KfK 2878
KfK-CAD 146
März 1980

GIPSY - Handbuch

G. Enderle, K. H. Bechler, F. Katz, K. Leinemann,
W. Olbrich, E. G. Schlechtendahl, K. Stölting
Institut für Reaktorentwicklung
Projekt Rechnerunterstütztes Entwickeln,
Konstruieren und Fertigen

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE
Institut für Reaktorentwicklung
Projekt Rechnerunterstütztes Entwickeln,
Konstruieren und Fertigen

KfK 2878
KfK-CAD 146

GIPSY-Handbuch

G. Enderle
K.H. Bechler
F. Katz
K. Leinemann
W. Olbrich
E.G. Schlechtendahl
K. Stölting

Kernforschungszentrum Karlsruhe G.m.b.H., Karlsruhe

Kernforschungszentrum Karlsruhe GmbH

Zusammenfassung

Das GIPSY-Handbuch enthält alle notwendigen Informationen, die zur Anwendung des graphischen Systems GIPSY erforderlich sind. Es umfaßt eine vollständige Beschreibung aller dem Anwender zur Verfügung stehenden Fähigkeiten des Systems mit Beispielen und Fehlermeldungen.

GIPSY (Graphical Information Processing System) bietet die Möglichkeit, zweidimensionale Liniengraphik und dreidimensionale Körpergraphik durch eine auf PL/1 basierende Sprache in benutzerfreundlicher Weise zu beschreiben.

GIPSY-User Manual

Abstract

The GIPSY-Manual contains all informations that are necessary for the application of the graphics system GIPSY. It gives a complete description of all user accessible system functions, examples and error messages.

GIPSY (Graphical Information Processing System) offers capabilities for two-dimensional line graphics and three-dimensional space graphics on the base of a user-oriented language extension to PL/1.

Dieses Handbuch und die darin beschriebene Version von GIPSY sind aus dem ursprünglich von Dr. R. Schuster entwickelten REGENT-Subsystem GIPSY hervorgegangen. Allen GIPSY-Anwendern innerhalb und außerhalb der KfK, die durch die Anwendung von GIPSY und konstruktive Kommentare zur Weiterentwicklung des Systems beigetragen haben, sei gedankt. Besonderer Dank gebührt Frau A. Brunner für das sorgfältige Schreiben des Manuskriptes und die Geduld, die sie angesichts der vielen Änderungen aufbrachte.

I n h a l t

1. Übersicht

Das Subsystem GIPSY (Graphical Information Processing System) stellt im Rahmen des Systems für das rechnergestützte Entwickeln und Konstruieren REGENT graphische Fähigkeiten für die Behandlung zwei- und dreidimensionaler Objekte bereit. Die Formulierung der graphischen Aufgaben ist in einer um graphische Anweisungen und Datentypen erweiterten, auf PL/1 basierenden Sprache möglich. GIPSY kann für die verschiedensten graphischen Aufgaben eingesetzt werden: für technische Zeichnungen, für Computerkunst, für dreidimensionale Architekturentwürfe, zur Darstellung von Wertefeldern und vieles mehr.
Je nach Anwendungsgebiet werden sich diese oder jene GIPSY-Fähigkeiten als mehr oder weniger wertvoll erweisen. Eine Gliederung der GIPSY-Anwendungen muß daher stets ein Maß an Willkür aufweisen. Dennoch erweist sich aufgrund der bisherigen Erfahrungen in Einsatz von GIPSY folgende Unterteilung als zweckmäßig:

- Anwendungen mit vorwiegend zweidimensionaler Liniengraphik.
  Hier geht es um graphische Darstellungen, die sich dem Anwender überwiegend aus geraden oder gekrümmten Linien zusammengesetzt darstellen, evtl. mit Textinformation versehen. Technische Zeichnungen gehören in diesen Bereich ebenso wie der Aufbau von Menüfeldern auf einem Bildschirm.
- Anwendungen zur Darstellung von Wertefeldern.
  Hierbei handelt es sich sowohl um zweidimensionale wie auch um dreidimensionale Probleme. Im zweidimensionalen Fall wird eine Größe als Funktion einer anderen Größe in einem DIAGRAMM dargestellt; im dreidimensionalen Fall kann eine Größe als Funktion von zwei anderen Größen entweder als HÖHENLINIEN oder als RELIEF wiedergegeben werden. Im allgemeinen Fall wird der Anwender die Darstellung durch zweidimensionale Liniengraphik (Linien und Texte) ergänzen wollen.

- Anwendungen zur Darstellung dreidimensionaler Körper. Hier muß eingeschränkt werden, daß die dreidimensionalen Körper nicht durch beliebige Flächen begrenzt sein dürfen, sondern durch ebene, kreiszylindrische, kreiskegelförmige und kugelige Flächenstücke. Allerdings kann ein Körper beliebig aus derart geformten Teilkörpern zusammengesetzt sein. Selbstverständlich kann der Anwender mit GIPSY nicht nur Körper im dreidimensionalen Raum bearbeiten, sondern auch dreidimensionale Linien beschreiben,d.h. Linien und Texte im Raum definieren und sie in die Zeichenebene abbilden. Auch bei diesen Anwendungen wird der Anwender meist die Darstellung der dreidimensionalen Körper, Linien und Texte (d. h. ihre Projektion in die Zeichenebene) durch zweidimensionale Linien ergänzen wollen.

Der GIPSY-Anwender wird daher zweckmäßigerweise sich zunächst mit den zweidimensionalen Linien vertraut machen. Dazu ist es erforderlich, sich nicht nur mit den rein graphischen Fähigkeiten von GIPSY (den graphischen Objekten und graphischen Operationen) zu befassen. Es ist auch die Kenntnis einiger Verwaltungs- und Steuerungsfunktionen von GIPSY notwendig. Erst auf dieser Grundkenntnis aufbauend, sollte der Anwender sich mit dreidimensionalen Aufgaben befassen. Die Erfahrung hat gezeigt, daß die Behandlung dreidimensionaler Probleme oft Schwierigkeiten bereitet, die darauf zurückzuführen sind, daß das dreidimensionale Vorstellungsvermögen beim Anwender nicht ausreichend trainiert ist.

Das vorliegende Handbuch ist gültig für GIPSY-Release 3.1. Die Unterschiede gegenüber dem in [1] beschriebenen GIPSY-Release 1 sind zum Teil wesentliche funktionelle Änderungen sowie Verbesserungen und Korrekturen.

2. Einführung

Diese Einführung ist nicht als Lehrbuch aufzufassen. Sie soll lediglich anhand von Beispielen dem Anfänger den Einstieg in GIPSY erleichtern. Vollständigkeit und Details sind den folgenden Kapiteln vorbehalten.

2.1 Das GIPSY-Konzept

GIPSY ist eine Programmiersprache.

Diesen Sachverhalt muß sich der Anwender klar machen, bevor er daran geht, sein Problem für die Bearbeitung mit GIPSY aufzubereiten. Es gibt in sich geschlossene graphische Systeme, die es dem Anwender erlauben, am Bildschirm mit Hilfe eines Lichtgriffels und anderer Techniken (z. B. Menü) graphische Darstellungen zu erzeugen. Andere Systeme verwenden statt der Lichtgriffel bzw. Menütechnik eine alphanumerische Tastatur, über die Kommandos eingegeben werden können. In jedem Fall aber wird die Reaktion des graphischen Systems auf die Anweisung des Anwenders sofort ausgeführt. Für bestimmte Anwendungen ist eine solche Betriebsweise auch durchaus angebracht. GIPSY selbst ist kein solches graphisches System. Wohl aber lassen sich mit GIPSY derartige interaktive graphische Systeme erstellen (siehe hierzu Kap. 7.5). GIPSY dagegen ist eine Programmiersprache für graphische Aufgaben in derselben Art wie FORTRAN eine Programmiersprache für Algebra ist. Das bedeutet, daß die Lösung einer graphischen Aufgabe stets die Planung und Erstellung eines Programms voraussetzt, das zur Lösung dieser und verwandter Aufgaben geeignet ist. Es gibt andere graphische Programmiersprachen. Der besondere Aspekt bei GIPSY ist folgender:

- GIPSY basiert auf PL/1 und enthält PL/1. Damit sind alle Möglichkeiten der Programmiersprache PL/1 auch dem GIPSY-Anwender zugänglich. Dies ist deswegen wichtig, da in der Praxis (außer bei Computerkunst) graphische Aufgaben nie alleine auftreten, sondern stets verknüpft sind mit Berechnungs- oder Verwaltungsaufgaben (z. B. festigkeitsmäßige Dimensionierung von Bauteilen oder Aufsuchen von Normgrößen in Tabellen).

- GIPSY ist ein REGENT-Subsystem. Daraus folgt zweierlei. Zunächst stehen dem Anwender die spezifischen Fähigkeiten des REGENT-Systems /2/, die über PL/1 hinausgehen, zur Verfügung. Darüber hinaus aber kann nun die Bearbeitung graphischer Aufgaben als Teilaufgabe in den größeren Rahmen eines umfassenderen Problems eingebettet werden, indem GIPSY-Programme zu Unterprogrammen anderer Subsysteme gemacht werden (siehe Kap. 7.1 und 7.4).

Dieses Handbuch richtet sich vor allem an denjenigen GIPSY-Anwender, der mit Hilfe der GIPSY-Programmiersprache (einschließlich PL/1) in REGENT GIPSY Programme erstellt ("GIPSY-Programmierer"). Derjenige, der lediglich bestehende GIPSY-Programme anwendet, sollte sich jedoch ebenfalls mit Kap. 1 bis 3, Kap. 7.5 und Kap. 10 vertraut machen.

2.2 Voraussetzungen

Es versteht sich von selbst, daß die Anwendung von GIPSY eine Rechenanlage voraussetzt, die in ihrer hardware- und softwaremäßigen Ausstattung bestimmte Voraussetzungen erfüllen muß (siehe Kap. 7.5). Grob gesprochen wird für GIPSY Release 3.1 eine IBM-Anlage vom Typ /360 oder aufwärtskompatibel mit mehr als 256 k Bytes Arbeitsspeicher und einer Peripherieausstattung, wie sie bei Rechnern dieser Klasse üblich ist, benötigt. Als Betriebssystem kann MVT, MVS oder VS/1 mit oder ohne TSO vorliegen. Notwendig ist, daß PL/1 unterstützt wird und daß das System REGENT auf der Anlage implementiert ist. Damit sind die Voraussetzungen für den Einsatz von GIPSY im allgemeinen gegeben.
Jede einzelne GIPSY-Anwendung erfordert wie jede Programmanwendung überhaupt, daß durch Steuerkarten der Job-Control-Sprache im Stapelbetrieb bzw. durch entsprechende Kommandos im Dialogbetrieb das Programm, welches eine GIPSY-Anwendung enthält, gestartet wird und die nötigen Dateien bereitgestellt werden. Auf diese (teilweise installationsspezifischen) Voraussetzungen soll an dieser Stelle nicht eingegangen werden (siehe Kap. 7.5).

2.3 GIPSY als REGENT-Subsystem

Eine GIPSY-Anwendung ist stets eingebettet in ein REGENT-Programm. Anfang und Ende eines solchen REGENT-Programmes haben stets (etwa) folgendes Aussehen:

```
PROGRAM: PROC OPTIONS(MAIN)
         REGENT(PLOT=STATOS, andere REGENT-Optionen);

         Hauptteil des Hauptprogramms
FINISH;
END PROGRAM;
```

Die erste und letzte Zeile ist Bestandteil überhaupt jeden PL/1-Programms. Die zweite Zeile gibt (unter anderem) an, auf welches Zeichengerät die Ausgabe erfolgen soll (hier auf STATOS). Diese Option kann von Installation zu Installation verschieden lauten. Unabdingbar aber ist der FINISH-Befehl, der kurz vor dem Programmende und nur hier stehen muß. Fehlt der FINISH-Befehl, so kann es sein, daß der Anwender keine Zeichnung erhält, obwohl das GIPSY-Programm sonst fehlerfrei arbeitete. Er würde nicht einmal eine Fehlermeldung erhalten.

2.4 Ein einfaches Beispiel

Aufgabenstellung:
Die Aufgabe lautet:
"Zeichne ein Sechseckmuster etwa so:

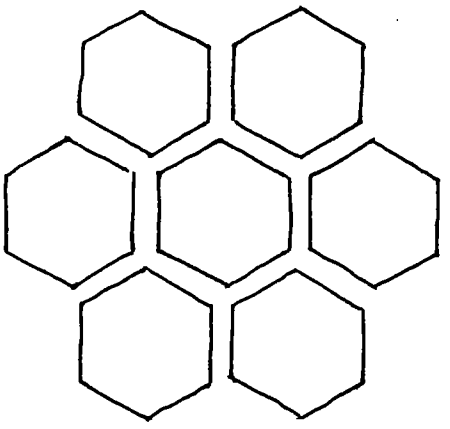


Abb. 2.1:
Handskizze

Schon diese einfache Aufgabe zeigt ein charakteristisches Merkmal des Arbeitens mit GIPSY:

Man beginne mit einer Handskizze

Die Bedeutung dieser Vorgehensweise wird klar, wenn man versucht, die obige Skizze in Worten zu beschreiben. Das Ergebnis wäre umfangreich und unanschaulich.

Aufgabenvorbereitung:

Die der Aufgabenstellung beigefügte Skizze reicht allerdings noch nicht aus, die Aufgabe nun mit GIPSY zu bearbeiten. Vielmehr wird man die Skizze so ergänzen, daß sie die einzelnen Arbeitsschritte die durchzuführen sind, einigermaßen vollständig wiedergibt. Beispielsweise fehlen folgende Angaben:

a) Welches Zeichenpapier soll gewählt werden?
b) Wie groß soll das Zeichenpapier sein?
c) Wie soll die Zeichnung darauf angeordnet sein?
d) Wie sollen die Striche dargestellt sein (Farbe, Strichstärke)?
e) Soll noch Zusatzinformation (Symbole, Texte) auf das Papier?

Die Punkte a und d werden in der vorliegenden GIPSY-Implementierung durch Wahl des Zeichengerätes bereits weitgehend entschieden. Die Punkte b, c und e jedoch werden in dem zu erstellenden GIPSY-Programm festgelegt. Sie bestimmen letztlich den optischen Eindruck der Zeichnung, d. h. ob sie "schön" ist. Dies läßt sich letztlich erst entscheiden, wenn die Zeichnung fertig vorliegt. Daher muß man sich von vornherein darauf einstellen:

Die Gestaltung einer neuen Zeichnung mit GIPSY ist ein Iterationsprozeß.

Dies gilt auch dann, wenn schon die erste erstellte Zeichnung im Sinne der Aufgabenstellung richtig ist. Sie wird aber im allgemeinen noch nicht "schön" sein.
Um diesen Iterationsprozeß abzukürzen, ist eine Aufgabenvorbereitung notwendig, und zwar in Form einer

Erstellung einer möglichst maßstabsgetreuen Skizze der Gesamtzeichnung.

In diese Skizze werden einige

<u>für die Gestaltung wichtige Maße eingetragen.</u>

Wir wollen nun die obige Aufgabenstellung in dieser Weise aufbereiten

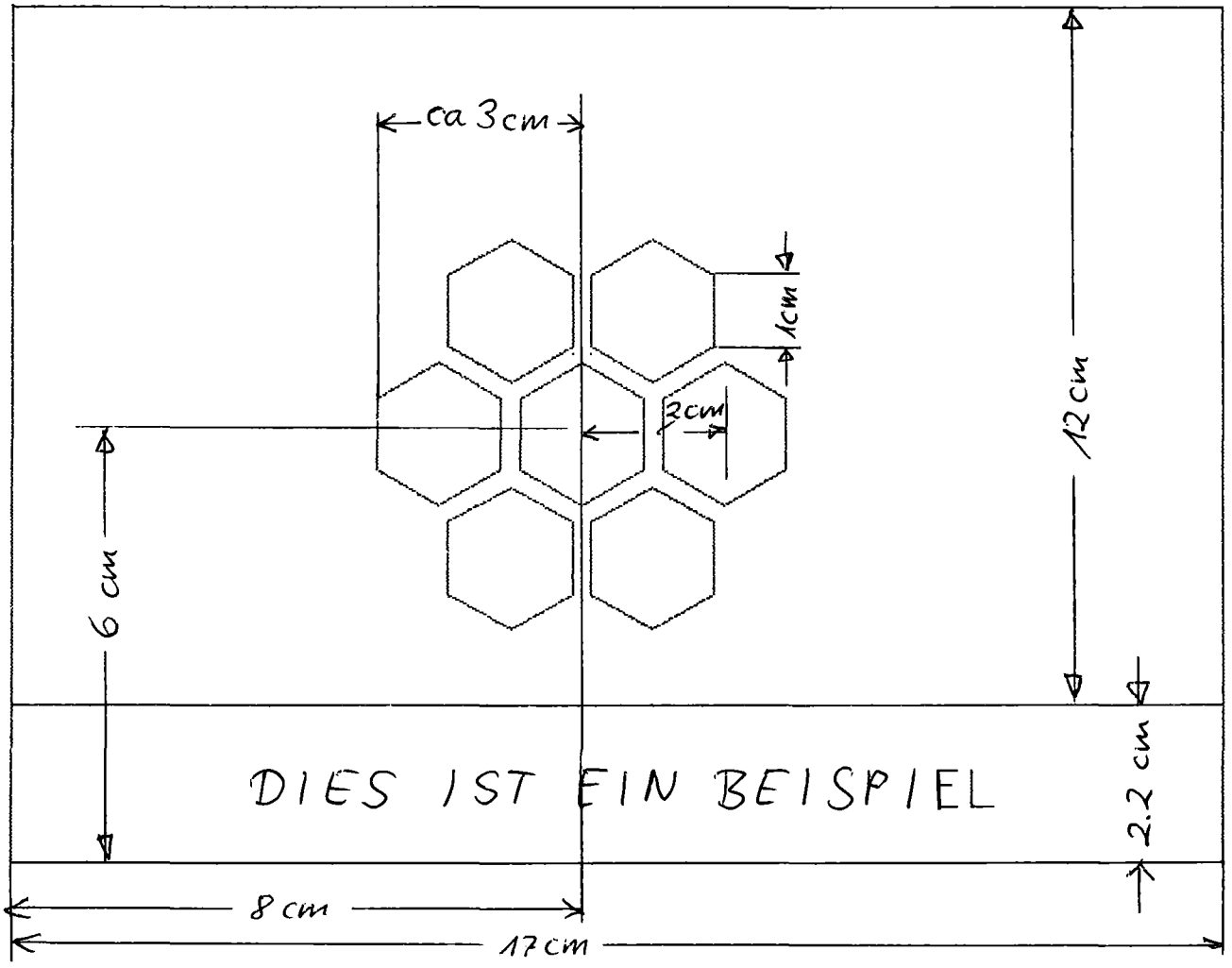


Abb. 2.2: Skizze mit Vermaßung

Diese Skizze ist zwar immer noch nicht vollständig, aber als Ausgangsbasis ausreichend. Mit ihrer Hilfe kann ein Programm wie das folgende aufgebaut werden. (Es gibt natürlich viele Möglichkeiten, das Programm zu gestalten, so daß es das gewünschte Bild erzeugt. Das folgende Programm ist mehr auf Anschaulichkeit als auf Effektivität orientiert!)

```
//IRE0251 JOB (0025,330,P0H1A),SCHLECHTENDAHL,REGION=512K,                00000010
//  NOTIFY=IRE025,MSGLEVEL=(0,0),MSGCLASS=A                               00000020
// EXEC REGENT,PVOL=PNS001,PLIB='TSO253.TIPSY',                           00000030
//   CPARM='OPT(2),S,INC',GPARM='/ISA(100K),REPORT'                       00000040
//P.SYSPRINT DD SYSOUT=A                                                  00000050
//P.SYSIN DD *                                                            00000060
 PROGRAM:PROC OPTIONS(MAIN)                                              00000070
      REGENT(PLOT=CALCOMP,MPOOL=80000);                                  00000080
   ENTER GIPSY;                                                          00000090
       DCL PI_DRITTEL INIT((ACOS(-1)/3));                                00000100
       DCL RADIUS INIT(0.01);                                            00000110
       DCL ABSTAND INIT(0.02);                                           00000120
       DCL 1 SECHS_PUNKTE(6),2(X,Y);                                     00000130
       DCL HEXAGON POLY2(6);                                             00000140
       DCL PUNKT POINT2;                                                 00000150
       DCL SECHSECK COLLECTION;                                          00000160
      CHANGE UNIT LENGTH M;                                              00000170
      OPEN PLOT SIZE(0.17,0.12);                                         00000180
  DO N=1 TO 6;                                                           00000190
  SECHS_PUNKTE(N).X=RADIUS*COS(N*PI_DRITTEL);                            00000200
  SECHS_PUNKTE(N).Y=RADIUS*SIN(N*PI_DRITTEL);                            00000210
  END;                                                                   00000220
  DO N=1 TO 6;                                                           00000230
      SET SECHSECK=  COLLECTION (SECHSECK,                               00000240
         POINT2(SECHS_PUNKTE(N).X,SECHS_PUNKTE(N).Y)  );                 00000250
  END;                                                                   00000260
      SET HEXAGON=POLY2(SECHSECK);                                       00000270
      EDIT CLOSED FOR(HEXAGON);                                          00000280
      SET HEXAGON=ROTATE2(HEXAGON,PI_DRITTEL*0.5 RAD);                   00000290
      SET SECHSECK=COLLECTION(HEXAGON);                                  00000300
  DO N=1 TO 6;                                                           00000310
      SET SECHSECK=COLLECTION(SECHSECK,                                  00000320
            SHIFT2(HEXAGON,                                              00000330
               ABSTAND*COS(N*PI_DRITTEL),                                00000340
               ABSTAND*SIN(N*PI_DRITTEL)   )  );                         00000350
  END;                                                                   00000360
       SET SECHSECK=SHIFT2(SECHSECK,0.08,0.06);                          00000370
      PLOT (SECHSECK);                                                   00000380
      EMPTY(SECHSECK);                                                   00000390
      CHANGE UNIT LENGTH MM;                                             00000400
    PLOT(TEXT2(POINT2(35.,8.),0.,'  DIES IST EIN BEISPIEL  '));          00000410
      PLOT(LINE(POINT2(0.,22.),POINT2(170.,22.)));                       00000420
   END GIPSY;                                                            00000430
   FINISH;                                                               00000440
 END PROGRAM;                                                            00000450
//C.SYSPRINT DD SYSOUT=A                                                  00000460
//G.SYSPRINT DD SYSOUT=A                                                  00000470
//G.PLOTTAPE DD       UNIT=T0800,VOL=SER=P0251,LABEL=(,NL),DCB=DEN=2      00000480
```

- Zeilen 10-60 und 460-480 bilden die (installationsabhängige) Umgebung eines REGENT-Laufes.
- Zeilen 70, 80 und 440, 450 stellen Anfang und Ende eines REGENT-Programmes dar. Bezüglich der REGENT-Optionen in Zeile 80 sei auf /-1_/verwiesen. Hier bedeuten die Optionen

  Es wird graphische Ausgabe auf einem CALCOMP-Plotter erzeugt.

  Ein Speicherbereich von 80000 Bytes (Modulpool) ist für die Module der benutzten REGENT-Subsysteme (hier nur GIPSY) reserviert. Empfehlungen zur Wahl der Poolgröße siehe Kap. 7.2.
- Zeilen 90 und 430 kennzeichnen Anfang und Ende einer GIPSY-Anwendung. In Kap. 7.4 wird erläutert, wie man eine derartige Anwendung innerhalb eines REGENT-Programmes unterbrechen (LEAVE) und wieder aufnehmen (REENTER) kann, ohne daß Information verloren geht.
- Die Zeilen 100 bis 160 enthalten die für das Programm erforderlichen Vereinbarungen:

  PI_DRITTEL ist selbsterklärend. Man beachte, daß der Wert des Winkels 60° im Bogenmaß (RAD) angegeben ist. Dies muß berücksichtigt werden, wenn der Wert in GIPSY-Anweisungen als Winkelangabe verwendet wird (siehe Zeile 290). Das Verwechseln der Einheiten für Winkelmaße ist ein häufig auftretender Fehler.

  RADIUS ist der Zahlenwert der Kantenlänge eines Sechsecks (bzw. der Radius des einhüllenden Kreises). Ein Vergleich des Zahlenwertes 0.01 mit der Skizze zeigt, daß diese Länge vom Programmersteller in Meter verstanden wurde. Daß es sich um Meter handelt, ist jedoch dem Zahlenwert nicht anzusehen. Diese zusätzliche Information werden wir hinzufügen müssen, wenn wir den Zahlenwert in GIPSY-Anweisungen verwenden. Siehe Zeile 170.

  ABSTAND ist der Abstand der Sechseckmittelpunkte. Hinsichtlich der Längeneinheit (auch hier Meter) gilt dasselbe wie für RADIUS.

SECHS_PUNKTE soll die Koordinaten der sechs Eckpunkte eines Sechseckes aufnehmen. Nach diesen PL/1-Deklarationen folgen Deklarationen von graphischen "Objekten", die GIPSY-spezifisch sind.
HEXAGON ist ein Polygonzug mit 6 Stützpunkten. Das Attribut POLY2 besagt, daß es sich um einen zweidimensionalen Polygonzug handelt; die 6 in Klammern gibt die Maximalzahl der Stützpunkte an. HEXAGON soll im Programm jeweils ein Sechseck enthalten.
PUNKT ist ein einzelner Punkt. Auch hier gibt die "2" in POINT2 an, daß es ein zweidimensionales Objekt ist. Generell unterscheiden sich in GIPSY die zweidimensionalen Objekte von den entsprechenden dreidimensionalen durch eine solche angehängte "2". Wir werden PUNKT in diesem Programm nicht verwenden.
SECHSECK ist eine graphische Kollektion (oder Menge). Sie gestattet es, eine Menge graphischer Objekte unter einem Namen zusammenzufassen. SECHSECK wird später in verschiedener Weise verwendet. Man beachte die Ähnlichkeit der GIPSY-Deklarationen mit normalen PL/1-Deklarationen. Ebenso wie

```
DCL A(N) BIN FLOAT;
```

ein Feld von N Zahlen vom Typ BIN FLOAT (binäre Gleitkommazahl)deklariert, so deklariert

```
DCL B(N) POINT2;
```

ein Feld von N zweidimensionalen Punkten bzw.

```
DCL X(N) COLLECTION;
```

ein Feld von N Kollektionen.

- Mit Zeile 170 beginnen die ausführbaren Anweisungen des GIPSY-Programms. Der Befehl

```
CHANGE UNIT
```

ist ein Sonderfall des allgemeineren CHANGE-Befehls. Die Option UNIT besagt, daß hiermit festgelegt wird, welche Einheit mit den Werten der Dimensionen Länge und Winkel zu verbinden ist, sofern diese Einheit nicht an Ort und Stelle explizit angegeben wird. Standardwert ist
für Längen: Meter /¯M_/
für Winkel: Grad /¯DEGREE_/.

Demnach wäre die Zeile 170 überflüssig. Es empfiehlt sich jedoch zur Erhöhung der Lesbarkeit eines GIPSY-Programmes, die Einheiten stets explizit anzugeben. In den meisten Fällen hat sich die Anweisung

CHANGE UNIT LENGTH M, ANGLE RAD;

bewährt, da dann Längen immer in der Normeinheit Meter und Winkel in der Einheit verwendet werden, die man auch für die Winkelfunktionen SIN und COS benötigt.

- Zeile 180 eröffnet ein neues Bild (hier das erste und einzige). Die beiden Zahlenangaben legen die Horizontalabmessung (17 cm) und Vertikalabmessung (12 cm) fest. Wegen des vorangegangenen CHANGE UNIT-Befehls sind die Zahlenwerte als Meter zu verstehen. Die Wirkung dieses Befehls ist die Erzeugung der rechteckigen Bildumrandung. Ferner wird sichergestellt, daß auf der Zeichenfläche außerhalb dieses Rahmens nichts gezeichnet wird, auch wenn das Programm dies fehlerhafterweise versuchen sollte. Außerdem wird standardmäßig der Nullpunkt des Koordinatensystems mit der linken unteren Ecke assoziiert.
- Die Zeilen 190 bis 220 sind reines PL/1. Sie dienen zur Berechnung der Koordinaten von sechs Punkten, die um den Nullpunkt im Abstand RADIUS angeordnet sind. Man beachte, daß ein aus diesen Punkten gebildetes Sechseck mit zwei Spitzen nach rechts und links weisen würde und nicht - wie verlangt - nach oben und unten.
- In den Zeilen 230 bis 260 wird aus den eben berechneten Koordinaten eine Kollektion von 6 Punkten gemacht. Dabei wird von der Tatsache Gebrauch gemacht, daß jede vereinbarte Kollektion (hier also SECHSECK) zunächst leer ist. Daher kann in Zeile 240 SECHSECK auf der rechten Seite verwendet werden, ohne daß ihm zuvor ein Wert zugewiesen wurde.
- Zeile 240, 250 soll im Detail erläutert werden. Die Anweisung

  SET graphische Variable = ein graphischer Ausdruck;

  stellt eine graphische Zuweisung dar. Hier wird also der Variablen SECHSECK (welche eine Kollektion darstellt) eine Kollektion zugewiesen. COLLECTION ist eine GIPSY-spezifische Funktion (sogenannte BUILTIN-Funktion), die als Argument eine Liste graphischer Objekte besitzt. Ebenso ist POINT2 eine GIPSY-BUILTIN-Funktion, die aus zwei Koordinaten einen zweidimensionalen Punkt erzeugt. Bei jedem der 6 Durchläufe der Schleife

wird also der bereits bestehenden Kollektion SECHSECK ein weiterer Punkt hinzugefügt.

