.
12E1	4833	Formatliste enthält kein wiederholbares Formatelement.
12E2	4834	Zeichenketten-Formatliste ist falsch.
12E3	4835	Formatelement falsch; ggf. Programm zerstört.
12E6	4838	Formatangabe als Zeichenkette zu lang.
12E8	4840	Falsches Zeichen im Eingabebereich.

Fehlercode hex	dezimal	Beschreibung des Fehlers
12E9	4841	Gleitkomma-Überlauf bei Konvertierung.
12EA	4842	Eingabe: Zahl zu groß.
12EB	4843	Wiederholungsfaktor für listgesteuerte Eingabe zu groß.
12FF	4863	Fehler bei der E/A-Initialisierung.
1501	5377	Syntaktische Fehler bei der Dateivorverbindungs-Angabe in der Kommandozeile oder der UNIT-Datei.
1601	5633	Kein ASSIGN für Markenvariable vor dem GOTO.
1602	5634	Marke nicht in der Markenliste des GOTO.
8020	32800	Division einer komplexen Zahl mit (0.0).
8021	32801	CLOG(0.0)
8022	32802	(0,0)**i i . LT. 0
8023	32803	(0,0)**c und REAL(c).LE.0 oder IMAG(c).NE.0
9000	36864	Es wurde eine Gleitkomma-Operation ausgeführt, zum Programm ist jedoch eine falsche Gleitkomma-bibliothek gelinkt worden.

Anlage 4 PUBLIC-Namen des FOR77-Laufzeitsystems

Bestandsname	PUBLIC-Name	PUBLIC-Name
F77BAS.LIB	CONTROL_CHARACTER_CON	CONTROL_CHARACTER_LST
	CONTROL_DATA	$$OVLINIT
	F77_INI	
	F77_PC	FI_2ABS
	FI_2CHAR	FI_2DIM
	FI_ABS	FI_2SIGN
	FI_AIMAG	FI_ACOS
	FI_ALOG	FI_AINT
	FI_AMOD	FI_ALOG1
	FI_ASIN	FI_ANINT
	FI_ATAN2	FI_ATAN
	FI_CHAR	FI_CABS
	FI_CON2C	FI_CON
	FI_CONIC	FI_CONDC
	FI_CONSC	FI_CONJG
	FI_COSH	FI_COS
	FI_DACOS	FI_DABS
	FI_DATA2	FI_DASIN
	FI_DCOS	FI_DATAN
	FI_DDIM	FI_DCOSH
	FI_DIM	FI_DEXP
	FI_DLOG	FI_DINT
	FI_DMOD	FI_DLOG1
	FI_DPROD	FI_DNINT
	FI_DSIN	FI_DSIGN
	FI_DSQRT	FI_DSINH
	FI_DTANH	FI_DTAN
	FI_IABS	FI_EXP
	FI_IDNIN	FI_IDIM
	FI_INTER	FI_INDEX
	FI_LEN	FI_ISIGN
	FI_LGT	FI_LGE
	FI_LLT	FI_LLE
	FI_NINT	FI_SIGN
	FI_SIN	FI_SINH
	FI_SQRT	FI_TAN
	FI_TANH	FQ_861
	FQ_860	FQ_915
	FQ_911	FQ_CFBL2
	FQ_CFBL1	FQ_CFBL4
	FQ_CFBL3	FQ_CFBL6
	FQ_CFBL5	FQ_CHECK
	FQ_CFBL7	FQ_CHOMI2
	FQ_COM2I	FQ_EXPR2
	FQ_COMII	FQ_EXPRR
	FQ_EXPRI	FQ_FSC
	FQ_EXT	FQ_GO_AS
	FQ_GO_AL	FQ_IND2
	FQ_HP	FQ_INT
	FQ_IND4	FQ_PGM2
	FQ_ROUND	FQ_STOP
	F_CHCOMP	F_MOVEC