- In Zeile 270 wird unter Verwendung der BUILTIN-Funktion POLY2 aus der Kollektion der sechs Punkte ein Polygonzug erzeugt. Man beachte, daß eine Anweisung

```
SET HEXAGON = SECHSECK;
```

ein Fehler wäre, da beide Seiten der Zuweisung verschiedene graphische Objekttypen (POLY2 bzw. COLLECTION) darstellen. Bei der Ausführung würde dann eine Fehlermeldung erzeugt. Die Funktion POLY2 erzeugt stets einen offenen Polygonzug, d.h. der letzte Punkt wird mit dem ersten Punkt nicht durch einen Strich verbunden.

- Zeile 280 dient dazu, den Polygonzug zu schließen. Die EDIT-Anweisung könnte auch dazu benutzt werden, den Linientyp des Polygonzugs zu ändern. Da bislang nichts besonderes festgelegt wurde, entspricht HEXAGON einer durchgezogenen Linie, die das Sechseck darstellt. Eine Anweisung

```
EDIT CLOSED,DASHED,LENGTH(3MM) OF (HEXAGON);
```

würde bewirken, daß das Sechseck durch eine gestrichelte Linie dargestellt würde.
- In Zeile 290 wird der Polygonzug um 30° gedreht. Man beachte die Spezifikation der Winkeleinheit RAD. Dies ist erforderlich, da immer noch der Standardwert DEGREE gilt (siehe Zeile 170).
- In Zeile 300 erhält die Kollektion, die bislang sechs einzelne Punkte umfaßte, als neuen Inhalt das inzwischen gedrehte Sechseck zugewiesen.
- Zeilen 310 bis 360 fügen der Kollektion SECHSECK bei jedem der sechs Schleifendurchgänge ein neues Sechseck hinzu, das durch Verschiebung von HEXAGON um ABSTAND in Richtung $n \cdot \frac{\pi}{3}$ entsteht. Für den Verschiebungsbetrag gilt gemäß Zeile 170 die Einheit Meter.
- Die Kollektion der sieben Sechsecke ist noch um den Nullpunkt des Koordinatensystems angeordnet. Der Mittelpunkt des innersten Sechsecks liegt also noch in der linken unteren Ecke des OPEN PLOT-Rahmens (siehe Zeile 180). In Zeile 370 wird die Sechseck-Kollektion um 8 cm nach rechts und 6 cm nach oben (Längeneinheit immer noch Meter) verschoben.

- In Zeile 380 wird die Kollektion gezeichnet.
- Zeile 390 weist auf eine Eigenheit der Kollektionen hin, die mit der Implementierung zusammenhängt. Kollektionen können sehr umfangreiche Datenmengen beinhalten. Diese Datenmengen belegen Platz im Arbeitsspeicher. Man sollte, um diesen Platz nicht unnötig zu belegen, eine Kollektion wieder "leer" machen, sobald man sie nicht mehr benötigt. Dies geschieht durch die Anweisung EMPTY.
  Der Anwender sollte sich dies von vornherein angewöhnen, auch wenn in einem einfachen Programm wie diesem das Speicherplatzproblem keine Rolle spielt. Bei umfangreicheren Programmen, speziell wenn Kollektionen innerhalb eines mehrfach aufgerufenen Unterprogrammes (Blockes oder Moduls) erzeugt werden, kann das Nichtbeachten dieser Regel zur Folge haben, daß der gesamte Arbeitsspeicher bis zum Programmabbruch mit längst irrelevanten und nicht mehr zugänglichen Daten vollgestopft wird.
- Zeile 400 setzt die Längeneinheit auf mm.
- In Zeile 410 wird ein Text gezeichnet. TEXT2 ist eine GIPSY-Funktion (ähnlich wie die bereits kennengelernten POINT2 etc.), welche einen graphischen Text erzeugt. Die Argumente der Funktion sind
  der Ort des linken unteren Eckpunktes des ersten Zeichens, definiert durch ein Punktobjekt;
  die Richtung der Grundlinie des Textes, hier also 0 Grad;
  die Zeichenkette des Textes.
  Die Schriftgröße des Textes wird hier nicht durch die Funktion selbst festgelegt. Sie ist standardgemäß 5 mm. Die Schriftgröße (Höhe und Breite der Buchstaben) und die Neigung der Buchstaben könnte jedoch hier bei der Erzeugung des Textes angegeben werden. Diese Angaben können aber auch später noch durch EDIT verändert werden.
- Abschließend wird in Zeile 420 eine Linie gezeichnet, die die Punkte (0 mm, 22 mm) und (170 mm, 22 mm) verbindet.

Plotausgabe des GIPSY-Programms:

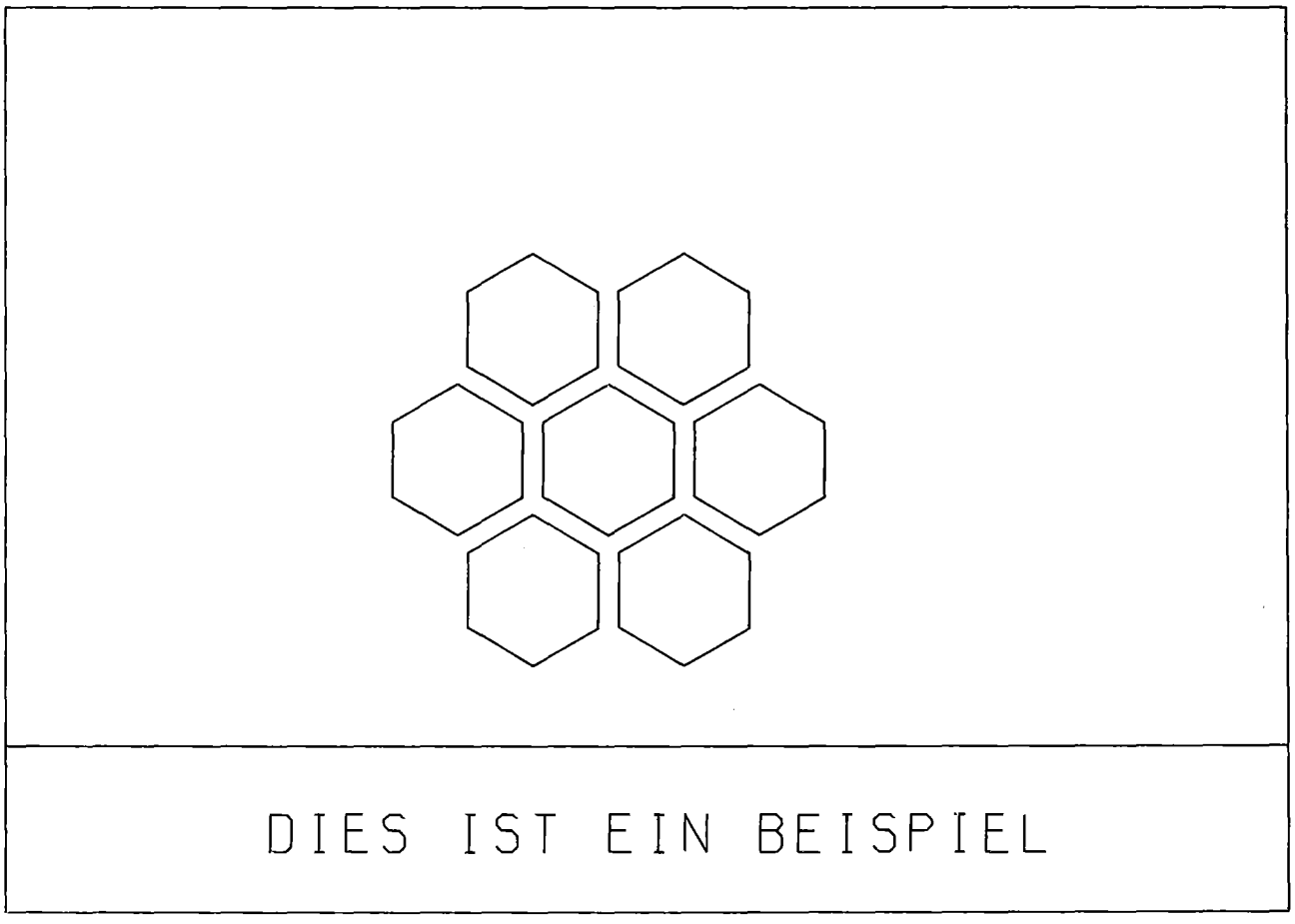


Abb. 2.3: Plotausgabe

3. Die GIPSY-Welt

3.1 Übersicht

Die graphischen Objekte, die von GIPSY behandelt werden, können in drei Gruppen eingeteilt werden:

- zweidimensionale Objekte.
  Dazu gehören u. a. Polygonzüge gegeben durch Punte (x_i, y_i), Punkte (x, y) und zweidimensionale Texte
- dreidimensionale Strichobjekte,
  z. B. dreidimensionale Polygonzüge (x_i, y_i, z_i), Punkte (x,y,z)
- dreidimensionale Objekte zum Aufbau von Körpern.
  Zum Beschreiben von Körpern werden Flächen im Raum benutzt: Ebene, Zylinder, Kegel und Kugel.

Die Gruppe der zweidimensionalen Objekte existiert in einem zweidimensionalen Raum, dem 2D-Raum.
Die beiden anderen Gruppen sind in einem dreidimensionalen Raum (3D-Raum) definiert. In den folgenden Abschnitten wird der 2D-Raum und der 3D-Raum, der Zusammenhang zwischen ihnen und die Abbildung der Objekte aus diesen Räumen auf eine Zeichenfläche beschrieben.

Abb. 3.1 gibt einen Überblick über die GIPSY-Welt, die Räume, in denen GIPSY-Objekte existieren und die Übergänge zwischen ihnen. In den folgenden Abschnitten wird auf die Ziffern in dieser Abbildung Bezug genommen.

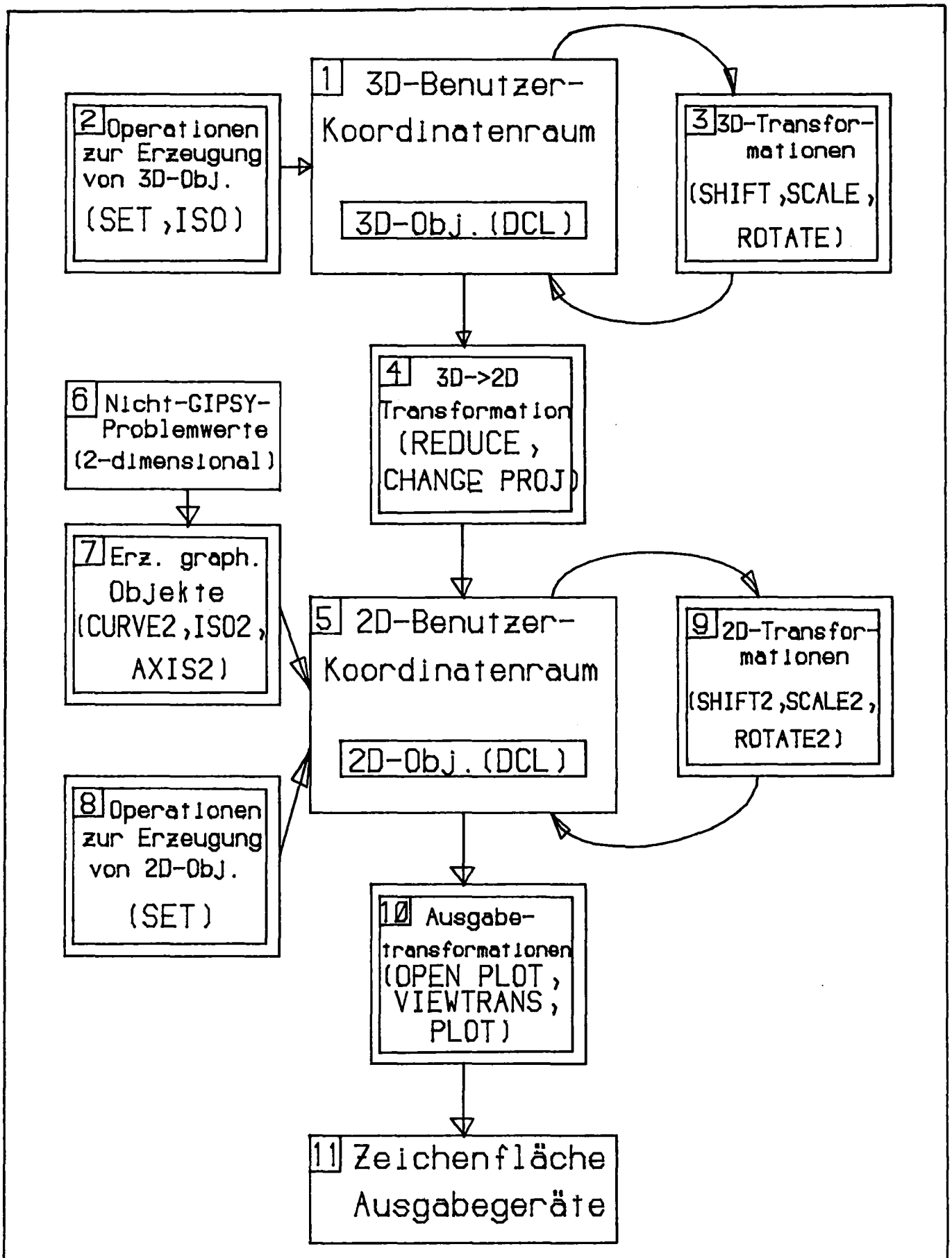


Abb. 3.1 : Die GIPSY-Welt

3.2 Der 2D-Raum

Die zweidimensionalen GIPSY-Objekte existieren auf einer unendlich ausgedehnten, zweidimensionalen ebenen Fläche. Punkte auf dieser Ebene werden durch ein Koordinatenpaar (x,y), bezogen auf ein rechtwinkliges kartesisches Koordinatensystem, beschrieben. Weitere Objekte im 2D-Raum sind Polygonzüge, die aus der Verbindung mehrerer Punkte bestehen, Kreise, Bögen und Texte. Auch der Mittelpunkt von Kreis und Bogen und der Anfangspunkt eines Textes werden durch Punktkoordinaten (x,y) gegeben. Alle diese Koordinatenwerte haben in GIPSY die Dimension einer Länge (normalerweise Meter). Auch der Radius eines Kreises und die Höhe eines Textes haben die Dimension einer Länge. Da Längen und Koordinatenwerte in der Rechenanlage durch Gleitkommazahlen dargestellt werden, ist der Wertevorrat auf den auf der Anlage darstellbaren Zahlenraum beschränkt. Zur Beschreibung der Richtung eines Textes und für Anfang und Ende eines Bogens werden außer Längen- auch Winkelangaben benötigt. Ein Winkel im 2D-Raum wird stets von der positiven x-Achse aus im Gegenuhrzeigersinn positiv gezählt, siehe Abb. 3.2.

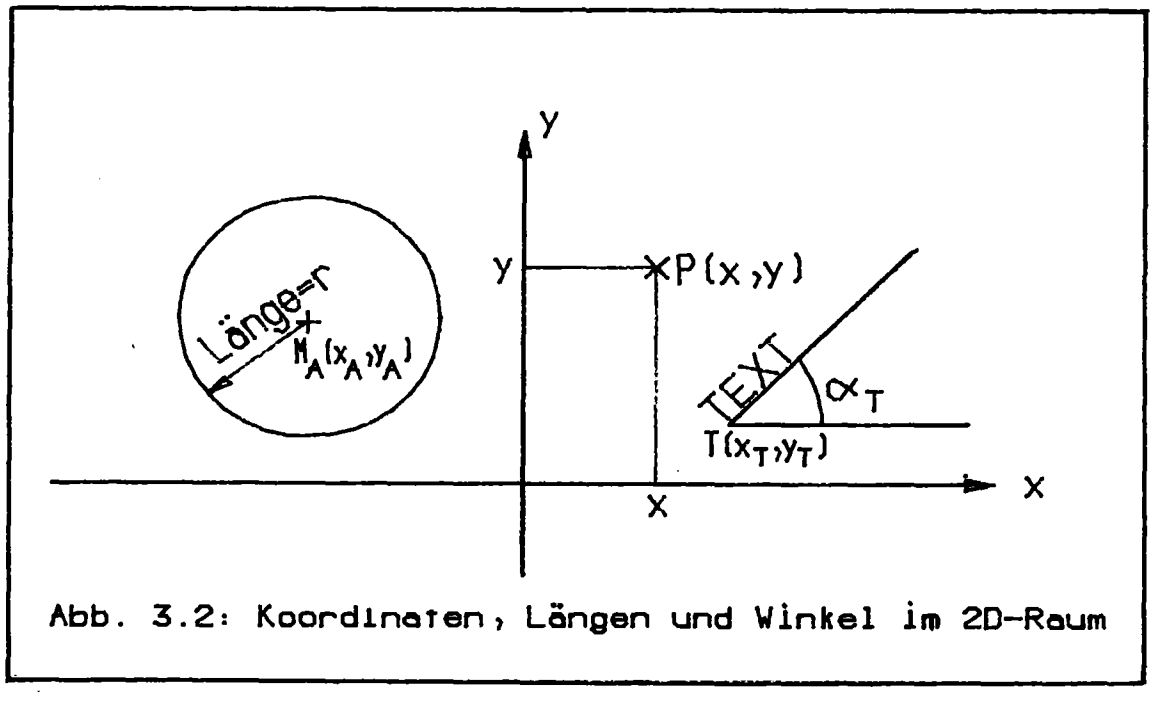


Abb. 3.2: Koordinaten, Längen und Winkel im 2D-Raum

Abb. 3.2: Koordinaten, Längen und Winkel im 2D-Raum.

In Abb. 3.1 ist der 2D-Raum durch den Kasten Nr. 5 dargestellt. Alle Objekte, deren Typenbezeichnung mit der Ziffer 2 endet, sind durch ihre Deklaration in diesem 2D-Raum definiert (DCL P POINT2). Dieser Raum ist ein abstrakter Raum, in dem die Koordinaten und Abmessungen der graphischen Objekte von denjenigen realen Objekten abgeleitet sind, die sie darstellen. Eine Landkarte kann also in diesem Raum Koordinatenwerte von Hunderten von Kilometern besitzen, während ein Atommodell in diesem Raum eine Ausdehnung von 10^{-10} m haben kann. Die Koordinaten sind also durch das vom Benutzer behandelte Problem bestimmt, wir sprechen daher von Benutzer-Koordinaten.

Den durch eine Deklaration im 2D-Raum definierten Objekten (DCL) kann durch eine graphische Zuweisung (SET) ein Wert zugewiesen werden, dazu können GIPSY-Funktionen zur Erzeugung zweidimensionaler Objekte benutzt werden (Kasten Nr. 8). Die GIPSY-Funktionen erzeugen aus numerischen (auch alphanumerischen) Problemwerten des Anwenders graphische Objekte. Der Weg von Nr. 6 über 7 nach 5 macht die Erzeugung zweidimensionaler graphischer Objekte aus zweidimensionalen Problemwerten deutlich. Dabei wird aus einem Raum (a, b), wobei a und b beliebige Dimensionen haben können, in den 2D-Raum (x, y) transformiert. Die GIPSY-Funktionen CURVE2 und AXIS2 führen diese Transformation durch. Damit können physikalische Werte in einem Diagramm mit Achsen dargestellt werden. Im GIPSY-2D-Raum kann es also keine Drücke, Temperaturen, Zeiten oder Geschwindigkeiten geben, lediglich die Abbildung physikalischer Wertepaare wird in den 2D-Raum aufgenommen, die Koordinaten haben darin aber die Dimension einer Länge.

Graphische 2D-Objekte können innerhalb des 2D-Raums linear transformiert werden (Kasten 9). Lineartransformationen sind Verschiebung, Vergrößerung bzw. Verkleinerung und Drehung. Aus der Transformation eines 2D-Objektes entsteht stets wieder ein 2D-Objekt. Der Name der 2D-Transformationen endet mit der Ziffer 2: SHIFT2,SCALE2,ROTATE2.
Eine weitere Möglichkeit zur Erzeugung von 2D-Objekten ist die Transformation aus dem 3D-Raum in den 2D-Raum (Kasten 4). Darauf wird in Abschnitt 3.5 näher eingegangen. Den Ausgang aus dem 2D-Raum Richtung Ausgabegerät behandeln die Abschnitte 3.4 und 3.6.

3.3 Der 3D-Raum

Alle dreidimensionalen GIPSY-Objekte existieren in einem unendlich ausgedehnten dreidimensionalen Raum (Kasten 1). Die für den 2D-Raum gemachten Aussagen gelten sinngemäß auch für den 3D-Raum.

- Punkte sind definiert durch ein Wertetripel (x,y,z oder x(1),x(2),x(3)) in einem rechtwinkligen kartesischen Koordinatensystem.
- Alle Koordinatenwerte haben die Dimension einer Länge.
- Der Wertevorrat von Koordinaten und Längen ist durch den auf der Rechenanlage vorhandenen Zahlenraum der Gleitkommazahlen gegeben.
- Der 3D-Raum ist ein abstrakter Raum, die Koordinaten und Abmessungen spiegeln Eigenschaften der dargestellten realen Objekte wider (Benutzerkoordinaten).
- Definition von 3D-Objekten durch eine Deklaration (DCL). Die Typbezeichnungen der 3D-Objekte enden nicht mit einer Ziffer (DCL P POINT).
- Zuweisung von Werten durch graphische Operationen zur Erzeugung von 3D-Objekten (Kasten 2) aus Problemwerten.
- Lineartransformationen sind Verschiebung, Vergrößerung/Verkleinerung und Drehung (Kasten 3). Die 3D-Transformationen SHIFT,SCALE und ROTATE erzeugen aus einem 3D-Objekt stets wieder ein 3D-Objekt.

Ebenso wie der 2D-Raum enthält auch der 3D-Raum graphische Linienobjekte. Es sind dies Punkt, Polygonzug, Text und Kreis. Darüber hinaus existieren im 3D-Raum jedoch Objekte, die auch eine dreidimensionale Ausdehnung haben, nämlich Körper (BODY). Ein Körper ist zusammengesetzt aus mehreren Teilkörpern, die in GIPSY Raumelemente (SPACE) heißen. Jedes Raumelement ist durch Flächen begrenzt. Die Flächen sind Ebene, Zylinder, Kegel und Kugel. Flächen, Raumelemente und Körper werden wie die Liniengebilde durch eine Deklaration definiert (DCL B BODY). Funktionen dienen zur Wertezuweisung und zum Aufbau von Raumelementen und Körpern. Der einzige Ausgang aus dem 3D-Raum ist (zur Zeit) der Weg (1-4-5-10-11) über eine Transformation in den 2D-Raum und schließlich von dort Richtung Ausgabegerät. Die Tranformation aus dem 3D-Raum in den 2D-Raum wird in Abschnitt 3.5 behandelt.

3.4 Darstellung von 2D-Objekten

Da die 2D-Objekte in einem abstrakten 2D-Raum definiert sind, müssen sie zu ihrer Darstellung aus dem Benutzerkoordinatenraum auf ein reales Ausgabemedium wie Zeichengerät-Papierfläche oder Bildschirmoberfläche transformiert werden. Die Koordinaten auf den Ausgabegeräten heißen Gerätekoordinaten. Auch die Gerätekoordinaten haben die Dimension einer Länge und sind damit in GIPSY mit einer Längeneinheit versehen. Die Ausgabe erfolgt in zwei Schritten, dem Setzen von Transformationswerten und dem eigentlichen Ausgeben mit der PLOT-Anweisung. Mit Hilfe der OPEN-PLOT-Anweisung wird eine neue Arbeitsfläche (Papierbereich bzw. gelöschter Bildschirm) angefordert. Die VIEWING TRANSFORMATION erlaubt das Setzen von Transformationswerten. Außerdem kann mit ihr eine Blende definiert werden, die die Darstellung der graphischen Objekte außerhalb eines rechteckigen Bereiches unterdrückt. Die Ausgabe von 2D-Objekten erfolgt also durch die Folge:

```
OPEN PLOT
 VIEWING TRANSFORMATION
     PLOT
     PLOT
 VIEWING TRANSFORMATION
     PLOT....
```

Sowohl für OPEN PLOT als auch für VIEWING TRANSFORMATION(abgekürzt VIEW) sind Standardwerte vorgesehen. Ausführliche Beispiele zur Ausgabetransformation finden sich in Kap.8.1.

3.5 Darstellung von 3D-Objekten

Die Darstellung von 3D-Objekten erfolgt in zwei Schritten:

- der Überführung der 3D-Objekte durch eine Transformation in 2D-Objekte
- der Ausgabe der entstandenen 2D-Objekte mit einer Ausgabetransformation auf ein Ausgabegerät.

Der zweite Schritt wurde schon behandelt, der erste Schritt wird durchgeführt durch Projektion der 3D-Objekte auf eine 2D-Ebene. Die Ebene, auf die projiziert wird, heißt Bildebene.

GIPSY kennt zwei Arten von Projektionen, die Parallelprojektion (Abb. 3.3) und die Zentralprojektion (Abb. 3.4). Um die Wirkung der Projektion eindeutig zu bestimmen, sind folgende Angaben nötig:

a) Art der Projektion (Parallel, Zentral)
b) Lage der Bildebene im 3D-Raum (Punkt auf der Ebene und Normale)
c) Projektionsursprung (Zentralp.) bzw. Projektionsrichtung (Parallelp.) im 3D-Raum
d) Lage des Nullpunktes des 2D-Koordinatensystems (x_o, y_o) auf der Bildebene
e) Orientierung der Koordinatenachsen des 2D-Koordinatensystems.

Die Anweisung CHANGE PROJECTION wird benutzt, um die Projektionsparameter (Punkte a bis d) zu setzen. Punkt e (Orientierung der Koordinatenachsen) ist in GIPSY standardgemäß festgelegt (siehe Beschreibung der REDUCE-Funktion). Die Projektion geschieht durch die Funktion REDUCE, die als Ergebnis ein Objekt im 2D-Raum liefert oder aber implizit, wenn in der PLOT-Anweisung 3D-Objekte angesprochen werden. Im letzteren Fall wird der Weg 1-4-5-10-11 automatisch bei Ausführung der PLOT-Anweisung durchlaufen.

Ein weiterer möglicher Weg zur Darstellung von 3D-Objekten, nämlich eine direkte Ausgabe vom 3D-Raum auf ein 3D-Ausgabegerät, ist in GIPSY (noch) nicht enthalten. Dieser Weg wäre sinnvoll zur Ausnutzung von dreidimensionalen Bildschirmgeräten, die die Transformation 3D-2D und die Ausgabetransformationen selbst effektiv (durch Hardware) durchführen. Die Projektion ließe sich also am Ausgabegerät nachträglich einstellen.

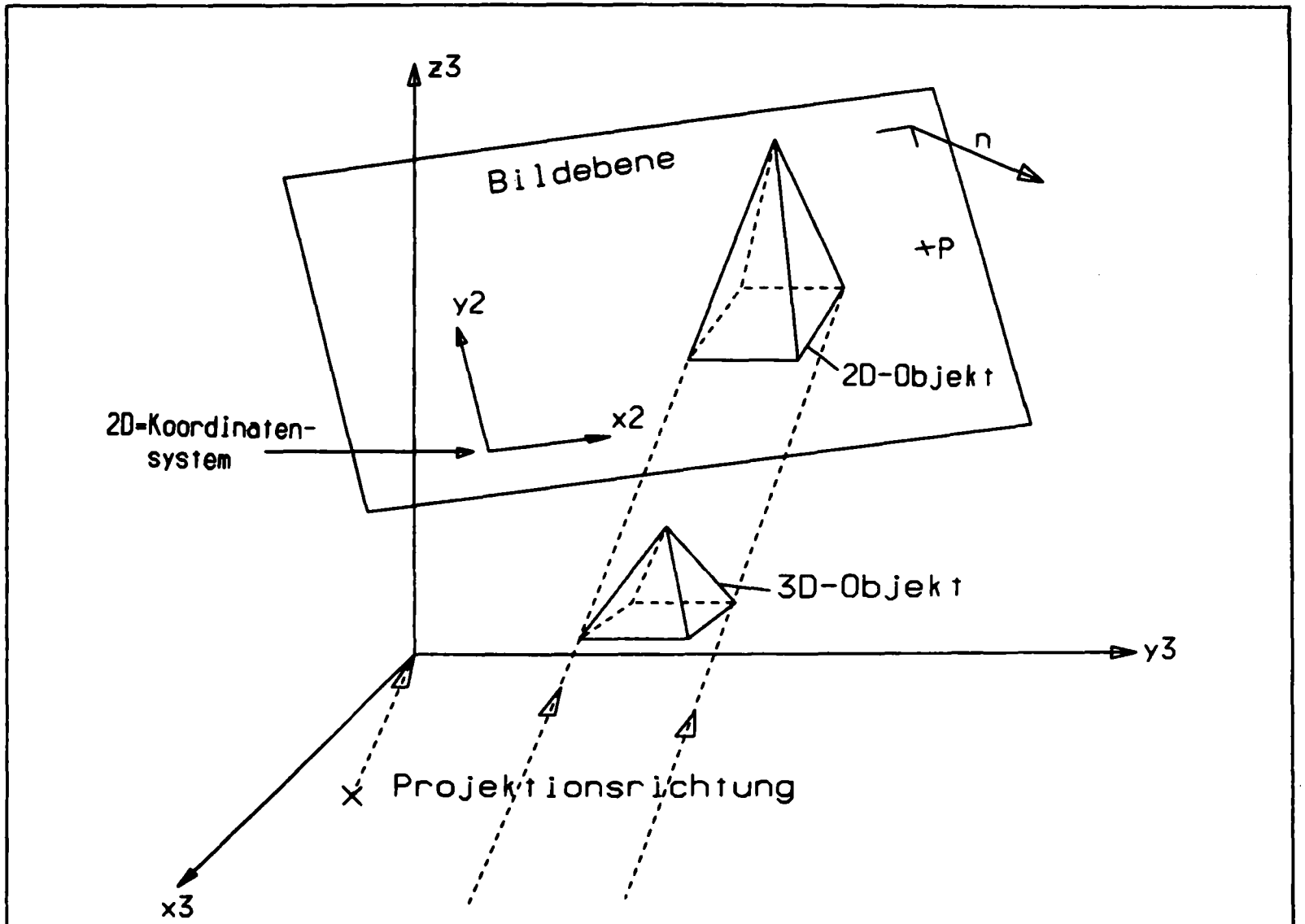


Abb. 3.3: Parallelprojektion

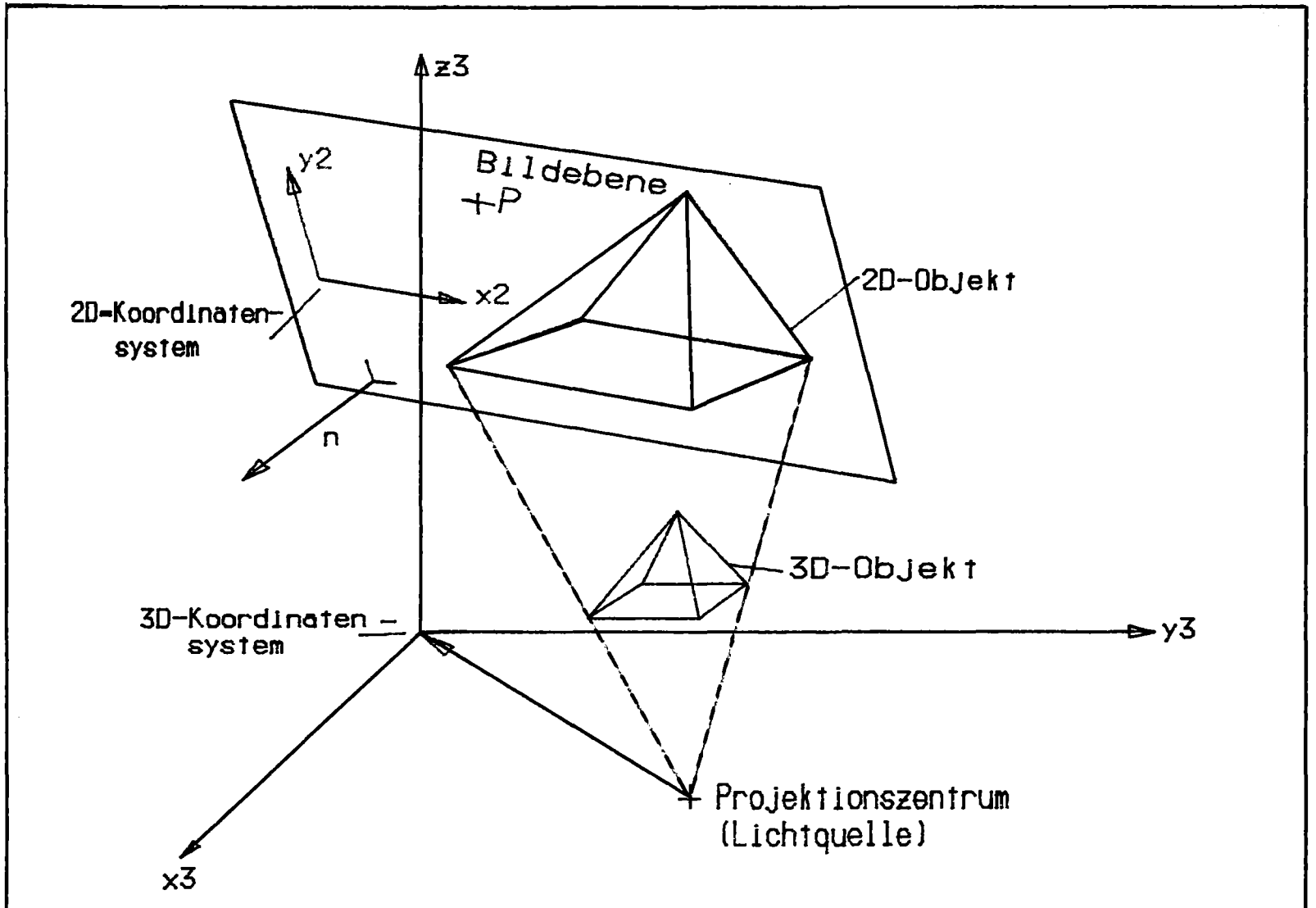


Abb. 3.4: Zentralprojektion

3.6 Ausgabegeräte

Um die graphischen Objekte, die im 2D- oder 3D-Raum existieren, sichtbar zu machen, müssen sie auf einem Ausgabegerät dargestellt werden. Zum langfristigen Speichern und zum Transportieren graphischer Information muß sie auf eine Datei ausgegeben werden können. In allen Fällen erfolgt die Ausgabe durch die PLOT-Anweisung, (Kapitel 6.3 bis 6.5) die Auswahl eines bestimmten Ausgabegerätes steuert der Benutzer (Kapitel 7.2 und 7.3). Aufgrund dieser Auswahl kennt GIPSY bestimmte Daten des Ausgabegerätes (z. B. Größe der vorhandenen Zeichenfläche) und kann auf Fehler spezifisch reagieren. Die Auswahl der Ausgabegeräte hängt von der Installation ab, einige gebräuchliche Ausgabegeräte sollen hier kurz beschrieben werden.

Trommelzeichenmaschine

Die Zeichenfläche ist eine Papierbahn mit einer Höhe von etwa 25 cm bis 1 m und einer großen Länge von einigen 10 m. Da die Papierbahn über eine Trommel geführt wird, spricht man von einer Trommelzeichenmaschine oder einem Trommelplotter. Auf dem Papier kann ein Stift in beliebige x-y-Positionen gefahren werden, dabei kann er gehoben oder abgesenkt sein. Als Stifte werden üblicherweise Kugelschreiber, Tuschefedern oder Filzschreiber verwendet. Aus GIPSY-Ausgabeanweisungen werden letzlich Stiftbewegungen des Plotters erzeugt. GIPSY sorgt dafür, daß die durch OPEN PLOT angeforderten Zeichenflächen sich auf dem Papier nicht überschneiden und die zur Verfügung stehende Papierfläche nicht überschreiten. Trommelplotter sind die einfachsten und billigsten graphischen Ausgabegeräte.

Tischzeichenmaschine

Im Gegensatz zu einem Trommelplotter benutzt ein Tischplotter eine Papierbahn fester Länge und Breite, die auf einem meist waagerechten Tisch aufgelegt ist, als Ausgabefläche. Viele Tischplotter können Zeichnungen hoher Präzision herstellen, oft ist ein Umschalten zwischen mehreren Stiften möglich. Die Stiftauswahl erfolgt in GIPSY mit der PEN-Attribut-Angabe (EDIT PEN).

Auch bei Tischplottern verhindert GIPSY das Überschreiten der zur Verfügung stehenden Papierfläche (sofern das richtige Papierformat aufgelegt wurde).

Elektrostatische Zeichenmaschine (Printer-Plotter)

Eine elektrostatische Zeichenmaschine beschreibt wie ein Trommelplotter eine sehr lange Papierbahn, benutzt dazu jedoch keine Stifte. Die Bahn aus Spezialpapier wird langsam an einer Schreibstation entlanggeführt, wo es zeilenweise, vergleichbar einem Fernsehschirm, beschrieben wird. Die Zeichenfläche ist also in ein Punkteraster eingeteilt, jeder Punkt ist entweder schwarz oder weiß. Die Zeichengeschwindigkeit ist bei diesem Plottertyp unabhängig von der Anzahl der Striche auf der Zeichnung, die Qualität ist schlechter als bei Geräten mit Stiften. Ein elektrostatischer Plotter ist immer dann von Vorteil, wenn große Mengen von Bildern gezeichnet werden, wenn die Bildinformation aus sehr vielen Einzelpunkten oder Einzelstrichen besteht, wenn eine schnelle Zeichenausgabe gewünscht wird oder wenn an die Qualität keine hohen Ansprüche gestellt werden. Ein elektrostatischer Plotter kann auch als Zeilendrucker verwendet werden.

Bildschirme

Bei Anwendung von GIPSY sind auch Bildschirme zunächst nur passive graphische Ausgabegeräte, da GIPSY keine Anweisungen zur graphischen Eingabe enthält. In Verbindung mit PL/1 können natürlich auch interaktive Anwendungen in der GIPSY-Sprache programmiert werden. Die Ausgabe auf einen Bildschirm erfolgt wie bei Zeichengeräten mit der PLOT-Anweisung. Bei OPEN PLOT, dem Anfordern einer neuen Zeichenfläche, wird auf eine Tastatureingabe gewartet und danach der Bildschirm gelöscht und das neue Bild bis zum nächsten OPEN PLOT gezeichnet. Dies gilt gleichermaßen für Speicherbildschirme, die das erzeugte Bild in der Bildröhre speichern, als auch für Bildschirme mit Bildwiederholspeicher. Eine nachträgliche Manipulation der Bilder am Terminal ist nur möglich in Verbindung mit der Datei zur Speicherung graphischer Information, dem Plotfile.

Mikrofilmplotter

Bei einem Mikrofilmplotter werden die Zeichnungen mit Hilfe eines Elektronenstrahles auf Mikrofilm-Karten geschrieben. Dies erlaubt eine sehr kompakte Form der Ausgabe großer Mengen von Zeichnungen. Wird statt der Film-Karten ein 16- oder 35-mm-Film in das Gerät eingelegt, so kann durch Erzeugung einer Sequenz von Bildern ein Film mit bewegten Objekten erzeugt werden.

Plotfile

Hier wird die graphische Information nicht auf ein graphisches Ausgabegerät geleitet, sondern in strukturierter Form auf eine Datei geschrieben, die Plotfile genannt wird. Von dieser Datei kann die graphische Information langfristig gespeichert, nachträglich geändert, transportiert und ausgegeben werden. Ein graphisches Editieren der von GIPSY erzeugten Bilder wird zweckmäßigerweise über den Plotfile vorgenommen. Außerdem bietet der Plotfile eine Schnittstelle zu anderen graphischen Systemen.

4. Die GIPSY-Objekte und ihre Attribute

4.1 Objekttypen und Attribute

Wie in Kap. 3 erläutert kennt GIPSY "graphische Objekte", die entweder zweidimensional oder dreidimensional sind. In einer konkreten Anwendung muß man unterscheiden zwischen

- dem konkreten Objekt, welches einen vom Benutzer im Rahmen gewisser Konventionen willkürlich gewählten und im Anwendungsprogramm festgelegten Namen hat;
- und dem Objekttyp, der angibt von welcher Art dieses Objekt ist.

Die Zuordnung von Objekt und Objekttyp geschieht in einer Deklaration (siehe Kap. 4.4). Die Objekttypen und ihre Bezeichnungen sind in GIPSY festgelegt. Es gibt (in der gegenwärtigen Version) genau 18 Objekttypen (siehe Kap. 4.5 - 4.7). Ein Objekt eines bestimmten Objekttyps ist durch einen bestimmten Satz von Attributen mit dazugehörigen Werten definiert. Der Satz von Attributen ist von Objekttyp zu Objekttyp verschieden. Ein Polygonzug hat beispielsweise die Attribute "Linienart" (LINETYPE) und "Anzahl der Punkte", welche beide für ein Objekt vom Typ Punkt irrelevant und daher nicht vorhanden sind. Man kann im allgemeinen unterscheiden zwischen den geometrischen Attributen der Objekte, die meist nur für jeweils einen Objekttyp relevant sind und daher gesondert in Kap. 4.5 und 4.6 behandelt werden, und den nahezu allen Objekten gemeinsamen Standardattributen, welche vor allem beim Darstellungsvorgang (Operation PLOT in Block 10 auf Abb. 3.1) wirksam werden. Diese Standardattribute sowie das allen Objekten gemeinsame Attribut "Referenzpunkt" sind in Kap. 4.3 erläutert.

4.2 Längen- und Winkelangaben

Alle Längenangaben und alle Winkelangaben in GIPSY sind in einer einheitlichen Form, die aber hinsichtlich der gewählten Einheiten sehr flexibel ist, anzugeben. Innerhalb von GIPSY werden alle Längenangaben in der Einheit m, alle Winkelangaben im Bogenmaß gespeichert und benutzt. Hierauf muß der Benutzer nur dann achten, wenn er zu Testzwecken auf die innere Datenstruktur der GIPSY-Objekte zugreift.

Im Handbuch wird eine Längenangabe stets als

lvalue

bezeichnet. Für eine Winkelangabe steht jeweils

avalue.

(z.B. in Kap.9).Längenangaben treten vor allem an folgenden Stellen auf:

- Koordinaten von Punkten
- Abmessungen von Zeichen und Symbolen
- Achsenlängen
- Abmessungen der Zeichenfläche und Ort des Koordinatenursprungs auf der Zeichenfläche
- Verschiebungsangaben
- Abschnittslänge gestrichelter Linien u.ä.

Winkelangaben treten vor allem an folgenden Stellen auf:

- Orientierung von Texten und Achsen
- Drehungsangaben
- Kreisbögen

Das folgende Beispiel soll die verschiedenen Möglichkeiten der Längen- und Winkelangaben erläutern.

```
//IRE0252 JOB (0025,330,P0H1A),SCHLECHTENDAHL,REGION=512K,                00000010
// NOTIFY=IRE025,MSGLEVEL=(0,0)                                           00000020
// EXEC REGENT,PVOL=PNS001,PLIB='TSO253.TIPSY',                           00000030
//   CPARM='OPT(2),S,INC',GPARM='/ISA(100K),REPORT'                       00000040
//P.SYSIN DD *                                                           00000050
          PROGRAM:PROC OPTIONS(MAIN)                                     00000060
               REGENT(PLOT=CALCOMP,MPOOL=80000,LIST);                    00000070
          ENTER GIPSY;                                                   00000080
            DCL P(4) POINT2;                                             00000090
            DCL LABEL TEXT2(2);                                          00000100
            X1=0.02;                                                     00000110
            Y2=2.0;                                                      00000120
  /*01*/    SET P(1)=POINT2(X1,1 CM);                                    00000130
  /*02*/    SET P(2)=POINT2(0.01,Y2 CM);                                 00000140
  /*03*/    CHANGE UNITS LENGTH MM;                                      00000150
            X3=10;                                                       00000160
            X4=0.02;                                                     00000170
            Y4PLUS3=23.;                                                 00000180
  /*04*/    SET P(3)=POINT2(X3,0.03 M);                                  00000190
  /*05*/    SET P(4)=POINT2(X4*200. CM,(Y4PLUS3-11B)*2);                 00000200
  /*06*/    OPEN PLOT SIZE(5 CM,5 CM);                                   00000210
            DO J=1 TO 4;                                                 00000220
  /*07*/    EDIT WIDTH( 1+J MM),HEIGHT((1+J)MM), SYMBOL(J) FOR(P(J));    00000230
            PLOT(P(J));                                                  00000240
            END;                                                         00000250
            CHANGE STANDARD HEIGHT(4 MM);                                00000260
  /*08*/    PLOT (TEXT2(P(1),45.,'P1'));                                 00000270
  /*09*/    PLOT (TEXT2(P(2),3*45. DEGREE,'P2'));                        00000290
            J=3;                                                         00000300
  /*10*/    CHANGE UNIT ANGLE RADIAN;                                    00000310
  /*11*/    PLOT (TEXT2(P(3),(2*J-1)*45. DEGREE ,'P3'));                 00000320
  /*12*/    PLOT (TEXT2(P(4),-ACOS(0)/2 ,'P4'));                         00000330
            END GIPSY;                                                   00000340
          FINISH;                                                        00000350
          END PROGRAM;                                                   00000360
//G.PLOTTAPE DD       UNIT=T0800,VOL=SER=P0252,LABEL=(,NL),DCB=DEN=2     00000370
```

Anweisung Anmerkung

(01) Da zuvor keine CHANGE UNIT LENGTH-Anweisung ausgeführt wurde, gilt die GIPSY-Längeneinheit m. Die x-Koordinate von P(1) erhält den Wert x1 verbunden mit der Standardlängeneinheit. Da x1 den Wert 0.02 hat, ist dies 0.02 m bzw. 2 cm. Die y-Koordinate ist vollständig angegeben, d. h. einschließlich der Einheitsangabe (1 cm).

(02) Für die x-Koordinate wird wieder die Standardeinheit m (also 1 cm). Wegen y2=2.0 wird die y-Koordinate 2 cm.

(03) Von nun an gilt als Standard-Längeneinheit mm.

(04) Wegen der neuen Standardlängeneinheit mm und wegen x3 = 10 wird die x-Koordinate 10 mm oder 1 cm. Die y-Koordinate ist voll angegeben.

(05) Anstelle des Zahlenwertes für die Längenangabe - sei es mit oder ohne explizite nachgestellte Längeneinheit - kann ein beliebiger arithmetischer Ausdruck stehen.

(06) Die Zeichenflächenabmessungen werden meist im DIN-Format oder - wie hier - in cm angegeben.

(07) Die Höhe der eben erzeugten Texte und Symbole wird neu festgesetzt. Ohne die Anweisung würde die Standardhöhe von 5 mm gelten. Auch der Symboltyp wird neu definiert.

(08) Da zuvor keine CHANGE UNIT ANGLE-Anweisung ausgeführt wurde, gilt die GIPSY-Winkeleinheit Grad. Der Text 'P1' wird also (mit der linken unteren Ecke des ersten Zeichens im Punkt P(1)) unter einem Winkel von 45^{o} gegenüber der positiven x-Achse angeordnet (siehe zweites Argument von TEXT2).

(09) Hier enthält die Winkelangabe Zahlenwert und Einheit.

(10) Die Standardwinkeleinheit wird auf das Bogenmaß festgelegt.

(11) Zahlenwert und Einheit Grad sind angegeben. Da J den Wert 3 hat, ergibt sich ein Winkel von $5 \cdot 45^{o} = 225^{o}$.

(12) Da keine Winkeleinheit angegeben ist, ist die in (10) festgelegte Standardeinheit wirksam; der Text erhält die Richtung $-ACOS(0)/2 = -90^{o}/2 = -45^{o}$.

4.3 Referenzpunkt und Standardattribute

4.3.1 Referenzpunkt

Jedes GIPSY-Objekt hat einen Referenzpunkt. Dieser Punkt kann in vielen Anwendungen als Ausgangspunkt für das Hinzufügen weiterer graphischer Operationen benutzt werden. Das ist vor allem dann erforderlich, wenn das graphische Objekt aus einer nicht ganz primitiven Operation hervorgegangen ist. Beispielsweise erzeugt folgendes Programm einen Punkt mit Beschriftung im Mittelpunkt eines Kreises durch drei Punkte.

```
//IRE0253 JOB (0025,330,P0H1A),SCHLECHTENDAHL,REGION=512K,                00000010
// NOTIFY=IRE025,MSGLEVEL=(0,0),MSGCLASS=A                                 00000020
// EXEC REGENT,PVOL=PNS001,PLIB='TSO253.TIPSY',                            00000030
//   CPARM='OPT(2),S,INC',GPARM='/ISA(100K),REPORT'                        00000040
//P.SYSPRINT DD SYSOUT=A                                                   00000050
//P.SYSIN DD *                                                            00000060
          PROGRAM:PROC OPTIONS(MAIN)                                      00000070
               REGENT(PLOT=CALCOMP,MPOOL=80000,LIST);                     00000080
          ENTER GIPSY;                                                    00000090
            DCL P(0:3) POINT2;                                            00000100
            DCL KREIS CIRCLE2;                                            00000110
            CHANGE UNIT LENGTH(CM);                                       00000120
            OPEN PLOT SIZE(10 CM ,10 CM);                                 00000130
            SET P(1)=POINT2(2,2);                                         00000140
            SET P(2)=POINT2(6,2);                                         00000150
            SET P(3)=POINT2(4,8);                                         00000160
            SET KREIS=CIRCLE2(P(1),P(2),P(3),'OUT');                      00000170
            PLOT (KREIS) ;                                                00000180
            DO J=1 TO 3;PLOT(P(J));END;                                   00000190
            SET P(0)=REFPOINT (KREIS);                                    00000200
            CHANGE STANDARD SYMBOL(3),STANDARD HEIGHT(3 MM);              00000210
            PLOT(P(0),TEXT2(P(0),0 DEGREE,'CENTER'));                     00000220
            END GIPSY;                                                    00000230
          FINISH;                                                         00000240
          END PROGRAM;                                                    00000250
//C.SYSPRINT DD SYSOUT=A                                                 00000260
//G.SYSPRINT DD SYSOUT=A                                                 00000270
//G.PLOTTAPE DD       UNIT=T0800,VOL=SER=P0253,LABEL=(,NL),DCB=DEN=2     00000280
```

Als Referenzpunkt ist definiert

bei 2-dimensionalen Objekten

POINT2	der Punkt selbst
CIRCLE2, ARC2	der Mittelpunkt des Kreises
TEXT2	die linke untere Ecke des ersten Zeichens
POLY2	der erste Punkt des Polygonzuges
AXIS2	der Anfangspunkt der Achse

bei 3-dimensionalen Objekten

POINT	der Punkt selbst
CIRCLE, ARC	der Mittelpunkt
TEXT	die linke untere Ecke des ersten Zeichens
POLY	der erste Punkt des Polygonzuges
CONE	die Kegelspitze
CYLINDER	der bei der Erzeugung angegebene erste Achsenpunkt
PLANE	der bei der Erzeugung angegebene Ebenenpunkt
BALL	der Kugelmittelpunkt
SPACE	Referenzpunkt des ersten Elementes
BODY	"
COLLECTION	"

Der Referenzpunkt ist nicht definiert für eine
leere COLLECTION und ein
leeres SPACE-Objekt und ein
leeres BODY-Objekt.
Der Referenzpunkt wird geliefert durch die GIPSY-Builtin-Funktionen

REFPOINT2	bei zweidimensionalen Objekten
REFPOINT	bei dreidimensionalen Objekten

Beide Funktionen erzeugen bei fehlerhafter Anwendung eine Fehlermeldung. Das Ergebnis der betreffenden Operation ist dann undefiniert.

4.3.2 Standardattribute

Zu den Standardattributen zählen

SYMBOL	Kennzeichnung von Punktesymbolen
LINETYPE	Linientypen für Striche
LENGTH	Länge von Teilstrichen
HEIGHT	Höhe von Texten und Punktesymbolen
WIDTH	Breite von Text-Buchstaben
SLANT	Neigung von Text-Buchstaben
COLOR	Farbe
LINEWIDTH	Strichstärke
PEN	Stiftauswahl
OPEN/CLOSED	Schließen von Polygonzügen
EVERY	Angabe, jeder wievielte Punkt eines Linearzuges markiert werden soll

Welche dieser Standardattribute für welche GIPSY-Objekte gültig sind, ist aus der folgenden Tabelle zu entnehmen.

Bei der Darstellung von Körpern entstehen Schnittlinien und Umrißlinien. Für alle sichtbaren Schnitt- und Umrißlinien gelten die Standardattribute mit Ausnahme von CLOSED. Für alle unsichtbaren Schnitt- und Umrißlinien gilt ein entsprechender Satz von Attributen (ebenfalls ohne CLOSED), der durch

CHANGE INVISIBLE

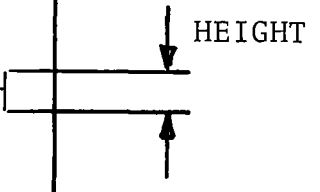
festgelegt wird.

G I P S Y -Standard-Attribute

Attribut	gültig für Objekt	erlaubte Werte	Default
SYMBOL Kennzeichnung von Punktesymbolen	POINT2, POINT CIRCLE2, CIRCLE POLY2, POLY ARC2, ARC	Integer ≥ 0	1
HEIGHT Höhe von Texten und Punktsymbolen	POINT2, POINT CIRCLE2, CIRCLE, POLY2, POLY, AXIS2, ARC2, ARC, TEXT2, TEXT	Real > 0.0	5 mm
EVERY Jeder n-te Punkt wird markiert	CIRCLE2, CIRCLE POLY2, POLY, ARC2, ARC	Integer >0	1
OPEN/CLOSED	POLY2, POLY	OPEN, CLOSED	OPEN
LINETYPE Linienart	CIRCLE2, CIRCLE POLY2, POLY, ARC2, ARC	SOLID, DASHED CENTER, DOTTED MARKED, OMITTED, HAND	SOLID (bei INVISIBLE: OMITTED)
LENGTH Länge von Teilstrichen	wie LINETYPE	Real > 0.0	5 mm
PEN Stiftauswahl	wie HEIGHT	Integer > 0	0 (undefiniert)
COLOR Farbe	wie HEIGHT	Integer > 0	0 (undefiniert)
LINEWIDTH Strichstärke	wie HEIGHT	Real > 0.0	0.0 (undefiniert)
WIDTH Buchstabenbreite	TEXT2, TEXT	Real > 0.0	5 mm
SLANT Buchstabenneigung	TEXT2, TEXT	Real -45 <= SLANT <= +45	0 (senkrechte Buchstaben)
DISTANCE	AXIS2	positiv REAL < Länge der Achse	2 cm

GIPSY-Zeichensatz (durch SYMBOL ansprechbar oder in Texten):

NR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0																				
1-20																				
21-40																				
41-60																				
61-80																				
81-100																				
101-120																				
121-140																				
141-160																				
161-180																				
181-200																				
201-220																				
221-240																				
241-255																				



Man beachte, daß Symbole nur die Höhe von Kleinbuchstaben haben (0.6 HEIGHT).
Die Tabelle wurde mit HEIGHT=WIDTH=5 mm und SLANT=0° erzeugt.

Dem GIPSY-Anwender stehen drei Möglichkeiten zur Verfügung, die Standardattribute zu beeinflussen.

a) Bei der Erzeugung eines Objektes mit einem der Objekttypen gemäß Spalte 2 obiger Tabelle erhält das betreffende Standardattribut den Wert, der gemäß GIPSY-Standardattributliste gerade gültig ist. Diese Standardwerte kann der Anwender mit der

CHANGE STANDARD-Anweisung

festlegen. Wird vom Anwender nichts festgelegt, so gelten die in der letzten Tabellenspalte festgelegten Defaultwerte. Ausnahme hiervon siehe b.

Die in der Tabelle angegebenen Standardwerte können mit der

RESET STANDARD-Anweisung

jederzeit wieder als gültige Werte gesetzt werden.

b) Für bereits bestehende Objekte der oben angegebenen Typen kann der Anwender nachträglich die Standardattribute neu festlegen. Dazu wird die

EDIT-Anweisung

benutzt. Wird die EDIT-Anweisung auf eine Kollektion angewandt, so wirkt sie auf alle Objekte dieser Kollektion einzeln. Wird die EDIT-Anweisung auf Objekte angewandt, für welche das betreffende Attribut nicht gültig ist, so ist sie wirkungslos (kein Fehler!)

4.4 Vereinbarung von GIPSY-Objekten

GIPSY-Objekte werden in der GIPSY-Sprache ganz ähnlich behandelt wie die primitiveren Objekte Festkommazahl, Gleitkommazahl, Zeichenkette etc. in der PL/1-Sprache. Alle Regeln, die von PL/1 her bekannt sind, sind sinngemäß zu übertragen. So muß z. B. ein GIPSY-Objekt in dem Programmblock, in dem es verwendet wird, vereinbart sein.