Bestandsname	PUBLIC-Name	PUBLIC-Name
	F_MOVEL	HIGH_
	MQERACS	MQERAIN
	MQERANT	MQERASN
	MQERAT2	MQERATC
	MQERATN	MQERCI2
	MQERCOS	MQERCSH
	MQERDIM	MQEREXP
	MQERIA2	MQERIA4
	MQERIAX	MQERIC2
	MQERIC4	MQERICX
	MQERIE2	MQERIE4
	MQERIEX	MQERINT
	MQERIRT	MQERLGD
	MQERLGE	MQERMAX
	MQERMIN	MQERMOD
	MQERNI2	MQERNIN
	MQERRI2	MQERRMD
	MQERRNT	MQERSGN
	MQERSIN	MQERSNH
	MQERTAN	MQERTNH
	MQERY2X	MQERYI2
	MQERY14	MQERYIS
	MQ_1	MQ_2XM1
	MQ_63U	MQ_63U1
	MQ_63UPI2	MQ_AT2
	MQ_CCOS	MQ_CEXP
	MQ_CLOG	MQ_CONST
	MQ_COS	MQ_CP2N63
	MQ_CSIN	MQ_CSQRT
	MQ_DECIDE	MQ_EXIT
	MQ_EXM1	MQ_I
	MQ_IRCHK	MQ_LOG
	MQ_LOG10	MQ_LOGDN
	MQ_MAXD	MQ_MAXI
	MQ_MAXL	MQ_MAXLR
	MQ_MAXR	MQ_MAXRL
	MQ_MIND	MQ_MINI
	MQ_MINL	MQ_MINLR
	MQ_MINR	MQ_MINRL
	MQ_MQRPI	MQ_NAN
	MQ_NOF	MQ_NORM
	MQ_NQ	MQ_OF
	MQ_PO	MQ_PI2
	MQ_PII	MQ_Q
	MQ_RAD	MQ_RERR
	MQ_SIN	MQ_SQRT
	MQ_TLOG	MQ_TXAM
	MQ_UO	MQ_YL2X
	NMI_A	PGM_ENDE
	PGM_ENDE1	PSP_B
	PSP_E	
	TAB_1	TAB_10
	TAB_11	TAB_12
	TAB_13	TAB_14
	TAB_2	TAB_3
	TAB_4	TAB_5
	TAB_6	TAB_7
	TAB_8	TAB_9
	TQESTART	TQGETERH

Bestandsname	PUBLIC-Name	PUBLIC-Name
	TQWHERESTRAP87	
	TQ_102	
	TQ_150	TQ_152
	TQ_160	TQ_170
	TQ_171	TQ_180
	TQ_181	TQ_190
	TQ_191	TQ_200
	TQ_210	TQ_CC
	TQ_CI2	
	TQ_DIVC	TQ_FLPERH
	TQ_HEXOUT	TQ_IN_8087_ERH
	TQ_LRSERH	TQ_MSGOUT
	TQ_MULC	TQ_PUSHSTR
	TQ_SYSERH	
	UQ_100	
F77EAB.OBJ	FEA_CONV	FEAROUTI
	FEAROUTX	FEAROUTY
	FEAROUTZ	ZK_VGL
	IF_ALLOCATE	IF_ALLOC_INI
	IF_ALLOC_PARA	IF_ATTACH
	IF_CLOSE	IF_CREATE
	IF_DELETE	IF_DETACH
	IF_FREE	IF_GCS1
	IF_GETCONNECTIONSTATUS	IF_OPEN
	IF_READ	IF_SEEK
	IF_TRUNCATE	IF_WRITE
F77EAR.OBJ/ F77EAS.OBJ/ F77EAN.OBJ	FEAROUT0	FEAROUT1
	FQ_CLOSE	
F77P87.LIB	FIARQQ	FICRQQ
	FIDRQQ	FIERQQ
	FISRQQ	FIWRQQ
	FI_MOD	FI_2MOD
	FJARQQ	FJCRQQ
	FJSRQQ	FQMD
	FQ_DIVI	FQ_EXPII
	FQ_MULI	INITFP
	M:_NCS	M:_NDS
	M:_NES	M:_NSS
	M:_NST	M:_WCS
	M:_WDS	M:_WES
	M:_WSS	M:_WST
	M:_WT	TQ_CI4
	TQ_100	TQ_110
	TQ_120	TQ_140
	TQ_TRAP87	