Beispiel:

```
BEGIN;
  DCL P1 POINT2;
  SET P1 = POINT2(3 CM, 4 CM);
  PLOT(P1);
END;
```

oder auch

```
DCL P2 POINT2;
BEGIN;
SET P2 = POINT2 (2 CM, 1 CM);
END;
```

sind richtig. Falsch wäre z. B.:

```
BEGIN;
DCL P3  POINT2;
END;
SET P3 =POINT2  (1 CM, 5 CM);
```

In diesem Fall ist P3 außerhalb des BEGIN-Blockes nicht explizit vereinbart. Wie in PL/1 üblich würde P3 in der SET-Anweisung als DEC FLOAT(6) Variable implizit vereinbart angenommen. Dies führt zu einer Fehlermeldung des PL/1-Compilers von der Art "Conflicting attributes of argument ... to entry ...". Folgendes Beispiel dagegen führt nicht zu einer Fehlermeldung in der Übersetzungsphase, jedoch zu fehlerhafter Programmausführung:

```
DCL P4 POINTER;
SET P4 = POINT2 (2 CM, 0);
```

Ursache dafür, daß der Compiler diesen Fehler nicht bemerkt, ist, daß die GIPSY-Objekte nach ihrer Übersetzung aus der GIPSY-Sprache in PL/1 ebenfalls als POINTER behandelt werden.

Hinsichtlich der Vereinbarung (Deklaration) von GIPSY-Objekten (oder PL/1-Datentypen) ist zu beachten:

- Name
- Alignment
- Speicherklasse
- Scope
- Felder
- Strukturen
- Aggregate (Felder von Strukturen, Strukturen von Feldern).

Für GIPSY-Objekte gelten hier gewisse abweichende Regeln gegenüber PL/1-Daten.

Namen

GIPSY-Objekte dürfen fast beliebige Namen haben, sofern diese den Regeln von PL/1 entsprechen; jedoch gelten folgende Ausnahmen:

- Namen von GIPSY-Builtinfunktionen und von GIPSY-Objekttypen sind als Namen für GIPSY-Objekte verboten.
- Namen für GIPSY-Objekte dürfen höchstens 20 Zeichen lang sein.

Verboten sind also z. B.:
POINT2, COLL, BODY, CONE, ROTATE2, XAXIS, etc.
Die Verwendung dieser Namen in einer Deklaration, z. B.:.
"DCL CONE POINT2;", führt zu einer Fehlermeldung des Übersetzers.

Alignment

Die Attribute ALIGNED und UNALIGNED werden ignoriert.

Speicherklasse

Nicht erlaubt ist die Speicherklasse STATIC; erlaubte Speicherklassen sind

AUTOMATIC (Standardwert)
BASED
PARAMETER

Die Bedeutung ist wie in PL/1 üblich. Folgende Abweichungen gelten:

- CONTROLLED ist nur erlaubt, wenn das Programm ein MAIN-Modul wird, d. h. wenn es mit der Anweisung PROCEDURE OPTIONS(MAIN) beginnt, oder durch den Linkage Editor in einem Modul eingebunden wird, der ein solches MAIN-Programm enthält. Nicht erlaubt ist CONTROLLED in REGENT-Modulen anderer Subsysteme (siehe 7.4 und /¯1, Seite 233_/); dieser Fehler führt zum unkontrollierten Abbruch des Programms bei der Ausführung.
- Wenn das GIPSY-Objekt ein Parameter ist, so muß (im Gegensatz zu PL/1) PARAMETER in der Vereinbarung angegeben werden:

```
SUB: PROC(P,T,S);
DCL (P POINT, T TEXT2(3)) PARAMETER;
```

Zum Allokieren und Freigeben von BASED-Objekten dienen spezielle ALLOC- und FREE-Anweisungen, siehe Kap.6.9.

Scope

Das Scope-Attribut EXTERNAL ist nicht erlaubt.
Das Scope-Attribut INTERNAL ist erlaubt.

Felder und Strukturen

GIPSY-Objekte können als Felder vereinbart werden.

```
DCL P(*) POINT2 PARAMETER;
DCL T(-3:4) TEXT2(12);
DCL 1 STRUKTUR,
    2 NAMEN(50,N) CHAR (8),
    2 LINIE(10,M,N) POLY2(K);
DCL 1 FELD(*)PARAMETER,
    2 PP(2) POINT2,
    2 N BIN FIXED(15);
DCL 1 FELD(*),
    2 POINT(2),
    2 N BIN FIXED(15);
```

Es sind also Felder von Strukturen und Felder in Strukturen erlaubt. Ist die Struktur ein Parameter, so ist die Angabe PARAMETER erforderlich.

Fehler als Argumente und Parameter

Als Argument dürfen GIPSY-Felder mit oder ohne die Dimensionsangabe übergeben werden.
Gültig ist z. B.

```
DCL PUNKT(10,10,10) POINT2;
CALL SUB1(PUNKT);
CALL SUB1(PUNKT(*,*,*));
CALL SUB2(PUNKT(5,*,*));
CALL SUB2(PUNKT(*,*,3));
```

4.5 Zweidimensionale GIPSY-Objekte

Im zweidimensionalen Bereich stellt GIPSY folgende graphischen Objekte zur Verfügung:

POINT2
POLY2, CIRCLE2, ARC2
AXIS2,
TEXT2

Die 2D-Objekte, für die im dreidimensionalen Raum entsprechende Objekte zur Verfügung stehen, sind an den angehängten Zwei zu erkennen.
Zur Kennzeichnung der Lage von Punkten auf einer Zeichenebene werden besonders die POINT2-Objekte benutzt, denen je ein Abszissen- und ein Ordinatenwert zugewiesen wird. Die Linienobjekte CIRCLE2 und ARC2 dienen zum Zeichnen von Kreisen oder Kreisbögen und POLY2 zum Zeichnen von Linienzügen. Für die Aufbereitung von Diagrammen wurde das AXIS2-Objekt eingerichtet.

4.5.1 Deklaration von 2D-Objekten

Für graphische Objekte ist immer eine Deklaration des Datentyps erforderlich, sie sieht für 2D-Objekte folgendermaßen aus:

Syntax

DCL /¯level_/ ident /¯dim_/ type /¯storage class_/;

ident: Name nach PL/1-Regeln mit bis zu 20 Zeichen
dim : Dimension nach PL/1-Regeln

type: POINT2 | CIRCLE2 | ARC2 |
POLY2 (point-number) |
AXIS2 (char-number) |
TEXT2 (char-number)

point-number : maximale Anzahl der Punkte im Polygonzug

char-number : maximale Anzahl aufnehmbarer Zeichen (bei AXIS2 zur Achsenbeschriftung)

storage-class: AUTOMATIC|BASED|PARAMETER

level : Level innerhalb von PL/1-Strukturen

4.5.2 Eigenschaften der 2D-Objekte

Im folgenden werden die einzelnen Objekttypen im Detail charakterisiert.

POINT2

Ein POINT2-Objekt wird durch seine zwei Koordinatenwerte im 2D-Raum beschrieben.

POLY2

Ein POLY2-Objekt wird durch die geordnete Menge seiner 2D-Stützpunkte und die Punkteanzahl beschrieben.

CIRCLE2

Ein CIRCLE2-Objekt wird durch den Mittelpunkt (2D-POINT) und den Radius gekennzeichnet.

ARC2

Ein ARC2-Objekt wird durch einen Punkt (2D-POINT), einen Radius und 2 Winkel beschrieben.

AXIS2 unterstützt das Editieren von Kurven. AXIS2 deckt denjenigen Bereich, den die Darstellung von Wertefeldern aus Experimenten und Rechnungen in Diagrammen benötigt, ab.
Beschrieben wird das AXIS2-Objekt durch folgende Parameter:

Punkt, an dem die Achse beginnt (POINT2)
minimaler und maximaler Koordinatenwert
Achsenlänge
Achsenneigung
Abstand der Skalenstriche
Achsenbeschriftung und weitere Achseneigenschaften wie linear oder logarithmisch, glatte Skalenwerte oder Koordinatenextremwerte auf den Achsenenden,
Achsenbeschriftung rechts oder links der Achse.

TEXT2

Ein TEXT2-Objekt wird durch seine Gestalt und seine Lage in der Fläche beschrieben. Zur Festlegung der Gestalt dienen die Attribute Höhe H, Breite B und die Neigung (Abb.4.1).

Die Lage einer Zeichenkette in der Zeichenfläche wird durch die Lage des Referenzpunktes (untere linke Ecke des 1. Zeichens, bzw. Nullpunkt des Zeichenkettenkoordinatensystems, A1), einen Punkt auf der positiven x-Achse (z.B.B1) und einem beliebigen Punkt der Zeichenkettenebene mit $y > 0$ (z.B.C1) festgelegt.

Bezüglich der Transformationen wird ein Text wie ein Polygonzug $C_1\ A_1\ B_1$ behandelt. Die Breite ergibt sich aus dem Abstand A_1B_1, die Höhe aus dem Abstand $A_1\ C_1$ und die Neigung aus dem Winkel $B_1\ A_1\ C_1$.

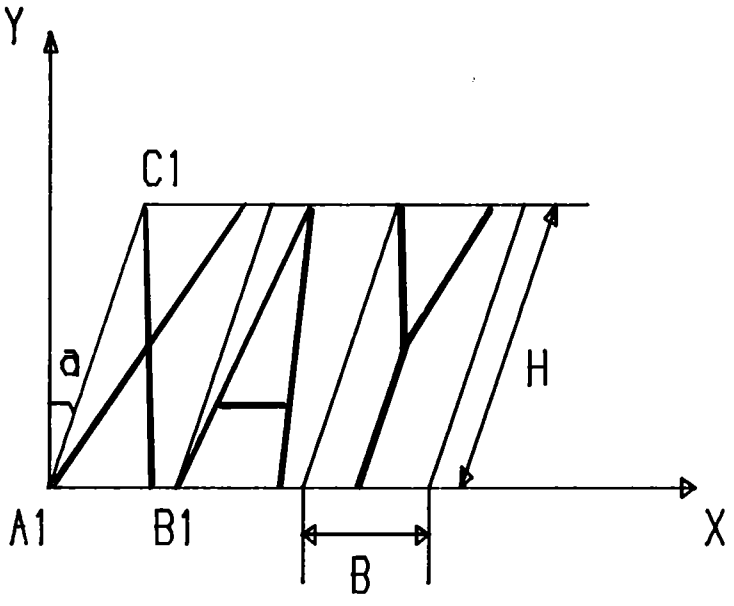


ABB.4.1: Zeichenkette und ihre Bestimmungsstuecke

4.6 GIPSY 3D-Objekte

GIPSY kennt im 3D-Raum 12 Objekttypen, die im folgenden in sechs Klassen eingeteilt sind:

(1)	punktförmige Typen	:	POINT
(2)	linienförmige Typen:		POLY, CIRCLE, TEXT, ARC
(3)	flächenhafte Typen	:	BALL, PLANE, CONE, CYLINDER
(4)	Raumelement	:	SPACE
(5)	Körper	:	BODY
(6)	Objektmengen	:	COLLECTION

Objekte vom Typ COLLECTION (siehe 4.7) können auch 2D-Elemente enthalten, sie sind in Abschnitt 4.5 näher erläutert.

4.6.1 Deklaration von 3D-Objekten

Für graphische Objekte ist immer eine Deklaration des Datentyps erforderlich, die für 3D-Objekte folgendermaßen aussieht:

Syntax

```
DCL /¯level_/ ident /¯dim_/ 3D-type /¯storage-class_/;

level    :  Level innerhalb von PL/1-Strukturen
ident    :  Name nach PL/1-Regeln
dim      :  Dimension nach PL/1-Regeln
3D-type  :  POINT | CIRCLE | PLANE | BALL | CYLINDER | CONE |
            POLYGON(point-number)
            | SPACE (surface-number)
            | BODY
point-number  :  maximale Anzahl der Punkte im Polygonzug
surface-number:  Anzahl der Flächen, die ein Raumelement defi-
                 nieren
storage-class :  AUTOMATIC | BASED | PARAMETER
```

4.6.2 Punktförmige und linienförmige 3D-Objekte

Diese 3D-Objekte unterscheiden sich nur hinsichtlich der dritten Raumkoordinate von den entsprechenden 2D-Objekten, insbesondere gelten für sie die gleichen Standardattribute und Referenzpunkte. Diese Objekte können bei geeigneter Wahl der Projektion unmittelbar gezeichnet werden.
Im folgenden werden die einzelnen Objekttypen im Detail charakterisiert.

POINT

Ein POINT-Objekt wird durch seine drei Koordinatenwerte im 3D-Raum beschrieben.

POLY

Ein POLY-Objekt wird durch die geordnete Menge seiner 3D-Stützpunkte und die Punkteanzahl beschrieben.

CIRCLE

Ein CIRCLE-Objekt wird durch den Mittelpunkt (3D-POINT), den Radius und die Lage der Kreisebene im 3D-Raum gekennzeichnet.

ARC

Ein ARC-Objekt wird durch drei Punkte beschrieben, von denen der erste und der letzte den Anfangs- und Endpunkt beschreiben. Alle drei Punkte definieren eine Ebene im Raum und auf dieser Ebene einen Kreis, von dem der Kreisbogen einen Abschnitt darstellt.

TEXT

Ein TEXT-Objekt wird durch seine Gestalt und seine Lage im Raum beschrieben. Zur Festlegung der Gestalt dienen die Attribute Höhe H, Breite B, und die Neigung α (Abb.4.1)
Die dargestellte Zeichenkette liegt in der x-y-Ebene (Zeichenkettenebene).

Die Lage einer Zeichenkette im Raum wird durch die Lage des Referenzpunktes (untere linke Ecke des 1.Zeichens, bzw. Nullpunkt des Zeichenkettenkoordinatensystems, A_1), einen Punkt auf der positiven x-Achse (z.B. B_1) und einem beliebigen Punkt der Zeichenkettenebene mit $y > 0$ (z.B.C_1) festgelegt.

Bezüglich der Transformationen wird ein Text wie ein Polygonzug $C_1\ A_1\ B_1$ behandelt. Das bedeutet, daß die drei Punkte im Raum transformiert werden und daß sich die Breite aus dem Abstand $A_1\ B_1$, die Höhe aus dem Abstand $A_1\ C_1$ und die Neigung aus dem Winkel $B_1\ A_1\ C_1$ ergibt.

4.6.3 Flächenhafte 3D-Objekte

GIPSY-Flächen nehmen eine Sonderstellung ein: sie können nicht gezeichnet werden, sondern dienen lediglich der Definition von anderen Objekten (Raumelemente, Linien, Punkte).
Die GIPSY-Flächen sind Flächen im mathematischen Sinne, d.h. insbesondere, daß sie unendlich ausgedehnt sind (mit Ausnahme der Kugel). Sie teilen den 3D-Raum in zwei Halbräume. Diese Halbräume werden unterschieden durch die Richtung der Flächennormalen. Derjenige Halbraum, in den die Flächennormale zeigt, wird als positiv bezeichnet.

GIPSY kennt die Flächen: Ebene, Kugel, Zylinder, Kegel.

PLANE

Eine GIPSY-Ebene wird durch einen Ebenenpunkt (Normalenfußpunkt) und die Normalenrichtung festgelegt. Sie ist unendlich ausgedehnt. Der Ebenenreferenzpunkt ist der Fußpunkt der Normalen.

BALL

Eine GIPSY-Kugel wird durch ihren Mittelpunkt und den Radius beschrieben, ihr Referenzpunkt ist der Mittelpunkt. Die Normale zeigt zum Mittelpunkt.

CYLINDER

Ein Zylinder wird durch den Radius und seine Achse festgelegt. Er ist unendlich lang. Die Normale zeigt zur Zylinderachse. Referenzpunkt ist der bei der Erzeugung angegebene Punkt auf der Achse.

CONE

Ein GIPSY-Kegel ist ein unendlich ausgedehnter doppelter gerader Kreiskegel. Seine Spitze ist der Referenzpunkt. Er wird charakterisiert durch die Lage der Spitze, die Achsenrichtung und den Öffnungswinkel.
Die Normale zeigt zur Achse.

Raumelemente

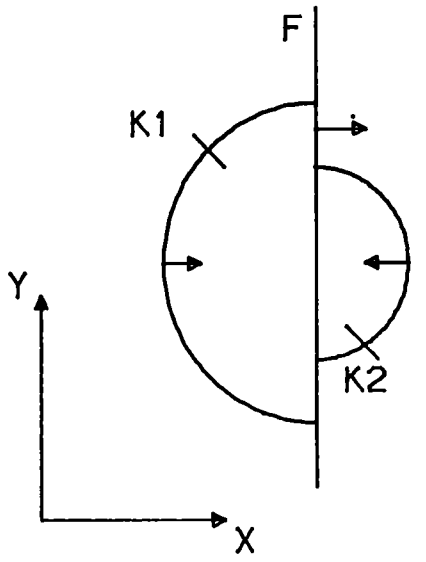
Ein GIPSY-Raumelement (SPACE) ist eine Menge von GIPSY-Flächen, die einen endlichen konvexen Raumbereich einhüllen. Die Normalen dieser Flächen zeigen in das Raumelement hinein.
Ein Raumelement wird beim Zeichnen (falls gewünscht) als undurchsichtig behandelt.

Durch die Verbindung unendlich ausgedehnter Flächen zur Beschreibung von Raumelementen ist es unmöglich, jeden beliebigen Körper (mit Hüllflächen der Typen Ebene, Kugel, Kegel, Zylinder) zu beschreiben, insbesondere die Beschreibung von konkaven Raumelementen mit ebener Begrenzung ist ausgeschlossen. Lediglich konkave Elemente mit kugeligen, zylindrischen und kegeligen Aushöhlungen sind beschreibbar.

Zur Beschreibung beliebiger Körper mit ebenen, kugeligen, zylindrischen und kegeligen Oberflächen dient der GIPSY-Körper (BODY).

Körper

Ein GIPSY-Körper (BODY) ist eine Menge aneinandergefügter ("verschweißter") Raumelemente (SPACES). Zwei Raumelemente sind in diesem Sinne zusammengefügt, wenn sie beide mindestens eine gemeinsame Fläche (mit entgegengesetzter Normale) enthalten.(Abb.4.2).




```
DCL (K1,K2) BALL;
DCL F PLANE;
DCL B BODY;
DCL (HK1,HK2) SPACE(2);
   Erzeugen der Flaechen
BUILD HK1-SPACE(K1-F);
BUILD HK2-SPACE(K2+F);
BUILD B-BODY(HK1,HK2);
```

ABB.4.2: Beispiel eines GIPSY-Koerpers

4.7 Kollektion (COLLECTION)

Die Kollektion ist ein besonderes Objekt, das sowohl 2D- wie auch 3D-Objekte enthalten kann. Eine Kollektion stellt eine geordnete Menge anderer Objekte dar.

Mit wenigen Ausnahmen (z.B. POLYGON) gilt für Kollektionen die Regel:

Eine Funktion, angewandt auf eine Kollektion, liefert als Ergebnis eine Kollektion, welche als Elemente die Ergebnisse der Anwendung der Funktion auf die Elemente der Kollektion enthält. Entsprechendes gilt für die Operationen PLOT, EMPTY etc.

4.8 Die Lebensdauer von GIPSY-Objekten

Für den mit PL/1 besser vertrauten GIPSY-Anwender sei an die Lebensdauer von PL/1-Daten erinnert.

- Daten der Speicherklasse AUTOMATIC existieren nur innerhalb des Blockes (PROCEDURE oder BEGIN) in dem sie vereinbart sind.
- Daten der Speicherklasse BASED existieren unter Umständen länger. Sie werden durch eine ALLOCATE-Anweisung erzeugt und müssen durch eine FREE-Anweisung explizit beseitigt werden. Zu ALLOCATE und FREE siehe Kap.6.9.

In der GIPSY-Implementierung werden die GIPSY-Objekte intern durch PL/1-Daten dargestellt. In den meisten Fällen handelt es sich um Daten der Speicherklasse AUTOMATIC; der Anwender braucht sich in allen diesen Fällen nicht um die Lebensdauer der Objekte zu kümmern. Ausnahmen sind jedoch die Objekte vom Typ

COLLECTION
SPACE
BODY

Hier wird GIPSY-intern die Speicherklasse BASED benutzt, da diese Objekte in ihrem Aufbau nachträglich verändert werden können. Bei diesen Objekten muß der Anwender sicherstellen, daß sie beseitigt werden. Dazu dient die EMPTY-Anweisung. Weitere Erläuterungen werden in Kap.6.2.5 gegeben. An dieser Stelle soll nur auf die Punkte eingegangen werden, die für den Anwender besonders wichtig sind.

COLLECTION

Eine COLLECTION ist eine(geordnete)Menge anderer GIPSY-Objekte. Sie kann durch Hinzufügen weiterer Objekte vergrößert werden und durch EMPTY geleert werden. Sie muß durch EMPTY geleert werden, bevor der Block verlassen wird, in dem die Collektion deklariert wurde (bei AUTOMATIC) oder bevor sie durch FREE beseitigt wird (bei BASED).

Beispiel

```
DO J = 1 TO 1000;
BEGIN;
DCL C1 COLLECTION;
DCL C2 COLLECTION BASED;
ALLOCATE COLLECTION C2;
SET C1 = COLLECTION(----);
SET C2 = INTERSECTION(----);
--------
EMPTY(C2);
FREE COLLECTION C2;
EMPTY (C1);
END;
END /* OF LOOP */;
```

Würden in diesem Beispiel die EMPTY-Anweisungen weggelassen, so könnte dies dazu führen, daß das Programm nach einigen Schleifen abbricht, da der Arbeitspeicher der Rechenanlage nicht mehr ausreicht. Selbst wenn es nicht zum Abbruch kommt, wäre eine unnötige Verschwendung von Speicherkapazität die Folge.

SPACE und BODY

Das oben für COLLECTION erläuterte gilt in gleicher Weise für SPACE und BODY.

Aufbau von Raumelementen mit BUILD

Die BUILD-Anweisungen unterscheiden sich von den Zuweisungen (SET-Anweisungen) darin, daß sie ein Objekt erzeugen (ein SPACE-Objekt oder ein BODY-Objekt), welches Zeiger auf die Objekte enthält, die in der Operation (SPACE oder BODY) als Argument angegeben werden. Diese Eigenheit der BUILD-Anweisung hat zur Folge, daß die Lebensdauer von Objekten des Typs SPACE und des Typs BODY (die SPACE enthalten) besonders wichtig ist. Der Anwender muß sicherstellen, daß zum Zeitpunkt der Verwendung eines Spaceobjektes oder Bodyobjektes alle Oberflächenobjekte noch existieren, die zur Definition benutzt werden.

Falsch ist beispielsweise

```
DCL S SPACE(1);
BEGIN;
  DCL B BALL;
  SET B = BALL(POINT(0,0,0),10.);
  BUILD S = SPACE(BALL);
END;
PLOT(S);
```

Ein Fehler dieser Art führt bei der Verwendung von S in der Plotanweisung zu einer undefinierten Fehlerreaktion (üblicherweise Addressierungsfehler, Abbruch mit entsprechender Meldung des Betriebssystems). Wegen dieses unvorhersehbaren Fehlerverhaltens ist im Zusammenhang mit der Verwendung von SPACE besondere Aufmerksamkeit geboten.

5. GIPSY-Builtin-Funktionen

5.1 Erzeugung von Objekten

Den graphischen Objekten werden durch die Anweisungen SET und BUILD Werte zugewiesen. Zur Erzeugung von Elementwerten, die den Objekten zugewiesen werden können, dienen die GIPSY-BUILTIN-Funktionen. Als Ergebnis ihres Aufrufes liefern sie ein graphisches GIPSY-Objekt zurück.

Die Argumente der GIPSY-Builtin-Funktionen können sein

(1) GIPSY-Objekte (Kap.4). Sie müssen deklariert sein und zuvor muß ihnen ein Wert zugewiesen worden sein

(2) GIPSY-Builtin-Funktionen

(3) Numerische Ausdrücke, je nach Funktion Real-, Integer-, Bit- oder Characterwerte

(4) Längen- oder Winkelausdrücke, die aus einem numerischen Ausdruck und einer Maßeinheit bestehen, siehe Kap.4.2

Die einzelnen Argumente werden in den folgenden Abschnitten erläutert und im Handbuch-Teil (Kap.9) im Detail beschrieben. Zur Kennzeichnung werden dabei folgende Abkürzungen verwendet:

lvalue : Längenangabe PL/1-Ausdruck mit einer optionalen Längeneinheit,

avalue : Winkelangabe. PL/1-Ausdruck mit einer optionalen Winkeleinheit,

g_obj : GIPSY-Objekt, das namentlich angegeben wird

g_func : GIPSY-Builtin-Funktion

g_ex : GIPSY-Ausdruck, = g_obj | g_func

pl1_ex : PL/1-Ausdruck

Es gilt außerdem folgende Regel:

Wenn eine GIPSY-Builtin-Funktion ein GIPSY-Ausdruck g_ex mit einem bestimmten Objekttyp ≠ COLLECTION erwartet und stattdessen eine Kollektion angeliefert wird, so wird nachgeprüft, ob das

erste Element der Kollektion vorhanden und vom richtigen Typ ist. In diesem Fall wird dieses erste Kollektions-Element ohne Fehlermeldung als Argument akzeptiert.

5.1.1 Builtin-Funktionen zur Erzeugung v. 2D-Objekten

Für die Erzeugung von 2D-Objekten stehen folgende GIPSY-Builtin-funktionen zur Verfügung:

Builtin-Funktion	Ergebnisobjekt
POINT2,REFPOINT2, NPOINT2	POINT2
POLY2, NPOLY2, CURVE2	POLY2
CIRCLE2	CIRCLE2
ARC2	ARC2
AXIS2,XAXIS, YAXIS	AXIS2
TEXT2	TEXT2
INTERSECTION, ISO2, COLLECTION	COLLECTION

Mit Hilfe dieser Funktionen können graphischen Variablen graphische Attribute zugewiesen werden. Dies geschieht in der graphischen Zuweisung, die mit dem Schlüsselwort SET beginnt und vollständig lautet:

```
SET 2D-elementvariable = 2D-builtin-funktion
```

Beispiel:

```
DCL PU POINT2;
SET PU = POINT2(50 MM, 50 MM);
```

Damit wird also dem Punkt PU ein geometrischer Wert zugewiesen.

Im folgenden werden die 2D-Funktionen näher erläutert.
POINT2 erwartet zwei Längenangaben und legt damit einen Punkt in der 2D-Ebene fest.

REFPOINT2 extrahiert den Referenzpunkt eines graphischen Objektes (siehe Kap.4.3.1) und erzeugt damit ein Punktobjekt.
REPOINT2 erwartet also ein 2D-Objekt, zugelassen ist auch die Angabe einer Kollektion. Als Referenzpunkte sind definiert:

POINT2	der Punkt selbst
POLY2	der erste Punkt des Polygonzuges
CIRCLE2	der Mittelpunkt des Kreises
ARC2	der Mittelpunkt des Kreisbogens
AXIS2	der Anfangspunkt der Achse
TEXT2	die linke untere Ecke des 1. Zeichens
COLLECTION	der Referenzpunkt des ersten Objektes in der Kollektion

NPOINT2 extrahiert aus einem Polygonzug die Koordinaten des n-ten Punktes für die Erzeugung eines Punktobjektes.

POLY2 erzeugt aus einer Kollektion ein Polygonzugobjekt.
Die Kollektion darf 2D-Punkte, Polygone, Kreise und Bögen enthalten.

Beispiel:

```
DCL VIERECK POLY2(5);
SET VIERECK=POLY2(COLLECTION(P1,P2,P3,P4,P1)); oder abgekürzt
SET VIERECK=POLY2(P1,P2,P3,P4,P1);
```

NPOLY2 kann aus einem Polygonzug einen Teilbereich herauskopieren. Der erste Punkt und die Länge des Teilbereiches muß dazu angegeben werden.

CURVE2 erzeugt aus einem Wertefeld $x_i,y_i,i=1....n$ einen Polygonzug, so daß die Stützpunkte des Polygonzuges die Kurve y=f(x) in einem durch zwei gegebene Achsen aufgespannten Koordinatensystem darstellen. Die Funktion benötigt also als Parameter zwei Achsen (AXIS2) und ein Wertefeld (x_i,y_i). Zusammen mit den Funktionen XAXIS und YAXIS oder AXIS2 lassen sich damit in einfacher Weise Diagramme darstellen.

ISO2 erzeugt aus einem Wertefeld x_i,y_i,z_i,zwei Achsen und einem Wert h eine Kollektion von Polygonzügen (POLY2), die die Höhenlinien mit z=h darstellen, wenn die Werte x, y, z den funktionalen Zusammenhang z=f(x,y) darstellen. Die beiden Achsen dienen zur Transformation der Höhenlinien in das durch die Achsen aufgespannte Koordinatensystem.

CIRCLE2 legt die geometrischen Daten, Mittelpunkt und Radius eines Kreises fest. Für die Festlegung dieser Daten gibt es folgende Eingabemöglichkeiten:

1) (point2, lvalue) für Mittelpunkt, Radius,
2) (point2, point2) für Mittelpunkt, Peripheriepunkt,
3) (point2, point2, point2, text)

Der Kreis wird definiert durch drei Eckpunkte eines Dreiecks. Im Text kann durch 'IN' oder 'OUT' der Inkreis oder Umkreis zu diesem Dreieck verlangt werden.

ARC2 legt einen Kreisbogen fest.Folgende Angaben sind möglich:

1) Mittelpunkt, Winkel zu Anfangs- und Endpunkt, Radius;
2) Zwei Punkte, Radius, Angabe ob großer oder kleiner Bogen;
3) drei Punkte und
4) zwei Punkte und Bogenlänge.

AXIS2, XAXIS, YAXIS sind Funktionen zur Definition von Koordinatenachsen z.B. bei Diagrammen. Für die allgemeine AXIS2-Funktion können folgende Parameter angegeben werden:
Anfangspunkt, minimaler und maximaler Koordinatenwert, Länge, Winkel, Achsenstrichabstand, einen die Achse erläuternden Text und Optionen zur Festlegung von weiteren Achseneigenschaften wie z.B. linear oder logarithmisch. XAXIS und YAXIS liefern zu einem Wertefeld x_i, y_i, i=1,...n eine passende Standard-Achse, in waagerechter (XAXIS) und senkrechter (YAXIS) Richtung.

TEXT2 wird benutzt zur Erzeugung von Textobjekten. Hier kann eine Zeichenkette zusammen mit einer Anfangskoordinate (1.Zeichen links unten) und einem Winkel,unter dem dieser Text gezeichnet werden soll, zugewiesen werden.

5.1.2 Builtin-Funktionen zur Erzeugung von 3D-Objekten

Für die Erzeugung von 3D-Objekten aus arithmetischen und/oder anderen graphischen Objekten stehen folgende Builtin-Funktionen zur Verfügung:

Builtin - Funktion	Ergebnisobjekttyp
POINT, REFPOINT, NPOINT	POINT
CIRCLE	CIRCLE
ARC	ARC
POLYGON, NPOLYGON	POLYGON
TEXT	TEXT
PLANE	PLANE
BALL	BALL
CYLINDER	CYLINDER
CONE	CONE
SPACE	SPACE
BODY	BODY
ISO, INTERSECTION, COLLECTION	COLLECTION

Eine einfache, elementare Zuweisung zur Erzeugung eines 3D-Objektes sieht folgendermaßen aus:

SET 3D-elementvariable = 3D-builtin-funktion;

Beispiel:

```
DCL C CONE;
SET C=CONE(....);
```

Zum Erzeugen eines 3D-Punktes stehen die Funktionen POINT, REFPOINT und NPOINT zur Verfügung.

POINT erzeugt ein Punktobjekt aus drei Koordinatenwerten.

REFPOINT extrahiert aus einem Polygon, einem Kreis, einer Ebene, einer Kugel, einem Zylinder oder einem Kegel die jeweiligen Referenzpunktkoordinaten und erzeugt damit ein Punktobjekt.
NPOINT extrahiert aus einem Polygonzug die Koordinaten des n-ten Punktes für die Erzeugung eines Punktes.

Die Erzeugung eines Kreises erfolgt mit CIRCLE unter Angabe folgender geometrischer Größen:

1) Mittelpunkt, Achsenpunkt, Radius oder
2) Mittelpunkt, Achsrichtung, Radius oder
3) Achsenpolygonzug(die ersten beiden Punkte), Radius

Ein Bogen im 3D-Raum wird mit der ARC-Funktion erzeugt. Anfangs- und Endpunkt sowie ein Punkt auf dem Bogen zwischen Anfangs- und Endpunkt müssen angegeben werden.
Ein Polygonzug wird durch POLY erzeugt.
POLYGON erwartet als Parameter eine Kollektion.
Die Erzeugung eines Textes erfolgt mit der Funktion TEXT, deren Parameter die räumliche Lage des Textes und die Zeichenkette beschreiben. Die Gestalt des Textes kann später mit EDIT TEXT festgelegt werden, aus Standardannahmen abgeleitet sein, oder auch direkt in der TEXT-Funktion angegeben werden. Zur Gestalt gehören Buchstabenhöhe und -breite und die Neigung der Buchstaben.

INTERSECTION (oder INT) liefert eine Kollektion von Punkten, die sich als Durchstoßpunkt einer Geraden durch eine Fläche ergeben. Die Gerade wird aus der ersten Teilstrecke eines Polygonzuges (POLY) gebildet, die Flächen sind alle in GIPSY enthaltenen Flächen, so daß als Ergebnis ein Punkt (Ebene, Kegel) oder zwei Punkte (Kegel, Kugel, Zylinder) zurückgeliefert werden. Mit Hilfe der Funktionen zur Behandlung von Kollektionen können die Punkte extrahiert werden (siehe nächsten Abschnitt 5.1.3).

ISO liefert zu einem Wertefeld x_i, y_i, z_i mit $z=f(x,y)$ die zu einer Höhe h gehörige Kollektion von Höhenlinien (POLY).

Der Beschreibung von GIPSY-Raumelementen bzw. Körpern dienen die GIPSY-Flächen Ebene, Kugel, Zylinder, Kegel. Sie werden durch die Funktionen PLANE, BALL, CYLINDER, CONE erzeugt.

PLANE erzeugt eine Ebene durch

1) einen Punkt in der Ebene (Normalenfußpunkt) und einen zweiten Punkt auf der Normalen. Die Normalenrichtung ist vom Fußpunkt zum zweiten Normalpunkt.

2) drei nicht auf einer Linie liegende Punkte in der Ebene. Die Normalenrichtung ist mathematisch positiv bezüglich der drei Punkte (Rechtsschraube)

3) einen Punkt in der Ebene und die Normalenrichtung

4) einen Polygonzug, dessen erster Punkt als Normalenfußpunkt und dessen zweiter Punkt als Normalenpunkt dient.

BALL erzeugt eine Kugel aus dem Mittelpunkt und dem Radius. Die Normalenrichtung ist positiv zum Mittelpunkt der Kugel bei positivem Radius, nach außen bei negativem Radius.

CYLINDER erzeugt einen unendlich ausgedehnten Zylinder aus

1) zwei Achsenpunkten und dem Radius
2) einem Punkt auf der Achse, der Achsenrichtung und dem Radius. In beiden Fällen legt das Vorzeichen des Radius die Normalenrichtung fest. Sie zeigt zur Zylinderachse bei positivem Radius
3) aus einem Polygonzug, dessen beiden ersten Punkte die Achse festlegen,und dem Radius des halben Öffnungswinkels.

GIPSY-Raumelemente werden mit Hilfe der SPACE-Funktion aus den einhüllenden Flächen erzeugt.

Mit Hilfe der BODY-Funktion werden Raumelemente zu komplexeren Körpern zusammengefügt, indem jeweils zwei Raumelemente an einer ihnen gemeinsamen (identischen) Fläche "zusammengeschweißt" werden.

5.1.3 Erzeugen von Kollektionen

Zwei Funktionen liefern Kollektionen als Ergebnis :-

COLLECTION
INTERSECTION

COLLECTION

Die COLLECTION-Funktion kann im Gegensatz zu anderen Funktionen eine von Fall zu Fall verschiedene Anzahl von Argumenten haben. Maximal 60 Argumente sind erlaubt. Die als Argumente angegebenen Objekte werden in der vorliegenden Reihenfolge zu einer geordneten Menge (Kollektion) zusammengefaßt. Ist ein Argument selbst eine Kollektion, so werden deren Elemente in ihrer Reihenfolge übernommen. In jedem Fall enthält die entstehende Kollektion Kopien der Argumente, nicht diese selbst. (Siehe auch Kap.5.1.2 (BODY) und 6.2). Besondere Funktionen zum Behandeln von Kollektionen sind in Kap. 5.3.1 beschrieben.

INTERSECTION

Das Ergebnis der Schnittpunktsfunktion INTERSECTION ist aus folgender Aufstellung zu entnehmen.

Schnittpunkte im 2D-Bereich

POLY2 - POLY2

Ergebniskollektion: die Schnittpunkte der einzelnen Strecken der Polygonzüge.(Leer falls kein Schnittpunkt existiert)
Geordnet nach den Teilstrecken des ersten Polygonzuges

POLY2 - CIRCLE2

Ergebniskollektion: entweder leer oder die Schnittpunkte der Strecken des Polygonzuges mit dem Kreis geordnet nach den Teilstrecken des Polygonzuges

CIRCLE2 - CIRCLE2

Ergebniskollektion: entweder leer oder die Schnittpunkte geordnet nach wachsender x-Koordinate (oder y falls x gleich)

CIRCLE2 - ARC2
ARC2 - ARC2
POLY2 - ARC2

Entweder leer oder die existierenden Schnittpunkte

Schnittpunkte im 3D-Bereich

POLY - SPACE

Ergebniskollektion: entweder leer oder die zwei Schnittpunkte des verlängerten ersten Polygonabschnittes mit der Oberfläche des SPACE-Objektes

POLY - BODY

Ergebniskollektion: entweder leer oder die gerade Anzahl von Schnittpunkten des (verlängerten) ersten Polygonabschnittes mit den Oberflächen des BODY-Objektes, geordnet in Richtung des Polygonabschnittes

POLY - PLANE
POLY - BALL
POLY - CYLINDER
POLY - CONE

entweder leer oder der Schnittpunkt des (verlängerten) ersten Polygonabschnittes mit der Ebene (bei PLANE) bzw. die zwei Schnittpunkte mit den Flächen, geordnet in Richtung des Polygonabschnittes.

5.2 Funktionen zur Objekttransformation

5.2.1 2D-Transformationen

Die graphischen Transformationen dienen der Veränderung eines Objektes. Folgende Funktionen stehen zur Verfügung:

SHIFT2
MOVE
MOVETO
SCALE2
ROTATE2

SHIFT2 verschiebt ein Objekt parallel zur Abszisse und/oder zur Ordinate. Als Argumente werden erwartet ein graphisches Objekt oder eine Funktion (g_ex) und zwei Längenangaben (l_value) für die Verschiebungen. Die Erfahrung hat gezeigt, daß SHIFT2, wegen seiner einfachen Anwendbarkeit, häufiger als die anderen Verschiebefunktionen benutzt wird. Z.B. kann man einen Polygonzug, der sich um den geometrischen Ursprung zieht in das positive Zeichnungsfeld folgendermaßen transformieren:

```
PLOT (SHIFT2(POLZUG,100MM,100MM))
```

In diesem Fall bleiben die ursprünglichen Koordinaten von POLZUG erhalten, auf der Zeichnung erscheint er jedoch um je 100MM in positiver Richtung verschoben.

MOVE bewirkt ähnlich wie SHIFT2 eine Verschiebung. Hier werden jedoch die Verschiebungswerte aus einem Punktobjekt entnommen. In der Argumentliste wird ein graphisches Objekt (g_ex) erwartet,welches verschoben werden soll und ein GIPSY 2D-Punkt, der die Verschiebungswerte enthält.

Beispiel:

```
DCL (P1,PV,P2) POINT2;
SET P1=POINT2(50MM,20MM);
SET PV=POINT2(10MM,10MM);
SET P2=MOVE(P1,PV);
```

Diese letzte Anweisung bewirkt dasselbe wie

```
SET P2=POINT2(60MM,30MM);  oder
SET P2=SHIFT2(P1,10 MM, 10 MM);
```

In diesem Beispiel wurde P2 als Wert eine Kopie der Verschiebung von P1 zugewiesen, eine echte Verschiebung von P1 ergibt sich durch die Anweisung

```
SET P1=MOVE(P1,PV);
```

MOVETO erwartet als zweites Argument wiederum ein Punktobjekt, das jedoch im Gegensatz zur MOVE-Operation nicht als Verschiebevektor angesehen wird, sondern als Ort, zu dem das im 1.Argument bezeichnete Objekt verschoben werden soll. Das bedeutet, daß der Referenzpunkt (Kap.4.3.1) des ersten Argumentes auf den zweiten Punkt zu liegen kommt.

SCALE2 skaliert die Ordinatenwerte eines Objektes bezüglich eines Skalierungsreferenzpunktes. Es werden 4 Argumente erwartet: der Namen eines Objektes, 2 Werte für Skalierung in x- und y-Richtung, sowie ein Punktobjekt. Es gilt

$$x_i' = (x_i - x_{Ri}) \cdot faktor_i + x_{Ri}$$

wobei $x_1 = x$, $x_2 = y$ = Objektkoordinaten

x_{Ri} = Referenzkoordinaten

x_i' = neue Koordinate

$faktor_i$ = Skalierungsfaktor

Die Angabe des Referenzpunktes kann fehlen. Dann bezieht sich die Skalierung auf den Koordinatenursprung $P_R(0,0)$.

ROTATE2 bewirkt die Drehung eines graphischen Objektes um einen Bezugspunkt. Es erwartet die Angabe eines zu drehenden Objektes, eines Winkels und eines Punktobjektes, um dessen Koordinatenwert gedreht wird. Unterbleibt die Angabe dieses Punktes, wird um den Koordinatenursprung gedreht.

5.2.2 Transformationen für 3D-Objekte

Die graphischen Transformationen dienen der Veränderung der Lage eines Objektes. Folgende Operationen stehen zur Verfügung: SHIFT, SCALE, ROTATE, MOVE, MOVETO.

SHIFT verschiebt ein 3D-Objekt im Raum um die Werte Δx, Δy, Δz.

MOVE verschiebt ein 3D-Objekt im Raum um den Vektor P.

MOVETO verschiebt den Referenzpunkt eines 3D-Objektes an einen spezifischen Punkt.

SCALE skaliert die Objektkoordinaten mit spezifizierten Faktoren bezüglich eines Referenzpunktes. Es gilt für i=1 bis 3:

$$x_i' = (x_i - x_{Ri}) = faktor_i + x_{Ri}$$

wobei x_i : Objektkoordinate
x_{Ri} : Referenzpunktkoordinate
x_i' : neue Koordinate
$faktor_i$: Skalierungsfaktor

Der Standard-Referenzpunkt ist der Koordinatenursprung $(x_{Ri}=0)_{i=1...3}$

ROTATE dreht die Objektpunkte um den Koordinatenursprung oder einen angegebenen Fixpunkt.

5.2.3 Transformation 3D-2D

Dreidimensionale Objekte müssen vor ihrer Darstellung in der Zeichenebene auf eine Projektionsebene projiziert werden. Diese Operation wird in GIPSY REDUCE-Operation genannt (Reduktion der Dimensionalität).
Diese Operation

- geschieht explizit durch Anwenden der REDUCE-Funktion, welche ein zweidimensionales Objekt erzeugt, oder

- implizit in der PLOT-Anwendung, die das Ergebnis der Operation sofort ausgibt.

Folgende Punkte sind bei der REDUCE-Operation zu beachten:

- Projektionsart
- Projektionsrichtung oder Projektionszentrum
- Projektionsebene
- Sichtbarkeit
- Lage und Orientierung des projizierten Bildes mit Bezug auf die Zeichenebene.

Die Erfahrungen mit GIPSY zeigen, daß der Anwender sich mit diesen Aspekten wirklich vertraut machen muß, wenn er ein dreidimensionales Objekt in gezielter Weise darstellen will. Ein unvollständiges Verständnis der REDUCE-Operation führt regelmäßig zu mehr oder weniger erfolgreichem Probieren.

Die REDUCE-Operation wird durch den Befehl CHANGE PROJECTION festgelegt.

Projektionsart

GIPSY kennt die Projektionsarten CENTRAL und PARALLEL. Standardwert ist PARALLEL.

Projektionsrichtung oder Projektionszentrum

Stets gemeinsam mit der Projektionsart werden in GIPSY drei Koordinaten festgelegt. Es sind dies

- bei Zentralprojekten die Koordinaten des Punktes, von dem aus projiziert wird
- bei Parallelprojektion die Koordinaten eines Punktes, der auf demselben Projektionsstrahl liegt wie der 3D-Koordinatenursprung. Als Projektionszentrum gilt in diesem Fall ein Punkt, der vom Koordinatenursprung gesehen in Richtung auf den angegebenen Punkt sehr weit entfernt ist.

Projektionsebene

Die Projektionsebene liegt beliebig im Raum. Sie ist definiert durch einen in ihr liegenden Punkt PI_POINT und den Normalenvektor (siehe Abb.5.1). Betrachtet wird die Projektionsebene stets aus dem Halbraum, in den die Flächennormale zeigt. Eine Umkehrung der Flächennormale bewirkt in dem projizierten Bild eine Vertauschung von "links" und "rechts" (siehe Abb.5.2 und 5.3).

Sichtbarkeit

Wenn unsichtbare Körperkanten bei Raumelementen (SPACE) und Körpern (BODY) unterdrückt werden, so werden nur diejenigen Kanten in das erzeugte 2D-Objekt übernommen, die vom Projektionszentrum aus gesehen nicht durch Raumelement- oder Körpermaterial verdeckt sind. 3D-Punkt- und Linienobjekte sind stets sichtbar, sie beeinflussen die Sichtbarkeit von anderen Objekten nicht und werden nicht durch andere Objekte verdeckt. Anstelle einer Unterdrückung der unsichtbaren Kanten kann der Anwender auch ihre Darstellung in einer bestimmten Strichart wählen. Achtung: Auch Objekte, die nicht zwischen Projektionszentrum und Projektionsebene liegen, sind möglicherweise sichtbar.

Lage und Orientierung des Bildes

Lage und Orientierung des Bildes in der Zeichenfläche ergeben sich aus den zwei Operationen

- Projektion des 3D-Objektes in die 2D-Ebene von GIPSY
- Abbildung dieser 2D-Ebene auf die Zeichenfläche.

In Kap. 8.2 und 8.4 werden "Kochrezepte" dafür angegeben, wie man diese Operationen zu spezifizieren hat, wenn man bestimmte, oft vorkommende Darstellungen erzeugen will. Hier soll vor allem der Projektionsvorgang beschrieben werden.
Alle 3D-Objekte sind im 3D-Koordinatensystem von GIPSY(mit den Achsen x_1, x_2, x_3) beschrieben. Für die Vorstellung ist es zweckmäßig, die positive x_3-Achse als "nach oben" gerichtet anzusehen. Durch die Projektion in die Projektionsebene entstehen 2D-Objekte, die in einem in der Projektionsebene liegenden 2D-Koordinatensystem definiert sind. Die Achsen dieses 2D-Koordinatensystems ergeben sich aus der Projektion der Achsen des 3D-Koordinatensystems. Für das Verständnis der Projektion in GIPSY ist es unumgänglich notwendig, die Entstehung dieses 2D-Koordinatensystems zu begreifen. Dieses 2D-Koordinatensystem entsteht nach folgenden Regeln.

Normalfall: Projektion auf eine Ebene nicht senkrecht zur x_3-Achse (siehe hierzu Abb.5.1)

1) Die Schnittlinien der Grundebene (x_1-x_2-Ebene) mit der Projektionsebene ergibt die Orientierung der x_1'-Achse in der Projektionsebene. In Abb.5.1 ist dies die Linie A_1-A_2. Damit ist noch nicht festgelegt, ob die positive x_1'-Richtung von A_1 nach A_2 oder umgekehrt zeigt.

2) Die Fallinie in der Projektionsebene, d.h. das Lot auf die Linie A_1-A_2 gibt die Orientierung der x_2'-Achse in der Projektionsebene. In Abb.5.1 ist dies die Linie A_3-L. Die positive x_2'-Richtung ist stets zur x_3-Achse hin. Grob gesprochen: Oben bleibt oben.

3) Die Positive x_2'-Richtung, der Normalenvektor der Ebene (in Abb.5.1 der Vektor $\underline{e}$) und die x_1'-Richtung bilden ein positives Dreibein, wie in Abb.5.1 in Punkt L gezeigt.

4) Nachdem die x_1'= und x_2'-Richtung festliegen, wird der Nullpunkt der x_1'= und x_2'-Achsen bestimmt. Die Projektion des

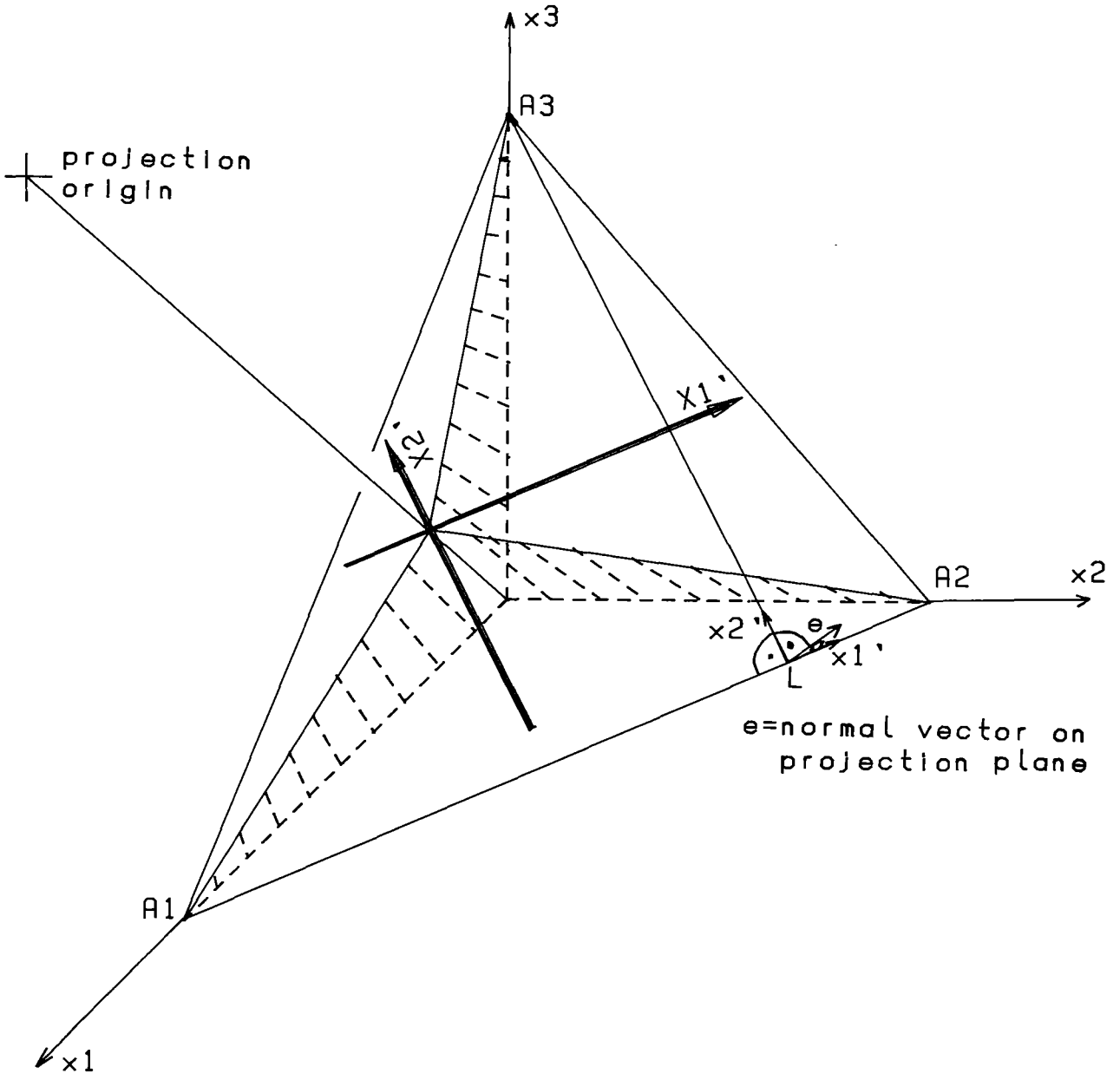


Abb. 5.1: Lage des 2D-Koordinatensystems im 3D-Raum

Nullpunktes des 3D-Raumes in die Projektionsebene ergibt den Nullpunkt des x_1'-x_2'-Achsenkreuzes. Damit liegt das Koordinatensystem in der Projektionsebene fest. Es ist in Abb.5.1 durch die stark ausgezogenen Achsenrichtungen dargestellt.

5) Alle durch die Projektion in der Projektionsebene entstehenden 2D-Objekte werden hinsichtlich ihrer Koordinaten in dem so bestimmten Koordinatensystem dargestellt. Dieses Koordinatensystem wird von GIPSY mit dem normalen 2D-Koordinatensystem der 2D-Objekte zur Deckung gebracht. Damit ist die Zuordnung der Projektionen von 3D-Objekten zu 2D-Objekten (d.h.die Projektion der 3D-Welt in die 2D-Welt) definiert.

Zur Erläuterung ist in Abb.5.2 dargestellt, was sich aus der Projektion des 3D-Achsenkreuzes ergibt, wenn eine Parallelprojektion aus den acht Richtungen der Raumdiagonalen auf eine zur Projektionsrichtung senkrechte Ebene durchgeführt wird. Die Vektorkomponenten des Projektionsvektors sind stets 1 mit jeweils wechselnden Vorzeichen. Auch die vier linken Beispiele von Abb.5.3 entsprechen dem Normalfall der Projektion. In diesem Fall wird jeweils aus der positiven bzw. negativen x= bzw. y-Achse parallel auf eine zum Projektionsstrahl normale Ebene projiziert.

Sonderfall: Projektion auf eine Ebene senkrecht zur x_3-Achse. (von unten oder von oben)

In diesem Sonderfall existiert keine Schnittlinie zwischen Projektionsebene und Grundebene (x_1-x_2-Ebene). Daher ist schon der erste oben genannte Schritt nicht durchführbar. Für diesen Sonderfall gelten folgende Regeln:

1) Die positive x_2-Richtung wird als x_2'-Richtung verwendet. Die x_1'-Richtung ist je nach Lage des Projektionszentrums mit der positiven oder negativen x_1'-Richtung identisch.

2) Als Koordinatenursprung des zweidimensionalen Koordinatensystems in der Projektionsebene gilt die Projektion des 3D-Koordinatenursprungs in die Projektionsebene.

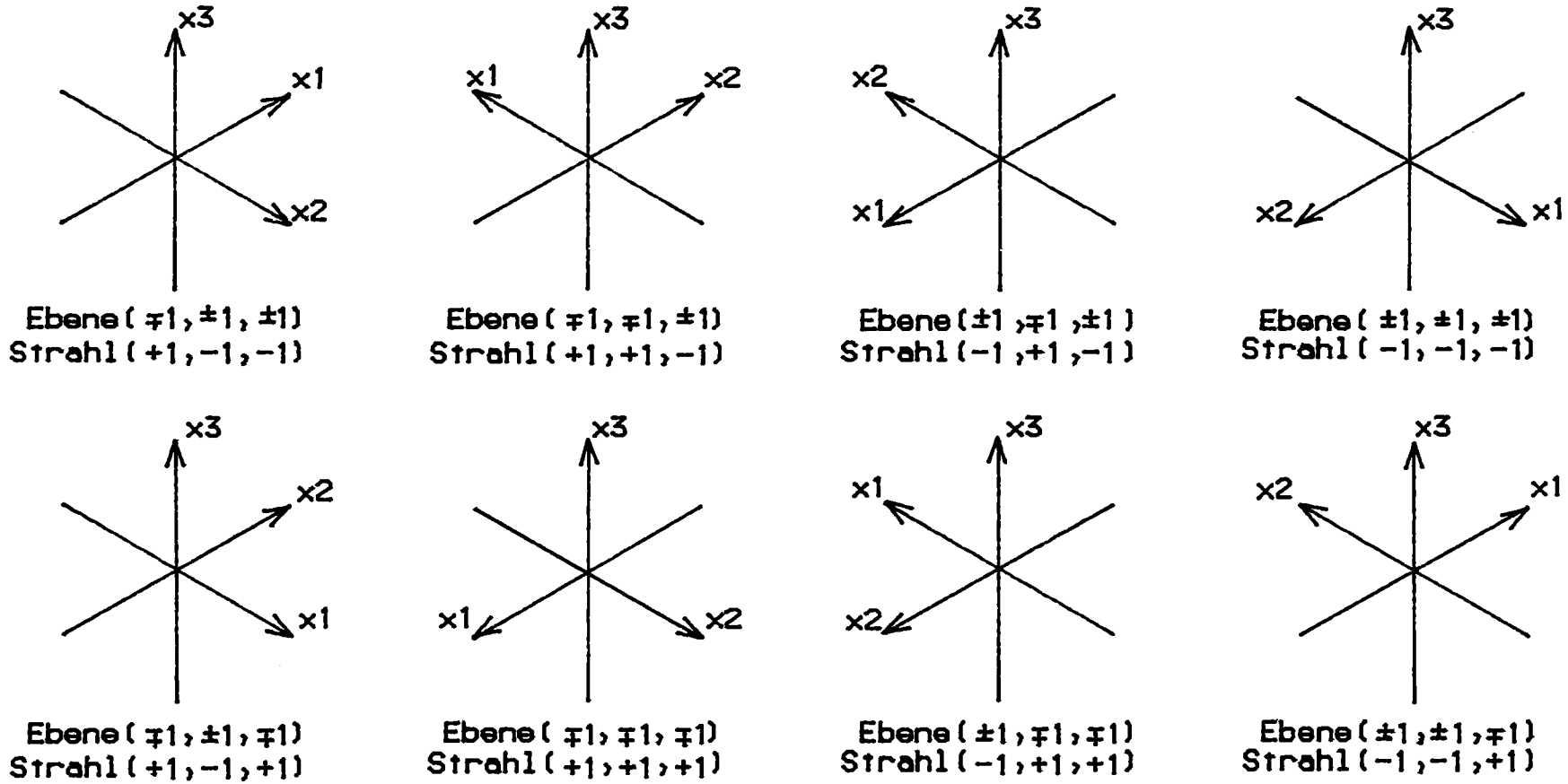


Abb. 5.2: Projektion aus den acht Raumdiagonalen des 3D-Raumes

Anmerkung: Als senkrecht zur x_3-Achse gilt eine Projektionsebene, wenn der Sinus der Neigung ihre Normalenvektors zur x_3-Achse kleiner ist als 10^{-5}.