Bestandsname	PUBLIC-Name	PUBLIC-Name
F77E87.LIB	FIARQQ	FICRQQ
	FIDRQQ	FIERQQ
	FISRQQ	FIWRQQ
	FI_MOD	FI_2MOD
	FJARQQ	
	FJCRQQ	FJSRQQ
	FQMD	FQ_DIVI
	FQ_EXPII	FQ_MULI
	INITFP	FJARQQ
	INT20	INT21
	INT22	INT23
	INT24	INT25
	INT26	INT27
	INT28	INT29
	INT30	INT31
	M:_NCS	M:_NDS
	M:_NES	M:_NSS
	M:_NST	M:_WCS
	M:_WDS	M:_WES
	M:_WSS	M:_WST
	M:_WT	TQ_CI4
	TQ_100	TQ_110
	TQ_120	TQ_140
	TQ_TRAP87	
F77N87.LIB	FIARQQ	FICRQQ
	FIDRQQ	FIERQQ
	FISRQQ	FIWRQQ
	FI_MOD	FI_2MOD
	FJARQQ	FJCRQQ
	FJSRQQ	FQMD
	FQ_DIVI	FQ_EXPII
	FQ_MULI	INITFP
	M:_NCS	M:_NDS
	M:=NES	M:_NSS
	M:_NST	M:_WCS
	M:_WDS	M:_WES
	M:_WSS	M:_WST
	M:_WT	TQ_CI4
	TQ_100	TQ_110
	TQ_120	TQ_140

Anlage 5 Implementierungsbedingte Besonderheiten

- FOR77 verwendet den ASCII-Code.
- Felder können maximal einen Speicherplatz von 32767 Bytes belegen.
- Markenlisten in der GOTO-Anweisung können maximal 50 Marken enthalten.
- Die maximale Schachtelungstiefe von DO-Anweisungen ist 25.
- Die Länge eines COMMON-Bereiches darf 65535 nicht übersteigen.
- Die Summe der Längen aller Elemente im DATA-Segment darf 65535 nicht übersteigen.
- Eine Programmeinheit darf nicht mehr als 99 Anweisungsfunktionen enthalten.
- Die Ausführung einer PAUSE-Anweisung führt zur Terminal-Meldung

  PROGRAM PAUSE 'text'

  Die Programmausführung wird erst fortgesetzt, wenn der Anwender eine beliebige Taste betätigt. Die Eingabe von CTRL-C oder "S" bewirkt die Beendigung des Programms mit implizitem Abschließen aller Dateien.
- Die Ausführung einer STOP-Anweisung führt nach der Terminal-Meldung

  PROGRAM STOP 'text'

  zur Programmbeendigung.
- Bei der Überlagerung mit EQUIVALENCE-Anweisungen dürfen keine Offsets größer 32767 entstehen.
- Zeichenkettenvariablen dürfen nicht länger als 32767 sein.
- Die Gesamtlänge des CODE-Segmentes (ausführbare Anweisungen einer Programmeinheit) ist auf 65535 beschränkt.
- Zur Ausführungszeit können standardmäßig 16 Dateien gleichzeitig eröffnet sein (siehe Abschnitt 7.1.).
- Bei Überlagerungen mit EQUIVALENCE-Anweisungen in einem COMMON-Bereich muß die Anfangsadresse des überlagerten COMMON-Elementes in den ersten 32K des COMMON-Bereiches liegen.
- Die Anzahl der formalen Parameter in einer ENTRY-Anweisung ist auf 30 beschränkt.
- In einer Anweisung dürfen maximal 300 Namen und Konstanten vorhanden sein.
- Eine Programmeinheit darf maximal 1000 externe Bezugnahmen enthalten.

Anlage 6 Interruptvektoren, die vom FOR77-Laufzeitsystem belegt werden

Interrupt hex.	dez.	Funktion	
0	0	INTEGER-Division durch Null	
2	2	Gleitkommafehlerbehandlung(Prozessor)	
4	4	INTEGER-]berlauf	(*)
82-8D	130-141	Gleitkommaemulation	
8E	142	Index-]berlauf	(*)
8F	143	Werte- oder Bereichs-]berl{ufe	(*)
90-93	144-147	E/A-Funktionen	
94	148	Gleitkommafehlerbehandlung(Emulator)	
95-A0	149-160	reserviert f}r Weiterentwicklung	

*) werden durch die Angabe der VALUE_CONTROL-Option aktiviert.

Literaturverzeichnis

[1] C 1016-0200 FORTRAN77
Sprachbeschreibung

[2] C 3015-0000 Anleitung für den Assemblerprogrammierer

[3] C 3013-0000 Anleitung für den Bediener, Software

Sachwortverzeichnis