Zur Erläuterung dieses Sonderfalles dienen die beiden rechten Beispiele von Abb.5.3.

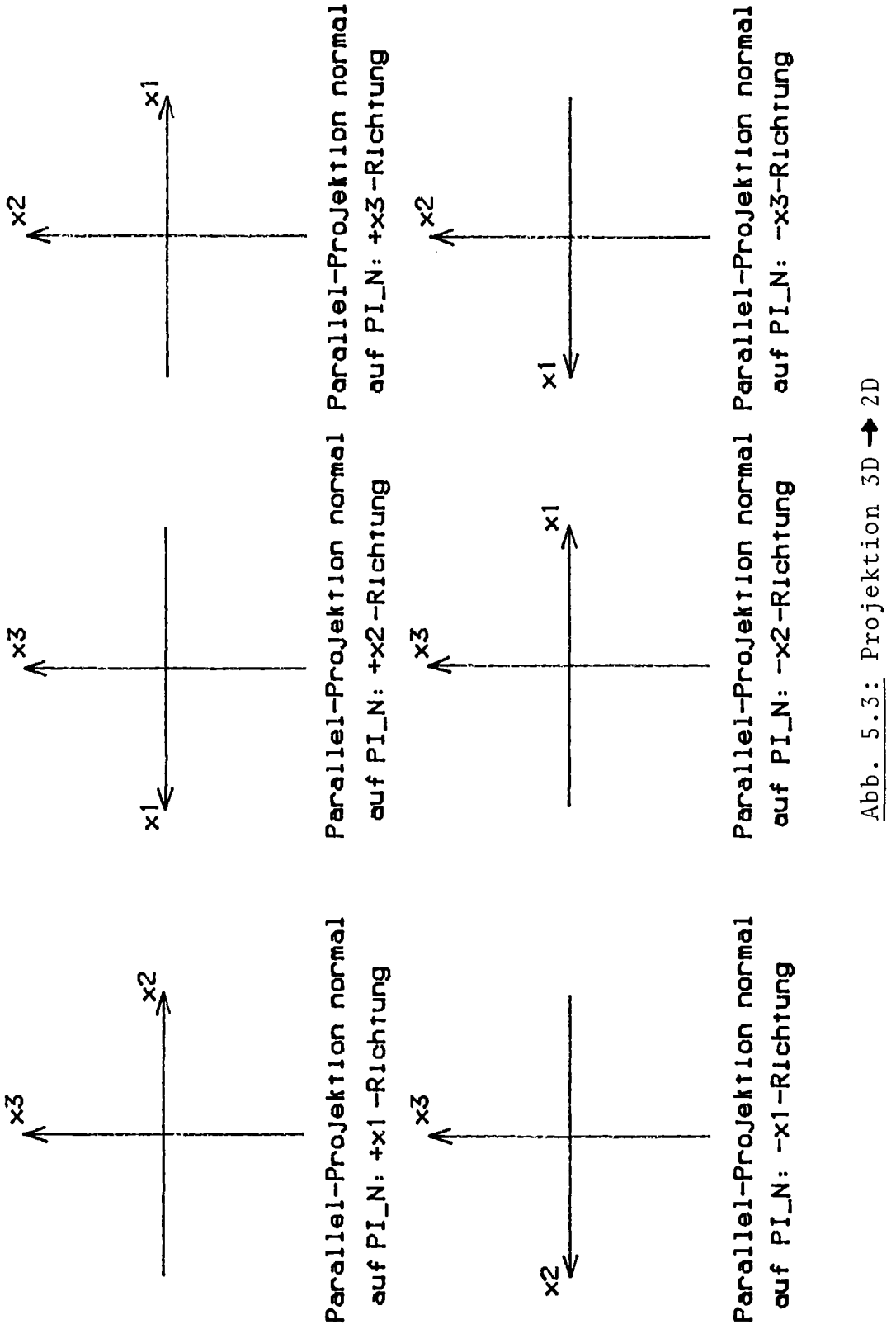


Abb. 5.3: Projektion 3D → 2D

5.3 Sonstige Funktionen

5.3.1 Funktionen zur Behandlung von Kollektionen

Es gibt drei GIPSY-Builtin-Funktionen zur Behandlung von Kollektionen:

CARDCOL
NCOLL, und
OTYP.

CARDCOL liefert als Ergebnis die Kardinalität (Anzahl der Elemente) einer Kollektion. Das Argument muß ein graphisches Objekt g_obj vom Typ Kollektion sein. Diese Funktion liefert also kein GIPSY-Objekt, sondern einen numerischen Wert als Ergebnis ihres Aufrufes. Sie kann daher auch in normalen PL/1-Anweisungen benutzt werden.

NCOLL erhält als Argumente eine Kollektion und einen Ausdruck n, sie liefert das n-te Element der Kollektion.

OTYP erhält ebenfalls als Argumente eine Kollektion und eine Zahl n, sie liefert den Typ des n-ten Elementes der Kollektion als Zeichenkette. Die Zeichenketten entsprechen den Schlüsselwörtern der Deklarationen.

'TEXT' bedeutet z.B. 3D-Text, 'ARC2' ist ein 2D-Bogen oder 'BALL' eine Kugel.

5.3.2 Die COORD-Funktion

Auch die Funktion COORD liefert kein graphisches Objekt, sondern einen DECIMAL FLOAT(6) Wert, der einer Koordinate eines Objektes entspricht. COORD ist das Gegenstück der Anweisung SET COORD (Kap. 6.2).
Die Syntax lautet

COORD(g_obj, index /¯, select_/)

Hinsichtlich der Bedeutung von index und select sei auf Kap.6.2.2 verwiesen. Die Funktion COORD kann in normalen PL/1-Anweisungen verwendet werden.

6. GIPSY-Anweisungen

6.1 Subsystemaufruf und -abschluß

Zum Verständnis dieses Kapitels und des Kapitels 7.4 muß zuerst der Begriff "GIPSY-Prozeß" erläutert werden.

Unter einem "GIPSY-Prozeß" wird die Beschreibung, Aufbereitung und Ausführung von GIPSY-Daten und GIPSY-Befehlen verstanden, die notwendig sind, um graphische Objekte zu erzeugen.

Ein GIPSY-Prozeß beginnt mit

```
ENTER GIPSY;
```

und endet mit

```
END GIPSY;
```

Ein GIPSY-Prozeß kann unterbrochen und wieder aufgenommen werden. Während einer solchen Unterbrechung ruht der GIPSY-Prozeß. Dabei bleibt der Zustand des GIPSY-Subsystems unverändert erhalten. Dies bedeutet, daß gespeicherte Informationen, wie z.B.die aktuelle Strichstärke erhalten bleibt und unverändert bleiben bis zur Wiederaufnahme des Gipsyprozesses. Die GIPSY-Sprache steht allerdings nur während eines aktiven Gipsyprozesses zur Verfügung. Eine Unterbrechung bzw. eine Wiederaufnahme geschieht über das Anweisungspaar

```
END GIPSY LEAVE(ptrref);        =  Unterbrechung
```

und `ENTER GIPSY REENTER(ptrref);` = Wiederaufnahme

"ptrref" ist eine PL/1-Pointervariable, die natürlich denselben Pointer referieren muß.

Im folgenden Beispiel wird schematisch ein Gipsy-Prozeß mit Unterbrechung und Wiederaufnahme skizziert:

```
DCL  PGIPSY POINTER;
 :
ENTER GIPSY;/*BEGINN DES GIPSY PROZESS */
 :
END GIPSY LEAVE(PGIPSY);/*UNTERBRECHUNG DES GIPSY PROZESS */
 :
ENTER GIPSY REENTER(PGIPSY); /* WIEDERAUFNAHME DES GIPSY
 :                                        PROZESS */
END GIPSY; /* ENDE DES GIPSY PROZESS */
```

Ein GIPSY-Prozeß muß nicht in einem Programm, sondern er kann über mehrere Programme oder mehrere Module eines REGENT-Subsystems ablaufen. Letzteres wird ausführlich in Kap.7.4 beschrieben.

Bei Aufnahme eines GIPSY-Prozesses kann durch die FILE-Option in der ENTER GIPSY-Anweisung der DD-Name eines Plotausgabefiles angegeben werden. Ausgabe auf diesen File bewirkt der SAVE-Befehl, siehe hierzu die Kap. 6.5 und 7.3.

6.2 Zuweisungen

Da die normalen PL/1-Zuweisungen auch in GIPSY als PL/1-Statements behandelt werden, müssen für GIPSY-Zuweisungen eigene Statements verwendet werden. Sie heißen SET, SET COORD, FILL und BUILD.

6.2.1 SET

Syntax:

```
SET g_obj = g_ex ;
```

Dabei ist g_obj eine graphische Elementvariable, d.h. ein allein durch den Namen (und evtl. Index) identifiziertes graphisches Objekt. g_ex ist entweder eine graphische GIPSY-builtinfunction (z.B. Transformation), die als Ergebnis ein graphisches Objekt liefert oder ein graphisches Objekt selbst.

Beispiel:

```
DCL (P,P1,P2) POINT2;
DCL (L1,L2) POLY2(2);
DCL C COLL;
SET P = POINT(0.,0.,0.);
SET P1= POINT(0.,0.,0.);   ‖ (*)
SET P2= POINT(0.,0.,0.);   ‖
SET L1= LINE(P,P1); /*P  P1 */
SET L2= LINE(P1,P2); /*P1 P2 */
SET C = COLL(L1,L2); /*P P1 P2 */
```

6.2.2 SET COORD

Um die Koordinaten eines Objektes zu setzen, hat man die Möglichkeit (*) aus 6.2.1. Allgemeiner, da nicht nur auf Punktobjekte anwendbar, ist die Zuweisung durch das Statement SET COORD.

Syntax:

```
SET COORD(g_obj,index[,select]) = PL/1_ex;
```

Hierbei wird die durch"index"angegebene Koordinate in einem Punkt des Objektes g_obj auf den Wert PL/1_ex gesetzt. g_obj kann außer BODY, SPACE und COLLECTION jedes andere graphische Objekt sein."select" wählt bei Polygonzügen den select-ten Punkt aus.

Der betreffende Punkt dabei ist

bei Objekt	Punkt	Anmerkung
POINT2	REFPOINT(g_obj) bzw. REFPOINT2	select wird ignoriert
POINT	"	"
CIRCLE2	"	"
CIRCLE	"	"
TEXT2	"	"
TEXT	"	"
PLANE	"	"
BALL	"	"
CYLINDER	"	"
CONE	"	"
POLY2	REFPOINT(NPOINT(g_obj,select))	
POLYGON	"	
ARC2 ARC	REFPOINT bei select ≠ 1,2 Anfangspunkt bei select = 1 Endpunkt bei select = 2	Standard:select =0

index = 1 bedeutet: x-Koordinate
index = 2 bedeutet: y-Koordinate
index = 3 für 3D-Objekte bedeutet: z-Koordinate

Um sicherzustellen, daß bei falscher Index-Angabe nicht auf Werte außerhalb der Objekte zugegriffen wird, wird folgender effektiver Index benutzt:

i_{eff} = modulo(index-1,3)+1 bei 3D-Objekten
i_{eff} = modulo(index-1,2)+1 bei 2D-Objekten

Entsprechend wird bei "select" vorgegangen.

Beispiel:

```
   SET COORD(P,1) = 0;
   SET COORD(P,2) = 0;          entspricht P
   SET COORD(P,3) = 0;            aus 6.2.1

DCL B BALL;

   SET COORD(B,1) = 0;          /* Die Kugel B  hat
   SET COORD(B,2) = 1;          den Mittelpunkt 0,1,2 */
   SET COORD(B,3) = 2;
DCL PL POLY2(N);
DO I=1 TO N;
   GET LIST(x,y);
   SET COORD(PL,1,I)=x;
   SET COORD(PL,2,I)=y;
END;
```

Dem Punkt I des Polygonzuges werden die Koordinaten x und y zugewiesen, I läuft von 1 bis N.

6.2.3 FILL

FILL dient dazu, einen Polygonzug mit Werten zu füllen.
Die Syntax lautet:

FILL g_obj WITH(PL1_ex1[,PL1_ex$_i$]*);

g_obj ist ein Polygonzug (2- oder 3-dimensional).
Die PL1_ex$_i$ bedeuten bei POLY2:

$x_1, y_1, x_2, y_2, \ldots$

bei POLYGON:

$x_1,y_1,z_1,x_2,y_2,z_2,\ldots\ldots$

Beispiel:

```
DCL  P3 POLY2(3);
FILL P3 WITH(0,0,x1,y1,2*x1,3*y1);
```

6.2.4 BUILD

Mit BUILD werden Raumelemente(SPACE) und Körper (BODY) aufgebaut.

Syntax:

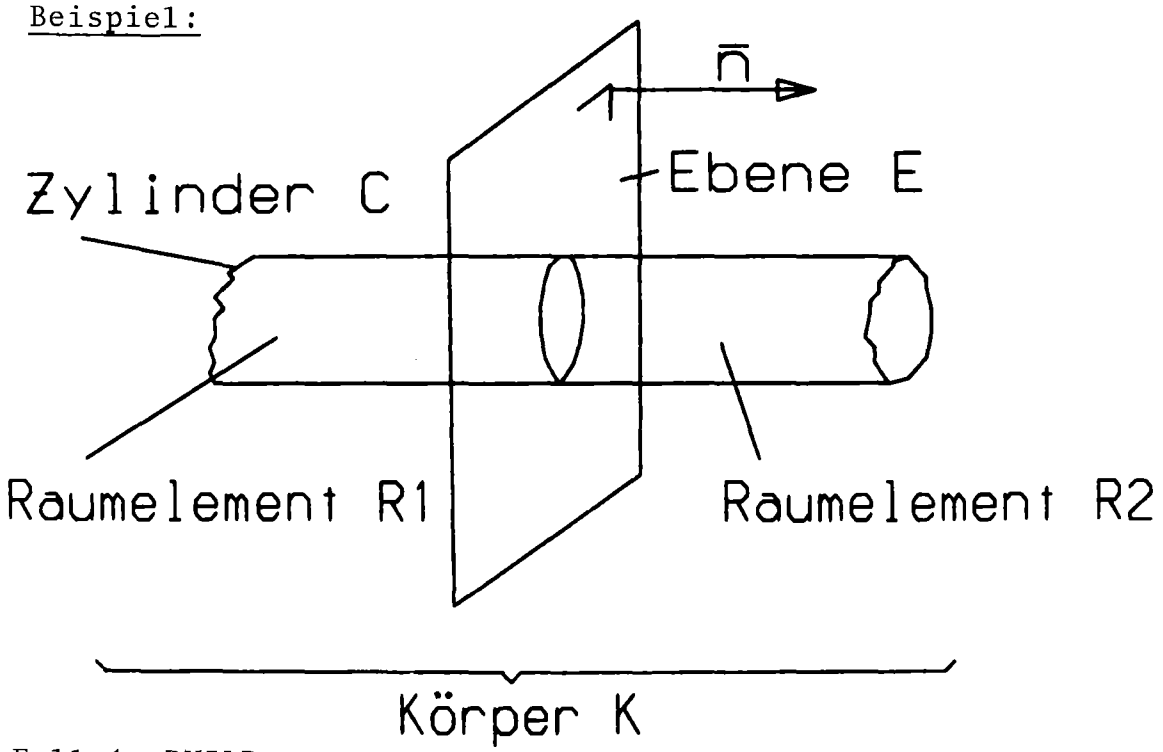
BUILD spaceobject = SPACE($O_1 \pm O_2 \pm O_3 \ldots \pm O_n$); (1)

BUILD bodyobject = BODY($s_1,s_2,\ldots,s_n$); (2)

Bei (1) werden die Oberflächen o_i zur Definition des Raumelementes spaceobject zusammengefaßt, während im Fall (2) ein Körper bodyobejct aus den Raumelementen s_i aufgebaut wird. Die o_i können Oberflächen, Raumelemente (SPACE) oder Kollektionen dieser Typen sein. Die s_i können Körper (BODY), Raumelemente (SPACE) oder Kollektionen dieser Typen sein.

Im Gegensatz zu SET werden beim Körperaufbau mit BUILD die Oberflächen, aus denen ein Raumelement aufgebaut ist, nicht kopiert, sondern lediglich referiert.

Beispiel:



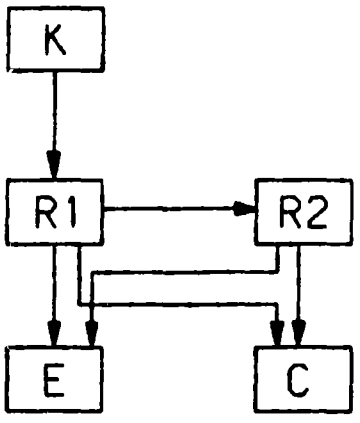
Fall 1 BUILD

```
BUILD R1 = SPACE (E+C+...)
BUILD R2 = SPACE (-E+C+...)
BUILD K  = BODY(R1,R2)
PLOT(K);
```

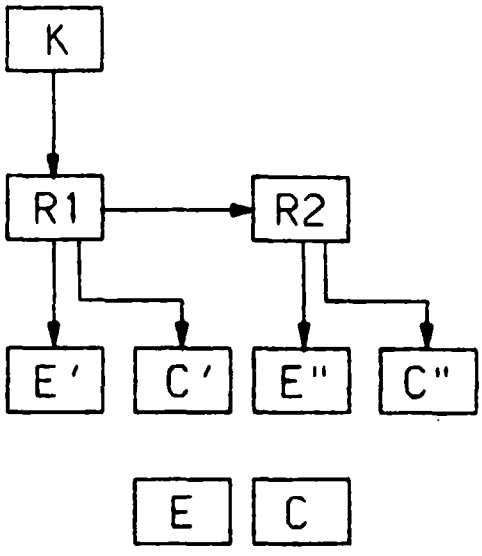
Fall 2 SET

```
SET  R1 = SPACE (E+C+...)
SET  R2 = SPACE (-E+C+...)
SET  K  = BODY (R1,R2);
PLOT(K);
```

Datenstruktur

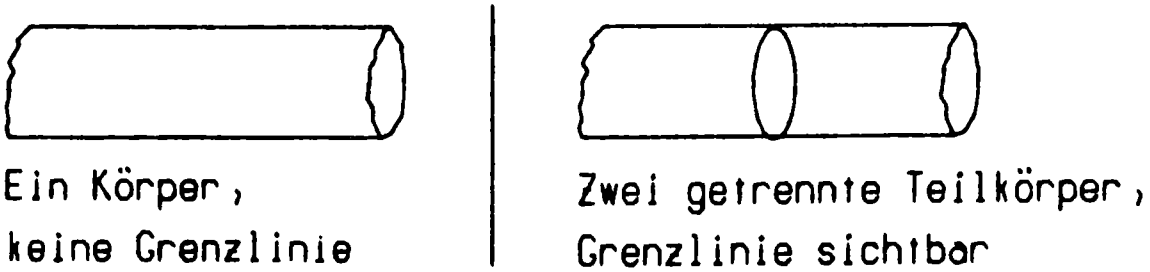


E und C nur im Original vorhanden



K besteht aus Kopien E',E",C',C"; E und C noch vorhanden

Bild



Ein Körper,
keine Grenzlinie

Zwei getrennte Teilkörper,
Grenzlinie sichtbar

6.2.5 EMPTY

EMPTY kann nur auf GIPSY-Objekte vom Typ SPACE, BODY oder COLLECTION angewandt werden und ist wirkungslos auf alle andere Objekte.

Syntax:

EMPTY({ space-obj / body-obj / coll-obj } [, { space-obj / body -obj / coll-obj }]*);

Wirkung:

Das SPACE-, BODY- oder COLLECTION-Objekt enthält nach Anwendung keine Elemente mehr.

6.3 Zeichnungsausgabe

Die Zeichnungsausgabe erfolgt durch das Statement PLOT. Die Größe und Anordnung des Bildfensters wird durch den Befehl OPEN PLOT festgelegt. Für Ausgabetransformationen steht ferner das Statement VIEW zur Verfügung. Beispiele siehe auch in den Kapiteln 8.1 und 8.2.

6.3.1 OPEN PLOT

Syntax:

OPEN PLOT { DIN A (expr) /¯BROAD_7 / SIZE (lvalue, lvalue) }
{ /¯NOFRAME_7 /¯POSITION (x, y)_7 / /¯POSITION (x,y)_7/¯NOFRAME_7 } ;

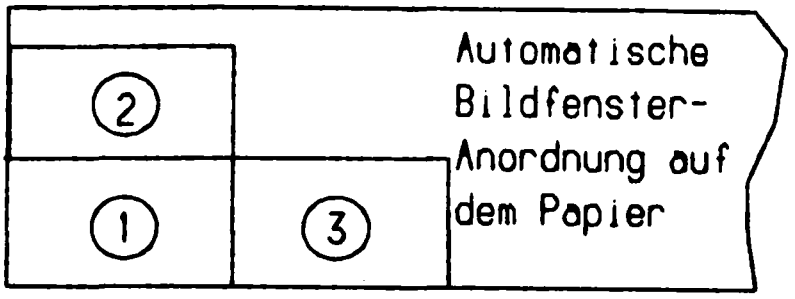
Die Abmessungen des Bildfensters können entweder über SIZE angegeben werden, also z.B.

```
OPEN PLOT SIZE(10 CM, 10 CM)
```

oder über ein DIN-Normformat

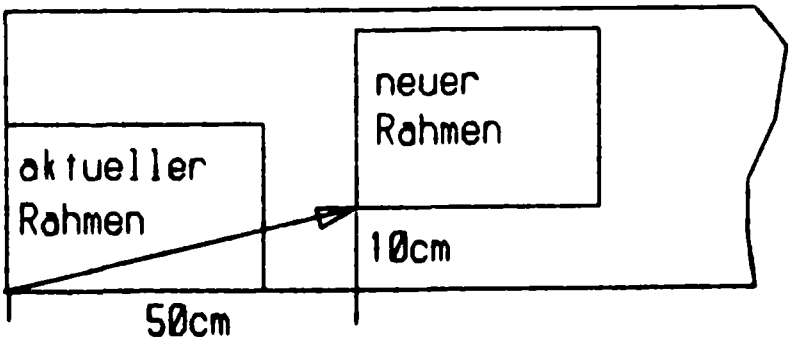
```
OPEN PLOT DIN A(4);
```

Durch Angabe der Option NOFRAME wird das Zeichnen des Bildrahmens unterdrückt.
Die Lage des Bildfensters auf dem Zeichenpapier kann durch den Parameter POSITION gesteuert werden oder automatisch erfolgen.




```
OPEN PLOT DIN A(4) POSITION (50 CM, 10 CM);
```

bewirkt, daß der Nullpunkt des neuen Bildes gegenüber dem Nullpunkt des zuletzt spezifizierten Bildfensters um 50 cm in x- und um 10 cm in y-Richtung verschoben wird.



6.3.2 PLOT

Syntax:

```
PLOT (g_ex [ , g_ex ]* ) ;
```

g_ex ist ein graphischer Ausdruck.

Die PLOT-Anweisung in dieser Form bewirkt die Ausgabe der zeichnerischen Darstellung von graphischen Objekten (Punkte, Polygone

Kollektionen, Texte etc.). 3D-Objekte werden dabei entsprechend den in der CHANGE-Anweisung spezifizierten Projektionsparametern projiziert.(Wurde kein CHANGE angegeben, gelten die Standardprojektionswerte).

Bei der Ausgabe werden alle Werte auf das Verlassen der Zeichenfläche überprüft. Liegen Werte außerhalb der Zeichenfläche, so erfolgt eine Fehlermeldung.

6.3.3 VIEWING TRANSFORMATION

Syntax:

VIEWING /TRANSFORMATION_/ /SHIFT(x,y)_/

/CLIPP(l,u,r,o)_/ /SCALE (sx,sy)_/ ;

VIEWING TRANSFORMATION legt die Abbildungstransformation vom 2D-GIPSY-Raum auf die Zeichenfläche fest. Dabei kann der Nullpunkt des 2D-Raumes auf der Zeichenfläche plaziert werden (SHIFT), ein Bildausschnitt kann angegeben werden (CLIPP) und ein Abbildungsmaßstab kann spezifiziert werden (SCALE). x und y sind die Koordinaten des 2D-Nullpunktes auf der Zeichenfläche. l, u, r, o sind die Koordinaten der linken unteren und rechten oberen Ecke des Bildausschnittes im 2D- GIPSY-Raum. Nur innerhalb des CLIPP-Ausschnittes liegende Objekte oder Objektteile werden gezeichnet. Alles andere wird ohne Fehlermeldung abgeschnitten. sx und sy sind Maßstäbe in x- und y-Richtung. Wird sy nicht angegeben, so ist sy=sx. Folgende Abb.6.1 verdeutlicht die Abbildungstransformation. Siehe auch Kapitel 8.1

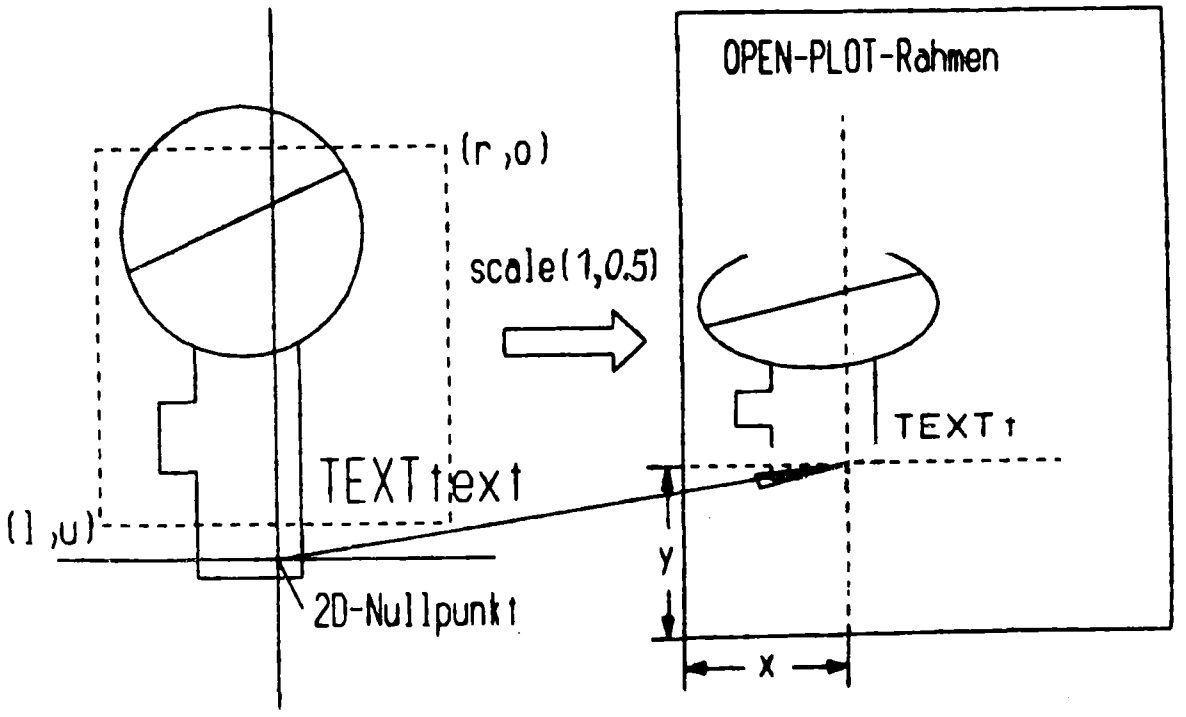


Abb. 6.1: Abbildungstransformation

6.4 Druckausgabe (STATUS, PRINT, TRACE)

Die Statements STATUS und PRINT dienen zur Ausgabe des Systemzustandes und zum Ausdrucken graphischer Objekte.

STATUS gibt eine Übersicht über den aktuellen Zustand der Systemvariablen.

Syntax:

```
STATUS { → ALL       }        (S1-S5)
       { UNITS       }        (S1)        Standardwert ALL
       { PROJECTION  }   ;    (S2)
       { PLOT        }        (S3,S4,S5)
```

Gelistet werden bei

S1 - Standardlängen- und Winkelangaben

S2 - Projektionsebene, Projektionsart und Projektionsrichtung

S3 - Inkremente bei der Kurvenberechnung

S4 - Ursprung, Skala, Linientyp

S5 - Papierabmessungen, Papierart und maximaler Koordinatenwert der Zeichnung.

Ein Beispiel ist in Abb.6.2 wiedergegeben.

PRINT listet die graphischen Attribute eines Objektes in leicht lesbarer Form unter namentlicher Kennzeichnung. Die Ausgabe erscheint in strukturierter Form und kann zu Testzwecken gut verwendet werden.

Syntax:

```
PRINT (g_obj [ , g_obj ]*);
```

Beispiel:

```
DCL P POINT2;
SET P = POINT2(3,28);
EDIT SYMBOL(5),HEIGHT(6 MM) OF (P);
PLOT(P); PRINT(P);
```

```
+-------------------------------------------------------------+
!                                                             !
! GRAPHIC STATE INFORMATION: ALL        OF GIPSY RELEASE 3.1  !
! ALL LENGTH VALUES IN METER                                  !
+-------------------------------------------------------------+
! STANDARD ATTRIBUTES                                         !
! DISTANCE      !   2.000000E-02   ! SYMBOL       !   1      !
! HEIGHT        !   4.999999E-03   ! EVERY        !   1      !
! LENGTH        !   4.999999E-03   ! COLOR        !   0      !
! LINEWIDTH     !   0.000000E+00   ! PEN          !   0      !
! SLANT         !   0.000000E+00   ! LINETYPE     !   SOLID  !
! CHARACTER WID!   4.999999E-03   ! OPEN                      !
+-------------------------------------------------------------+
! LENGTH UNIT: METER                                          !
! ANGLE UNIT: DEGREES                                         !
+-------------------------------------------------------------+
! PARALLEL PROJECTION WITH DIRECTION:                         !
!  1.000000E+00   0.000000E+00   0.000000E+00                 !
! PROJECTION ORIGIN:                                          !
!  0.000000E+00   0.000000E+00                                !
+-------------------------------------------------------------+
! PICTURE PLANE:                                              !
! PI_N          !   1.000000E+00   0.000000E+00   0.000000E+00  !
! PI_P          !   0.000000E+00   0.000000E+00   0.000000E+00  !
+-------------------------------------------------------------+
! PROSPECTIVE MATRIX:                                         !
+-------------------------------------------------------------+
!  1.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00   !
!  0.000000E+00   0.000000E+00   1.000000E+00   0.000000E+00   !
!  0.000000E+00   0.000000E+00   0.000000E+00   1.000000E+00   !
+-------------------------------------------------------------+
! INCREMENT FOR COMPUTATION OF INTERSECTIONS:                 !
! STRAIGHT      !   7.999998E-03                              !
! CURVED        !   1.200000E+01                              !
+-------------------------------------------------------------+
! WIEWING TRANSFORMATION:                                     !
! ORIGIN(M)     !   0.000000E+00   0.000000E+00               !
! SCALE         !   1.000000E+00   1.000000E+00               !
+-------------------------------------------------------------+
! CLIPP VALUES:                                               !
! LOWER LEFT    ! -1.000000E+50  -1.000000E+50                !
! UPPER RIGHT   !   1.000000E+50   1.000000E+50               !
+-------------------------------------------------------------+
! INVISIBLE LINE   ATTRIBUTES:                                !
! HEIGHT        !   4.999999E-03   ! EVERY        !   1      !
! LENGTH        !   4.999999E-03   ! COLOR        !   0      !
! LINEWIDTH     !   0.000000E+00   ! PEN          !   0      !
! SYMBOL        !   1              ! LINETYPE     !   DASHED !
+-------------------------------------------------------------+
! MINIMUM/MAXIMUM COORDINATES:                                !
! DOWNLEFT(M)   !   1.000000E+50   1.000000E+50               !
! UPRIGHT(M)    ! -1.000000E+50  -1.000000E+50                !
+-------------------------------------------------------------+
! FACTOR        !   3.937007E+01   ! ARC      !   3.500000E-02  !
+-------------------------------------------------------------+
! PAPER(M)      !   3.000000E+01   5.200000E-01               !
! FIGURE(M)     !   0.000000E+00   0.000000E+00               !
! MAX POSIT.(M)!   0.000000E+00   0.000000E+00               !
+-------------------------------------------------------------+
! PICTURE NR    !   1                                         !
! SAVE UNFORMATTED               ! SAVE CALL OFF              !
! PLOT CALL ON                   ! PLOT ALLOWED               !
+-------------------------------------------------------------+
```

Abb.6.2: Ausdruck des STATUS-Statements

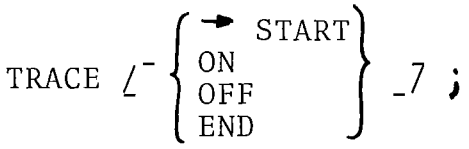
```
+-----------------------------------------------------------+
| POINT2:P                                                  |
+-----------------------------------------------------------+
| SYMBOL:    5              | HEIGHT: 5.999997E-03          |
+-----------------------------------------------------------+
| PEN:    0         | LINEWD.: 0.000000E+00 | COLOR:    0     |
+-----------------------------------------------------------+
| X(*)   | 3.000000E+00      2.800000E+01                     |
+-----------------------------------------------------------+
```

```
DCL T TEXT2(30);
SET T=TEXT2(P,45.,'BEISPIEL-TEXT UNTER 45 GRAD');
EDIT HEIGHT(2 MM) FOR(T);
PLOT(T);
PRINT(T);
```

```
+-----------------------------------------------------------+
| TEXT2:T                                                   |
+-----------------------------------------------------------+
| PEN:    0         | LINEWD.: 0.000000E+00 | COLOR:    0     |
+-----------------------------------------------------------+
| NCHAR:        30                                          |
| X1(*)  | 3.000000E+00      2.800000E+01                     |
| X2(*)  | 3.003535E+00      2.800352E+01                     |
| X3(*)  | 2.998583E+00      2.800140E+01                     |
+-----------------------------------------------------------+
+-----------------------------------------------------------+
| STRING:BEISPIEL-TEXT UNTER 45 GRAD                        |
+-----------------------------------------------------------+
```

Das TRACE-Statement schaltet einen Modulaufruf-Kontrollausdruck ein und setzt zusätzlich eine Kontrollvariable in GIPSY so, daß vor allem bei der Körperanalyse ein umfangreicher Kontrollausdruck erfolgt. Das TRACE-Statement ist wichtig zur Analyse von GIPSY-Fehlern.

Syntax:



TRACE; TRACE START; oder TRACE ON; starten den Kontrollausdruck, TRACE OFF; oder TRACE END; beenden den Ausdruck wieder.

6.5 Plotfile-Ausgabe (SAVE)

Im Allgemeinen wird die Ausgabe von GIPSY-Programmen auf einen Plotter gelegt. Dies wird in der REGENT-Option PLOT spezifiziert,

also z.B. PLOT=STATOS oder PLOT=XYNETICS. Es ist jedoch auch möglich, die Plot-Ausgabe auf eine Datei zu schreiben, deren Aufbau den Richtlinien des AGF-Koordinierungsausschuß Graphik entspricht. Diese Datei wird im folgenden als Plotfile oder GIPSY-SAVE-File bezeichnet.

Zur detaillierten Beschreibung bzw. für Benutzer, die mit GIPSY interaktiv an einem Bildschirm arbeiten wollen, wird auf Kapitel 7.3 verwiesen, das auch eine Benutzungsanleitung enthält. Für die Erstellung von Filmen siehe ebenfalls Kapitel 7.3. In diesem Kapitel wird nur gezeigt, wie ein GIPSY-SAVE-File erstellt wird.

6.5.1 Die Datei

Der GIPSY-Save-File hat das Record-Format FB. Es kann eine normale TSO-Datei verwendet werden oder ein Band. Der Standard-Plotfile hat die Werte: DCB=(BLKSIZE=960, LRECL=80, RECFM=FB). GIPSY und die Interpretierer im KfK können ein beliebiges Record-Format bearbeiten. Nachfolgend zwei Beispiele, wie die DD-Karten aussehen könnten.

```
//G.PLOTLIB DD DSN=TSO253.PLOTLIB.DATA,DISP=SHR (bestehende TSO-
                                                 Datei)

//G.PLOTLIB DD DSN=SAVEFILE,UNIT=TAPEX,VOL=SER=DVOO71,DISP=(NEW,KEEP),
                  LABEL=(1,SL),DCB=(BLKSIZE=3200,LRECL=80,
                     RECFM=FB,DEN=3)
            (Plotfile auf Band DVOO71, Label 1)
```

Der DD-Name, über den der Plotfile referiert wird, lautet PLOTLIB.

Der File, der mit der Datei verknüpft ist, wird an GIPSY im ENTER-Statement übergeben. Zusätzlich muß filename deklariert sein und zwar lediglich mit dem Attribur FILE.

```
DCL filename FILE;
ENTER GIPSY FILE(filename);
```

Standardwert für filename ist PLOTLIB.

6.5.2 SAVE

Mit den Statements SAVE;und SAVE END; wird nun im Programm spezifiziert, von wo bis wo Bilder auf den Plotfile ausgegeben werden. Es können dabei in ein und demselben Programm PLOT- und SAVE-Befehle gemischt werden. Wird nur mit SAVE gearbeitet, so kann die REGENT-Option PLOT=... entfallen. Das SAVE-Kommando bewirkt, daß sämtliche nachfolgende Plotbefehle in Save-Aufrufe umgesetzt werden bis zum Auftreten des SAVE END; Statements. Es ist also auf einfachste Weise möglich, ein größeres Programm vom Plotter auf den Plotfile umzustellen durch Ändern von ENTER GIPSY FILE(*) und Einfügen von SAVE. Hinzu kommt die File-Deklaration für den Plotfile und die PLOTLIB-DD-Karte.

Beispiel:

```
BEISP: PROC OPTIONS(MAIN) REGENT(PLOT=STATOS,MPOOL=300000);
       DCL AUSGABE FILE;/* GIPSY_SAVE_FILE */
       ENTER GIPSY FILE(AUSGABE);
        DCL P POINT2;
        DCL T TEXT2;
        :
        DCL SAVE_BIT BIT(1) INIT('1'B);
        :
         SAVE_BIT=...
        :
        IF SAVE_BIT THEN SAVE;                    Plotausgabe
         OPEN PLOT DIN A(4) BROAD;  }             auf Plotter
         PLOT(P,T);                 }             oder Plotfile
        IF SAVE_BIT THEN SAVE END;
        :
       END GIPSY;
       FINISH;
    END BEISP;
```

Das OPEN-PLOT-Statement darf in obigem Beispiel zwischen SAVE; und SAVE END; nicht vergessen werden. Soll sowohl eine Ausgabe auf Plotter als auch gleichzeitig eine Plotfile-Ausgabe erzeugt werden, so kann dies durch SAVE PLOT; anstelle von SAVE; erzeugt werden. Der Plotfile wird normalerweise im internen Datenformat beschrieben und so von den Plotfile-Programmen im KfK auch gelesen. Soll jedoch ein Plotfile im Standard-AGF-Format erzeugt werden, z.B. um ihn bei einer anderen Institution wieder auszugeben, so ist im ersten SAVE die Zusatzangabe START FORMATTED erforderlich. Der Plotfile benötigt dadurch etwa 80% mehr Speicherplatz. Die KfK-Programme können auch die formatierte Plotfile-Version verarbeiten.

Syntax des SAVE-Statements:

SAVE [START { FORMATTED / UNFORMATTED }] [PLOT] ;

Beginn der Plotfile-Ausgabe

SAVE END;

Ende der Plotfile-Ausgabe.

6.6 Ändern von Attributen (CHANGE, EDIT)

Mit dem Statement CHANGE ist es möglich, die Werte von Systemvariablen neu zu setzen. EDIT hingegen verändert nur die objektspezifischen Attribute der graphischen Elemente.

Beispiel:

```
CHANGE UNITS LENGTH(MM);
EDIT HEIGHT(5 CM)FOR(T);
```

Im obigen Beispiel bewirkt CHANGE, daß alle nachfolgenden Längenangaben als Millimeter interpretiert werden, während EDIT nur dem Objekt T (z.B.Text) die Zeichenhöhe 5 cm zuordnet.

6.6.1 CHANGE

CHANGE ändert die Systemvariablen für die

- Standardeinheiten (Längen-, Winkelmaß)
- Projektionsart (parallel, zentral,...)
- Inkremente bei Kurvenberechnung
- Standardattribute (Linientyp, Farbe, Texthöhe,...)
- Skalierung (FACTOR)
- Umwandlung von Kreisen und Bögen in Polygone.

Die volle Syntax lautet:

```
CHANGE
  ( STANDARD      Attribut_und_sein_Wert
    FACTOR        (PL1_ex)
    ARC           (avalue)
    UNITS       { LENGTH /¯(_/ lunit /¯)_/ }
                { ANGLE  /¯(_/ aunit /¯)_/ }
    PROJECTION  { PARALLEL   (lvalue, lvalue, lvalue) }
                { CENTRAL    (lvalue, lvalue, lvalue) }
                { PI_N       (lvalue, lvalue, lvalue) }
                { PI_P       (lvalue, lvalue, lvalue) }
                { ORIGIN     (lvalue, lvalue)         }
    INCREMENT   { LINEAR     (lvalue) }
                { ANGLE      (avalue) }
    INVISIBLE     Attribut_und_sein_Wert
    MAXPEN        (PL1_ex)
    MAXCOLOR      (P11_ex)
  )[,...]*;
```

Attribut_und_sein_Wert als Suboption bei STANDARD hat dabei folgende Optionen:

SYMBOL	I	LINETYPE	SOLID, DASHED, CENTER, OMITTED MARKED, DOTTED, HAND
HEIGHT	lvalue	LENGTH	lvalue
EVERY	I_+	PEN	I
OPEN	-	COLOR	I
CLOSED	-	LINEWIDTH	lvalue
SLANT	avalue	WIDTH	lvalue
DISTANCE	lvalue		

(I_+ = Integer > 0 I = Integer ≥ 0)

UNITS

Standardwerte sind M und DEGREE. Diese können überschrieben werden. Die Syntax dazu lautet:

CHANGE { UNITS LENGTH { KM | M | MM | CM | INCH | FOOT | YARD } | UNITS ANGLE { GRAD | DEGREE | RADIAN } } [,UNITS...];

PROJECTION

Als Standardansicht ist in GIPSY eine senkrechte Parallelprojektion in die im Ursprung befindliche (x_2,x_3)-Ebene, vorgesehen. Bei Parallelprojektion ist die Betrachtungsrichtung vom Punkt (x_n, y_n, z_n) zum 3D-Ursprung. Intern erfolgt dann eine Normierung auf den Einheitsvektor. Bei der Zentralprojektion bestimmen die Koordinatenangaben (x_p, y_p, z_p) das Projektionszentrum.

Die Bildebene π, die noch frei wählbar ist, wird durch einen Punkt (PI_POINT) und die Normalenrichtung (PI_NORMAL) festgelegt. Dabei ist natürlich darauf zu achten, daß die Projektionsstrahlen nicht parallel zur Bildebene verlaufen, da sonst kein reales Bild entsteht. Mit CHANGE PROJECTION ORIGIN (x,y) wird die Lage des 3D-Nullpunktes im GIPSY-2D-Raum festgelegt. Siehe auch Kapitel 3.3, 5.2.3 und 8.3.

INCREMENT

Durch die Wahl der Inkremente wird die Effektivität bei der Körperanalyse stark beeinflußt. Standardwerte sind 8 mm und 0.1(5.7 Grad). Wenn es für bestimmte Objekte von Vorteil ist, kann man dafür eine Optimierung vornehmen.
Die Inkremente sollen sich an der kleinsten (kürzesten) aufzulösenden Körperkante orientieren. Nur Konturen und verdeckte Kanten, die größer als das durch die Inkremente gesetzte Raster sind, werden bei der Körperanalyse erfaßt. Sehr kleine Inkremente dagegen vergrößern den Rechenzeitbedarf erheblich. Für das Längeninkrement kann als Faustregel ein Wert von 1% bis 2% der längsten Körperabmessung gewählt werden. Für Testläufe empfiehlt es sich, größere Inkremente zu wählen.

Syntax:

CHANGE INCREMENT { LINEAR lvalue / ANGLE avalue } [,INCREMENT...] ;

INVISIBLE

Syntax:

CHANGE INVISIBLE { LIKE STANDARD / Attribut_und_sein_Wert } [,INVISIBLE ...]* ;

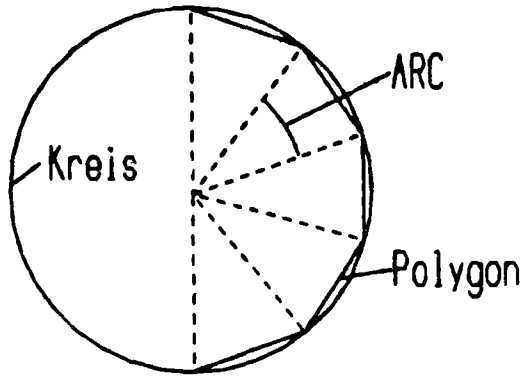
Attribut_und_sein-Wert: Siehe CHANGE STANDARD

Verdeckte Kanten von Körpern können unterschiedlich dargestellt werden. Der Standard-Linientyp ist für sichtbare Linien SOLID, für unsichtbare DASHED. Die sonstigen Attribute (Farbe, Strichstärke,...) sind die Standardsystemattribute. Für sichtbare Linien können sie durch CHANGE STANDARD geändert werden, für unsichtbare Linien durch CHANGE INVISIBLE. Hier sind alle für POLY (Bzw. POLY2) gültigen Attribute außer OPEN/CLOSED möglich, also HEIGHT, LENGTH, EVERY, PEN, COLOR und LINEWIDTH. Die unsichtbaren Linien lassen sich durch Angabe des Linientyps OMITTED unterdrücken. Die Überprüfung auf Unsichtbarkeit wird unterdrückt, wenn STANDARD und INVISIBLE Attribute gleich sind. Dies kann erreicht werden durch CHANGE INVISIBLE LIKE STANDARD. Dadurch wird Rechenzeit gespart (für Testläufe!)

Anmerkung: INVISIBLE mit kleinen Inkrementen ergibt teilweise 30-fach höhere Rechenzeiten, da der Algorithmus der Unsichtbarkeitsprüfung sehr aufwendig ist!

ARC

ARC setzt den Winkel beim Erzeugen eines gleichseitigen Polygons aus einem Kreis beim Umwandeln von Kreisen und Bögen in Polygonzüge. Diese Operation wird beim Zeichnen und bei der Operation POLY/POLY2 (Kreis oder Bogen) durchgeführt.



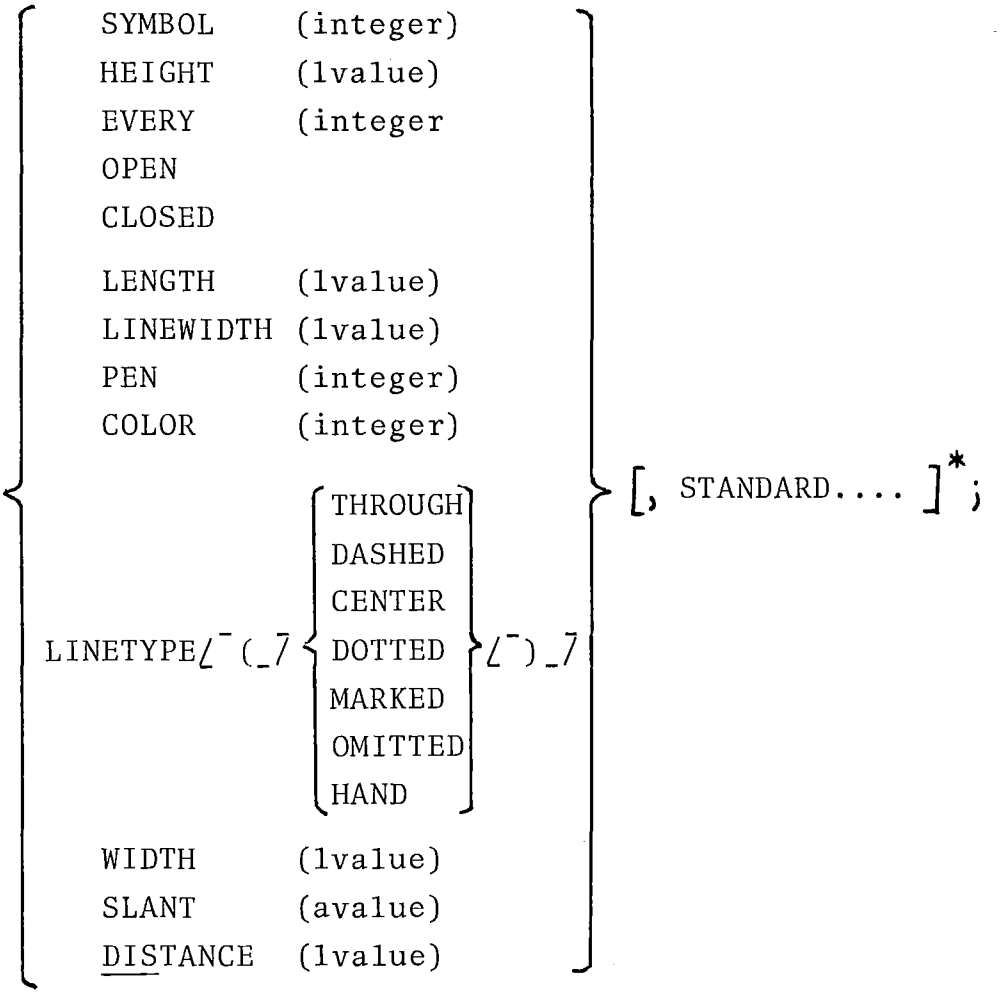
FACTOR

setzt den Wert, den 1 m Länge in Plotter-Koordinaten hat. (Für Inch-Plotter 100/2.54 für cm-Plotter 100.0).

STANDARD

Die Syntax hierfür lautet:

CHANGE STANDARD



CHANGE STANDARD dient dazu, die Standardwerte für Attribute zu ändern."integer"steht dabei für Integer ≥ 0,"integer$_+$" für Integer > 0. Die lvalues müssen alle positiv sein. Zur Bedeutung der Attribute siehe Kapitel 4.3.2 und EDIT,Kapitel 6.6.2.

MAXPEN, MAXCOLOR

Durch CHANGE MAXPEN (PL1_ex) wird die höchstzulässige Stiftnummer festgelegt. Durch CHANGE MAXCOLOR (PL1_ex) die maximal zulässige Farbnummer. Höhere Werte werden gemäß

$$n' = \mathrm{mod}\ (n-1,\ nmax)+1$$

umgerechnet.

6.6.2 EDIT

EDIT verändert die objektspezifischen Ausgabeparameter des graphischen Elementes wie

- den Symboltyp von Punkten
- die Schrifthöhe, -breite und Neigung von Texten
- die Höhe von Punktsymbolen
- die Linienart von Polygonzügen und deren Kennzeichnung
- Skalenstrichabstand von Koordinatenachsen
- Strichstärke, Farbe und Schreibstift (PEN).

Es steht daher im Gegensatz zum Statement CHANGE bei EDIT immer das graphische Objekt oder eine Liste graphischer Objekte, auf die EDIT wirken soll.

Das EDIT-Statement muß nicht unmittelbar vor dem Plotaufruf stehen; jedoch darf dazwischen kein SET-Statement auftreten, da sonst die EDIT-Attribute wieder geändert werden.

Die Syntax lautet:

```
EDIT { SYMBOL     (pl1_ex)      }
     { HEIGHT     (lvalue)      }
     { EVERY      (pl1_ex)      }
     { LENGHT     (lvalue)      }
     { LINEWIDTH  (lvalue)      }
     {   OPEN                   }
     {   CLOSED                 }
     { PEN        (pl1_ex)      }
     { COLOR      (pl1_ex)      }
     { LINETYPE   /¯(_7 typ /¯)_7 }
     { WIDTH      (lvalue)      }
     { SLANT      (avalue)      }
     { DISTANCE   (lvalue)      } [, { : } ]*

{ FOR } (g_obj   /¯,g_obj_7 *) ;
{ OF  }
```

Typ als Subparameter von LINETYPE kann sein:

SOLID, DASHED, CENTER, DOTTED, MARKED, OMITTED, HAND.

Wird EDIT auf eine Kollektion angewandt, so ist die Wirkung die gleiche, als wenn EDIT nacheinander auf alle Teile der Kollektion angewandt wird.

SYMBOL

Durch diese Angabe wird für Punkte und Polygonzüge das zur Kennzeichnung verwendete Symbol ausgewählt. Der Wert von pl1_ex entspricht dabei den Symbolnummern, siehe Kap. 4.3.2.

```
EDIT SYMBOL(3) FOR(P);
```

P wird durch das Symbol mit der Symbolnummer 3 dargestellt.

HEIGHT

```
EDIT HEIGHT(5MM)FOR(P,T);
```

Die Höhe von Punktsymbolen für Punkt P und Text T wird durch diese Anweisung festgelegt.

EVERY

```
EDIT EVERY(5)FOR(P);
```

kennzeichnet jeden 5. Punkt eines Polygonzuges P durch sein Punktsymbol. (Festlegung in EDIT SYMBOL.....)

LENGTH

Mit dem Befehl EDIT LENGTH(1CM)FOR(P); wird die Länge 1 (hier 1 cm) für die Segmente einer gestrichelten Linie vorgegeben (siehe auch EDIT LINETYPE).

LINEWIDTH

Mit diesem Befehl wird die Strichstärke festgelegt.

```
EDIT LINEWIDTH(1mm)FOR(O);
```

OPEN/CLOSED

gibt an, ob ein Polygonzug offen oder geschlossen ist.

```
EDIT OPEN FOR(P);
EDIT CLOSED FOR(P);
```

PEN

Mit dem Befehl EDIT PEN(5)FOR(T); wird für das Objekt T (Text, Punkt, Polygon...) Schreibstift 5 ausgewählt.

COLOR

```
EDIT COLOR(3)FOR(O);
```

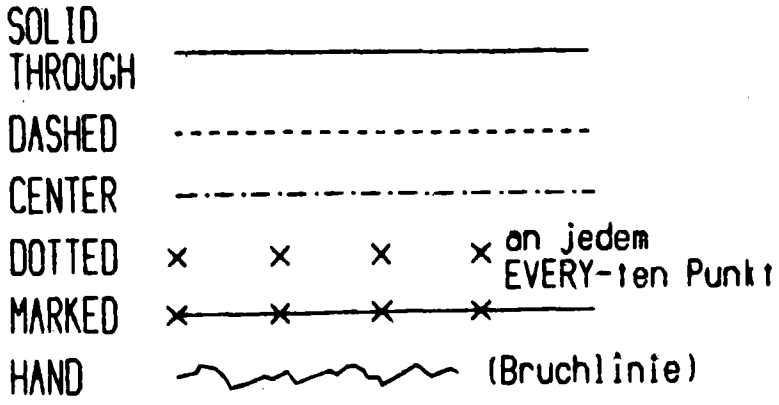
legt fest, daß das Objekt O mit der Farbe 3 gezeichnet wird.

LINETYPE

```
EDIT LINETYPE { SOLID    } FOR(O);
              { THROUGH  }
              { DASHED   }
              { CENTER   }
              { DOTTED   }
              { MARKED   }
              { OMITTED  }
              { HAND     }
```

Für das graphische Objekt (oder Liste von Objekten)O wird die Liniendarstellung durch die obige Angabe ausgewählt. Normal bei Polygonzügen und Kurven sind durchgezogene Linien. LINETYPE empfiehlt sich vor allem in Diagrammen zur Unterscheidung von Linien.

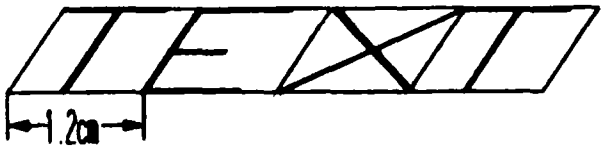
Folgende Kennzeichnung ist möglich:



WIDTH

```
EDIT WIDTH(1.2 CM)OF(T);
```

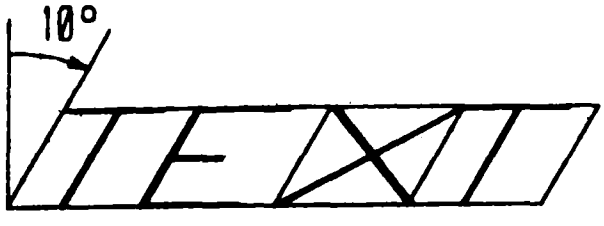
Damit wird die Breite eines Schriftzeichens des Textes T auf 1.2 CM gesetzt.



SLANT

```
EDIT SLANT(10GRAD)FOR(T);
```

Diese Anweisung setzt die Neigung der Schriftzeichen des Textes T auf 10 Grad nach rechts. Negative Winkel bedeuten Neigung nach links.




```
EDIT HEIGHT(8MM)WIDTH(5MM)
     SLANT(15DEG)COLOR(5)FOR(T1,T2);
```

Dies ist ein Beispiel für das Setzen mehrerer Attribute gleichzeitig für die Objekte T1 und T2.

DISTANCE

```
EDIT DIST(1CM)FOR(AX1);
```

DISTANCE ist nur für Achsen sinnvoll, damit wird der Skalenstrichabstand gesetzt.

6.6.3 RESET

Mit der RESET-Anweisung werden die in GIPSY eingebauten Ausgangswerte aller Standardattribute wiederhergestellt (siehe Kap. 4.3.2).

Syntax:

```
RESET { ALL
      | STANDARD  attribut
      | FACTOR
      | ARC
      | UNITS       { LENGTH }
      |             { ANGLE  }
      | PROJEKTION   { PARALLEL }
      |             { CENTRAL  }
      |             { PI_N     }
      |             { ORIGIN   }
      |             { PI_P     }
      | INCREMENT    { LINEAR }
      |             { ANGLE  }
      | INVISIBLE   attribut
      | MAXPEN
      | MAXCOLOR   } [ , { : } ] * ;
```

Beispiel:

```
RESET STANDARD COLOR, ARC, PROJECTION  ORIGIN,PROJECTION PI_P;
```

setzt die Farbe, den Winkel zur Umwandlung von Kreisen in Bögen, Projektionsursprung und Projektionsebenen-Punkt auf die Systemstandardwerte zurück. RESET ALL setzt sämtliche Systemwerte auf die Ursprungswerte zurück.

6.7 Spezielle PLOT-Anweisungen

Das im Kap. 6.3 beschriebene Statement PLOT zur Zeichenausgabe besitzt zwei Erweiterungen, mit denen spezielle Aufgaben gelöst werden können:

- perspektivische Darstellung von 3-dimensionalen Zahlenfeldern (PLOT RELIEF...)
- Schraffieren von Schnittflächen bei Raumelementen (PLOT SHADE...)

Beiden Anweisungen ist gemeinsam, daß keine graphischen Objekte (z.B. Polygonzüge) an das rufende Programm zurückgeliefert werden. Ferner können keine Objekttransformationen (z.B. SHIFT, ROTATE) durchgeführt werden.

6.7.1 PLOT RELIEF

Bei großen Mengen anfallender Ergebnisdaten werden an die Auswertung besondere Anforderungen gestellt. Die Auswertung kann nämlich praktisch nur noch graphisch erfolgen. Handelt es sich bei den Ergebnisdaten um eine Menge von Zahlenwerttripeln (z.B. x- und y-Koordinaten und ein zugehöriger Funktionswert, also Druckverteilung, Temperaturprofil oder Geschwindigkeitsfeld),so ist ein in vielen Fällen geeignetes Hilfsmittel die perspektivische Darstellung des Wertefeldes als Relief. Diese Darstellungsart kann mit GIPSY unter der Option PLOT RELIEF angewandt werden.

Die Reliefdarstellung, bei der in vielen Fällen jeweils über tausend Funktionswerte abgebildet werden, erfüllt demzufolge nicht die rein quantitative Auswertung, sondern in erster Linie die qualitative.
Erfahrungsgemäß bildet gerade die Reliefdarstellung eine gute Möglichkeit zur Fehlersuche bei den berechneten Daten.

Dargestellt werden die Problemwertetripel (x,y,z), die die Dimension (NX, NY) haben, indem für alle NX und/oder NY 3-dimensionale Polygonzüge unter Anwendung eines einfachen Sichtbarkeitskriteriums abgebildet werden.

Statement

Das Statement beginnt mit

```
PLOT RELIEF OF(tripel)..;
```

(genaue Syntaxbeschreibung Kap.9)

Aufbau des Zahlenwerttripels

Das graphische Objekt 'tripel', das die Koordinatenwerte x,y und z des darzustellenden Zahlenfeldes enthält, wird als mindestens 3-fach indiziertes Feld deklariert:

```
DCL tripel (N,M,3) DEC FLOAT(6);
```

Die x-Koordinaten werden in tripel (*,*,1),
die y-Koordinaten in tripel (*,*,2) und
die z-Koordinaten in tripel (*,*,3) abgespeichert (s.Beispiel Kap. 8.3).

Darstellungsart

Die perspektivische Darstellung des vollständigen Zahlenfeldes 'tripel' wird auf die Abbildung von mehreren 3-dimensionalen Polygonzügen zurückgeführt, die Schnittlinien durch das Zahlenfeld sind.

Mehrere Darstellungsarten sind mit der Option RELIEF möglich:

- Schnittlinien durch das Feld, die quer zur x-Richtung verlaufen,

- Schnittlinien durch das Feld, die quer zur y-Richtung verlaufen,
- Schnittlinien in beiden Richtungen,
- Beschränkung auf die sichtbare Reliefoberseite.

Die Grundfläche des Reliefs ist immer die x-y-Ebene, die z-Werte sind immer die Funktionswerte. Schnittlinien, die quer zur z-Richtung verlaufen, können nicht gezeichnet werden (dies kann durch Höhenlinien (ISO) erreicht werden).

Sichtbarkeitskriterium

Die Polygonzüge werden unter Anwendung eines einfachen Sichtbarkeitskriteriums dargestellt. Über die Verdeckung einer Kurve wird ausschließlich in der Bildebene entschieden, da es sich um ein Verfahren handelt, das nur die Grenzen eines Sichtbarkeitsfeldes verwaltet und jede neu zu zeichnende Kurve gegen diese prüft und dann die Berandung aktualisiert.

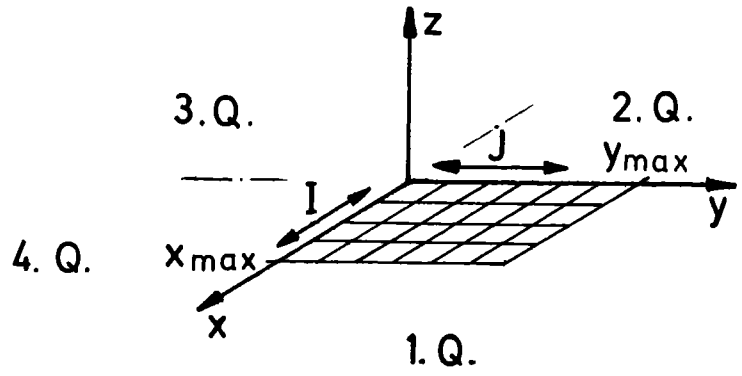
Nur der Teil der so entstandenen Reliefoberseite, aber nicht die erkannbare Reliefunterseite, wird bei der Option LINES ('ABOVE') gezeichnet.

Projektionsrichtung, Indexlaufrichtung

Um eine richtige Anwendung des Sichtbarkeitsalgorithmus im Relief zu gewährleisten, müssen demnach die Problemwerte im Zahlenfeld 'tripel' so geordnet sein, daß die Zählrichtung der Indizes und die Projektionsrichtung gleichgerichtet sind. Nur dann wird bei weiter hinten liegenden Kurven (deren Berandung später aktualisiert wird) die Verdeckung richtig erkannt.

Dieser Sachverhalt möge noch an einem Beispiel verdeutlicht werden.

Das zu zeichnende Relief liege im 1. Quadranten des räumlichen Koordinatenkreuzes (Abb.6.3).



Aus folgender Tabelle ist zu ersehen, bei welcher Projektionsrichtung die Indizes I und J mit ansteigenden oder abnehmenden Koordinaten laufen müssen.

Projektion aus	x_p	y_p	I=1 bei	J=1 bei
1.Quadrant	>0	>0	x_{max}	y_{max}
2.Quadrant	<0	>0	x_{min}	y_{max}
3.Quadrant	<0	<0	x_{min}	y_{min}
4.Quadrant	>0	<0	x_{max}	y_{max}

Tab.6.1:Projektionsrichtung und Indexanordnung

Hierbei sind x_p und y_p die Koordinaten des Raumpunktes, der die Blickrichtung (Lichtquelle) festlegt.

Wird diese genannte Regel nicht eingehalten, so ergeben sich falsche Reliefdarstellungen. Im Beispiel (Kap.8.3) sind die vier Kombinationen von möglichen Index-Laufrichtungen mit den jeweiligen Ergebnissen aufgezeigt.

Skalierung

Um das Relief in der gewünschten Abbildungsgröße zu erhalten, muß meist noch eine Skalierung durchgeführt werden. Sie kann mit der Option

SCALED BY (scal-array)

erfolgen. Hierbei stehen in scal-array(3) die Skalierungsfaktoren für die x_1-, x_2- und die x_3-Richtung. Ihre Größenordnung ist so zu wählen, daß die skalierten Zeichnungskoordinaten in der Einheit Meter (m) im Plot-Fenster dargestellt werden können.

Generierung

In vielen Anwendungsfällen wird Relief benutzt, um die qualitative Verteilung einer Zustandsgröße über einer Bezugsgeometrie aufzuzeigen. Oft werden diese Reliefs zu mehreren Zeitpunkten verlangt.

Besteht die Bezugsgeometrie aus einem äquidistanten Gitter, so bietet die Option

GENERATED BY(meshsize-array)ALONG(directions)

in Verbindung mit SCALED die Möglichkeit, die Koordinaten des Maschengitters zu erzeugen und gleichzeitig die Skalierung für die Zeichnung vorzunehmen. In diesem Fall müssen nur den x_3-Werten des Feldes 'tripel' die Problemwerte (z.B.Druck, Temperatur) zugewiesen werden.

In meshsize-array(3) sind die Maschenweiten in x_1- und x_2-Richtung anzugeben (x_3-Richtung unerheblich), und mit directions wird die Richtung festgelegt, für die die Koordinaten zu generieren sind.

Allgemeine Hinweise

Für die Darstellung eines Reliefs sind i.A. folgende Punkte zu beachten:

- Festlegung der Projektionsrichtung
- Zeilen und Spalten in Projektionsrichtung ordnen
- Verschiebung des Koordinatenursprungs
- Skalierung der Koordinatenwerte.

6.7.2 PLOT SHADE

Bei der Darstellung von 3-dimensionalen Gebilden und deren Durchdringungen ist die Schraffur geeignet, die Schnittflächen zu kennzeichnen. Die 3-dimensionalen Gebilde können Körper (BODY) oder Raumelemente (SPACE) sein, die schneidenden Objekte auch Flächen. Die so entstehenden Flächen oder Ebenen können mit der Erweiterung des PLOT-Statements, nämlich mit PLOT SHADE, schraffiert werden. Die Schnittfläche wird mit Schraffurlinien durchzogen, die der Anwender mit Richtung und Abstand beschreiben kann.

Der Befehl PLOT SHADE führt nur das Schraffieren der Schnittfläche aus. Sollen die zugehörigen Körper oder Raumelemente ebenfalls abgebildet werden, so muß deren Darstellung in einem gesonderten PLOT-Befehl aktiviert werden.

Statement

Das Statement beginnt mit

```
PLOT SHADE OF(space-obj)....;
```

(genaue Syntaxbeschreibung Kapitel 9)

Schnittebene

Wird ein 3-dimensionales Objekt (Typ: SPACE, BODY) von einem anderen 3-dimensionalen Objekt oder einer Ebene durchdrungen, so entsteht eine Schnittfläche, die mit dem Befehl PLOT SHADE....; dargestellt werden kann.

Die Schnittflächen bestehen aus

- den Durchdringungskurven und
- den Schraffurlinien.

Durchdringungslinien

Die Kontur der Durchdringung kann mit einen oder mehreren 3-dimensionalen Polygonzügen dargestellt werden.

Schraffurlinien

Die Schraffurlinien können vom Anwender festgelegt werden, indem er deren Lage und deren Abstand voneinander angibt. Die Lage der Schraffur wird definiert durch eine Ebene, zu der parallel die Schraffurlinien verlaufen. Wird keine Angabe über die Lage gemacht, so verlaufen sie parallel zur x-Achse.
Der Abstand zweier Schraffurlinien kann ebenfalls angegeben werden, der Standard-Wert ist 0.01m.

6.8 Effektivitätshilfen in GIPSY

LOAD, RELEASE

Syntax:

LOAD { ALL | 2D | 3D } ;

RELEASE { ALL | 2D | 3D } ;

Für den dynamischen Aufruf von Modulen in GIPSY (bzw. REGENT überhaupt) werden ca. 1-2 msec benötigt. Bei Rechenläufen mit einer großen Anzahl von Modulaufrufen (einige Hunderttausend oder 10^6) macht sich dies deutlich in einer großen CPU-Zeit bemerkbar. Mit dem Befehl LOAD ergeben sich im Einzelfall Zeitverbesserungen von über 50%, insbesondere bei 3D-Problemen (Unsichtbarkeitsalgorithmus) auf Kosten eines größeren Speicherplatzbedarfs.

LOAD lädt die GIPSY-Module - je nach Typ solche für 2D- bzw. 3D-Graphik oder alle - fest in den Speicher bis zum Auftreten der Anweisung RELEASE, die für diese Module wieder die dynamische Modul-Verwaltung aktiviert. (Siehe hierzu auch die Befehle LOAD, RELEASE im REGENT-Handbuch).

Bei LOAD ALL muß in der derzeitigen Version von GIPSY die Modulpoolsize mindestens 512 K betragen.

6.9 Allokieren von BASED-Objekten

Wenn GIPSY-Objekte, die nicht Teil einer PL/1-Struktur sind, die Speicherklasse (storage class) BASED haben, so können sie nur mit einer speziellen Form der PL/1-Anweisungen

ALLOCATE und
FREE

allokiert und wieder freigegeben werden.

Syntax:

{ ALLOCATE / FREE } type name;

type: GIPSY-Objekttyp wie POINT2, TEXT2, PLANE, POLY oder COLL, siehe Kap. 4.5

name: Name des GIPSY-Objektes.

Alle anderen Formen der Anweisungen ALLOCATE und FREE werden als PL/1-Anweisungen interpretiert.

Beispiel:

```
DCL POL1 POLY(30) BASED(Q);
ALLOC POLY POL1;
  :
  :
FREE POLY POL1;
```

Man beachte, daß bei BASED-Objekten, ebenso wie bei normalen PL/1-Datentypen, nur feste Längen und Dimensionen zulässig sind.

Falsch ist also:

```
DCL P2 POLY(N);      und
DCL P3(M) POLY(10);
```

7. Die GIPSY-Umwelt

7.1 Einbettung in PL/1

Die GIPSY-Sprache ist eine Erweiterung von PL/1. Damit sind alle von dieser Programmiersprache gebotenen Möglichkeiten auch dem GIPSY-Anwender verfügbar. Dazu gehören insbesondere die Möglichkeit, Variable und Ausdrücke statt Konstanten zu verwenden, Ein/Ausgabeanweisungen, Abfragen, Verzweigungen und Schleifen zu benutzen und mehrfach auszuführende Anweisungen in Unterprogramme zusammenzufassen.

7.1.1 Variable und Ausdrücke

An all den Stellen in GIPSY-Anweisungen, an denen nichtgraphische Werte (Real- oder Integerzahlen, Zeichenketten) stehen können, sind auch PL/1-Variable oder Ausdrücke zulässig. Dabei gelten die PL/1-Regeln unverändert. Bei Längen- und Winkelangaben kann auf die Variable oder den Ausdruck die Maßeinheit folgen.

Beispiel:

Anweisung mit Konstanten:

```
VIEW ORIGIN(0.4 M,0.2 M)
     SCALE(0.5,0.3);
```

Anweisung mit Ausdrücken und Variablen:

```
VIEW ORIGIN(XO M,YO M)
     (SCALE((X2-X1)/(XMAX-XMIN),(Y2-Y1)/(YMAX-YMIN));
```

Für die Verwendung graphischer Objekte gibt es dagegen einige Einschränkungen: Die Variablennamen für graphische Objekte (POINT, BALL, TEXT2, etc.) dürfen höchstens 20 Zeichen lang sein und nicht mit den Zeichen '#'oder 'Q' beginnen (auch andere vom Anwender benutzte Variablen dürfen nicht mit '##' oder 'Q' beginnen). Die Namen graphischer Builtin-Funktionen (INTERSECT,

SPACE, POLY. etc.) dürfen nicht als Variablennamen verwendet werden. Werden diese Regeln verletzt, erfolgt eine Fehlermeldung des REGENT-Übersetzers.

Graphische Objekte dürfen nur an zwei Stellen in Nicht-GIPSY-Anweisungen stehen: In der Parameterliste der OTYP und COORD-Builtin-Funktionen (siehe 5.3) und in der Parameterliste eigener GIPSY-Unterprogramme (in diesem Abschnitt beschrieben).

7.1.2 Die Lebensdauer von GIPSY-Objekten

Zur Erläuterung der Lebensdauer von GIPSY-Objekten sei zunächst an die Lebensdauer von PL/1-Daten erinnert.

- Daten der Speicherklasse AUTOMATIC existieren bis zum Verlassen des Blockes (BEGIN= oder PROCEDURE-Block), in dem sie vereinbart sind.
- Daten der Speicherklassen BASED und CONTROLLED existieren von der Ausführung der ALLOCATE-Anweisung bis zur Ausführung der zugehörigen FREE-Anweisung.

Im Zusammenhang mit der Lebensdauer muß auch die Adressierbarkeit (Möglichkeit des Zugriffs, Verwendbarkeit) beachtet werden.

- Daten der Speicherklasse AUTOMATIC und CONTROLLED sind in dem Block, in dem sie vereinbart sind, und in allen darin statisch enthaltenen Blöcken adressierbar. Hinzu kommt die Adressierbarkeit in Prozeduren, an die die Daten als Parameter übergeben wurden.
- Daten der Speicherklasse BASED sind adressierbar, solange ein POINTER existiert und adressierbar ist, der auf diese Daten zeigt. Daten dieser Klasse können also auch noch existieren nachdem sie ihre Adressierbarkeit verloren haben.

(Anmerkung: Die Attribute STATIC, CONTROLLED und EXTERNAL sind für GIPSY-Objekte verboten und daher hier nicht relevant).

Die meisten GIPSY-Objekte richten sich nach diesen Regeln. Ausnahmen hiervon sind Objekte vom Typ

COLLECTION, SPACE und BODY.

Der Sachverhalt sei für COLLECTION eingehend erläutert. Für SPACE und BODY gilt entsprechendes.

Eine COLLECTION ist eine (geordnete) Menge anderer GIPSY-Objekte. Da bei der Vereinbarung eines Kollektionsobjektes Art und Anzahl der Kollektionselemente noch nicht bekannt sind, baut die Realisierung einer Kollektion auf einer verzeigerten Datenstruktur auf. Eine Kollektion besteht demzufolge aus zwei Bestandteilen:

- die Kollektion selbst, welche die in der Deklaration festgelegte Speicherklasse hat
- die Elemente der Kollektion, die stets die Speicherklasse BASED haben.

Die Elemente einer Kollektion verlieren ihre Adressierbarkeit mit der Adressierbarkeit der Kollektion selbst. Bei AUTOMATIC ist das beim Verlassen des Blockes, bei BASED beim FREE. Die Kollektionselemente (sofern noch vorhanden) würden jedoch über diesen Zeitpunkt hinaus noch weiter existieren(und Speicherplatz belegen!). Aus diesem Grunde muß der GIPSY-Anwender folgende Regel beachten:

> Kollektionen müssen durch die EMPTY-Operation geleert werden, bevor sie ihre Adressierbarkeit verlieren.

(siehe auch Kap. 4.8).

7.1.3 Ein-Ausgabeanweisungen

Die Übernahme von Daten aus anderen Programmen oder von Meßwerten in GIPSY ist mit Hilfe von PL/1-Ein/Ausgabeanweisungen problemlos möglich.

Beispiel: Einlesen von Dreieckelementen eines Finite-Element-Programms und perspektivische Abbildung mit GIPSY.

```
GET LIST(N);                  N = ANZAHL DER ELEMENTE
DO I=1 TO N;                  FUER ALLE ELEMENTE
 GET LIST(((X(K,J) DO K=1 TO 3) DO J=1 TO 3));
                              ELEMENTKOORDINATEN EINLESEN IN X(K,J)
 CALL SCHWERPUNKT(X,XS);      SCHWERPUNKT BERECHNEN
 SET PPP=POLY(                POLYGONZUG BILDEN
     POINT(X(1,1),X(1,2),X(1,3)),
     POINT(X(2,1),X(2,2),X(2,3)),
     POINT(X(3,1),X(3,2),X(3,3)));
 EDIT CLOSED OF(PPP);         GESCHLOESSENER POLYGONZUG
 PLOT(PPP,TEXT(               ZEICHNE ELEMENT UND SEINE NUMMER
      POINT(XS(1),XS(2),XS(3)),I));   IN DEN SCHWERPUNKT
END;
```

Die Schwerpunktsberechnung mittels eines PL/1-Unterprogramms wird leicht durch die PL/1-Fähigkeiten ermöglicht.

7.1.4 Unterprogramme, nicht-graphisch

Unterprogramme ohne GIPSY-Anweisungen können in der gleichen Weise wie in PL/1 verwendet werden, als interne und externe Prozeduren. Interne Prozeduren können innerhalb von ENTER-END GIPSY stehen und außerhalb. Sowohl eigentliche Prozeduren als auch Funktionsprozeduren sind möglich.

Beispiel:

```
BEISPIEL:
M: PROC OPTIONS(MAIN) REGENT;
 DCL SUB1 ENTRY(BIN FIXED(15)) EXT;     EXTERN AUSSERHALB GIPSY DEKLA-
                                        RIERT,EIGENTLICHE PROZEDUR

 ENTER GIPSY;
  SUB2: PROC(K);                        INTERN INNERHALB GIPSY DEKLA-
                                        RIERT,EIGENTLICHE PROZEDUR

  ..............
  END SUB2;

  CALL SUB1(5);
  CALL SUB2(J);                         AUFRUFE
  P1=SUB3(P);
  C=SUB4;

  DCL SUB3 ENTRY(PTR) RETURNS(PTR) EXT; EXTERN,INNERHALB GIPSY DEKLA-
                                        RIERT,FUNKTIONSPROZEDUR
 END GIPSY;

 SUB4: PROC RETURNS(CHAR(4));           INTERN,AUSSERHALB GIPSY DEKLA-
  .......                               RIERT,FUNKTIONSPROZEDUR.
  RETURN(...);
 END SUB4;

 FINISH;
END M;
```

7.1.5 Unterprogramme mit GIPSY-Anweisungen

GIPSY-Unterprogramme können interne und externe Prozeduren und getrennte Module sein. Sie können entweder für den Aufruf zwischen ENTER und END GIPSY oder für einen Aufruf von außerhalb GIPSY erstellt sein. Für alle Unterprogramme, die GIPSY-Anweisungen enthalten, gelten folgende Regeln für die Parameterübergabe graphischer Objekte:

- Als Funktionswert von Funktionsprozeduren sind graphische Objekte verboten.
  "DCL.....ENTRY RETURNS(POINT)" ist also ebenso unzulässig wie "RETURN(TEXT2)". Die betreffende Aufgabenstellung kann aber durch eine eigentliche Prozedur mit Ein- und Ausgabe parametern erreicht werden. Numerische Werte als Funktionswerte von GIPSY-Funktionsprozeduren sind möglich.

- Graphische Objekte, die Parameter sind, müssen im Unterprogramm mit dem Schlüsselwort PARAMETER deklariert werden.
  "A:PROC(P); DCL P POINT;" reicht nicht, es muß heißen:
  "A:PROC(P); DCL P POINT PARAMETER;".

- An Unterprogramme können als Parameter sowohl graphische Ausdrücke (graphische Builtin-Funktionen) übergeben werden, als auch namentlich identifizierte graphische Objekte:

```
CALL A(POINT(X,Y,Z));  oder
DCL PPP POINT;
SET PPP=POINT(X,Y,Z);
CALL A(PPP);
```

- Graphische Objekte sind in der ENTRY-Deklaration einer externen Prozedur mit dem Attribut PTR zu deklarieren.

Bei Verletzung dieser Regeln erfolgen Fehlermeldungen des PL/1-Compilers (siehe weiter hinten in diesem Abschnitt unter Fehlermeldungen).

7.1.6 Interne GIPSY-Prozeduren

Interne Unterprogramme, die GIPSY-Anweisungen enthalten, müssen sowohl innerhalb ENTER-END GIPSY definiert als auch aufgerufen werden. Funktionsprozeduren, die einen numerischen Wert liefern, sind möglich.

Beispiel:

```
ENTER GIPSY;
 SUB1: PROC(T,P,P1);
  DCL T TEXT2 PARAMETER;                          DEFINITION SUB1
  DCL (P,P1) POINT2 PARAMETER;
  PLOT(T,P);
  SET P1=SHIFT2(P,2,5);
 END SUB1;

 SUB2: PROC(P) RETURNS(DEC FLOAT(6));
  DCL P POINT PARAMETER;
  RETURN(COORD(P,1));                             DEFINITION SUB2
 END SUB2;

 DCL TT TEXT2(20);
 DCL (PP,PP1) POINT2;
 SET PP=....
 SET TT=.....
 DCL PPP POINT;
 SET PPP=........
 CALL SUB1(TT,PP,PP1);                           AUFRUFE
 X=SUB2(PPP);

END GIPSY;
```

7.1.7 Externe GIPSY - Prozeduren

Externe Prozeduren, die GIPSY-Anweisungen enthalten, werden mit dem REGENT-Modulgenerator erzeugt. Folgende Anweisungen sind erforderlich (ab Spalte 1 kodieren):

```
//  JOB                                   JOB-Karte
//  EXEC    QQPCL                         Modulgenerator-Aufruf
//P.SYSIN   DD   *
*SUBSYSTEM GIPSY;                         SUBSYSTEM IST GIPSY
*ROUTINE NAME LIB(DDNAME);
*PROCESS REGENT;
 NAME: PROC ......                        ES FOLGT EINGABE IN
                                          EINER PROBLEMORIENTIERTEN
                                          SPRACHE.(HIER GIPSY)
 ..........
 END NAME;
```

Die Modulgenerator-Eingabe ist im REGENT-Handbuch / 7 / näher beschrieben. Für den GIPSY-Anwender sollten jedoch die in diesem Abschnitt gegebenen Angaben ausreichen. "name" ist der Name der externen Prozedur, er darf maximal 6 Zeichen lang sein. "ddname" ist der DD-Name der Datei, auf die die externe Prozedur nach der Übersetzung und dem Binden geladen werden soll. Diese Datei entspricht der mit dem DD-Namen SYSLMOD angesprochenen Datei in einem normalen Linkage-Editor-Lauf. Beim Aufruf der externen Prozedur muß die Datei "ddname" hinter SYSLIB im REGENT-G-Step verkettet werden.

Der erste Parameter aller externen GIPSY-Routinen muß QQ sein. Der Modulgenerator fügt diesen Parameter automatisch ein, falls er fehlt. Beim Aufruf jedoch erfolgt dieses automatische Einfügen nicht, daher muß der Anwender sowohl in der ENTRY-Deklaration als auch im CALL die Pointervariable QQ berücksichtigen:

Definition:

```
name:  PROC(QQ,parameter  )....;
```

QQ wird eingefügt, wenn es fehlt

Aufruf:

```
DCL name ENTRY(PTR,parameterattribute)EXT;
CALL name (QQ,parameter   );
```

Externe Prozeduren können zwischen ENTER und END GIPSY deklariert und aufgerufen werden, dann muß bei der Erzeugung der Routinen folgendes beachtet werden:
Die erste Karte muß die REGENT-Option REGENT(SUB=GIPSY) enthalten, weitere REGENT-Optionen werden ignoriert. Graphische Objekte als Parameter sind erlaubt, ENTER GIPSY und END GIPSY sind in der externen Prozedur verboten.

Definition:

```
name:  PROC(QQ,graphische+sonstige Parameter)
       REGENT(SUB=GIPSY);
   DCL (graphische Parameter) typ PARAMETER;
   GIPSY+PL/1-Anweisungen
 END name;
```

Aufruf:

```
ENTER GIPSY;
        /* Für QQ, für alle graphischen Parameter */
DCL name ENTRY(PTR,...,PTR,...)EXT;
CALL name (QQ,graphische+sonstige Objekte);
END GIPSY;
```

Externe Prozeduren können aber auch ENTER und END GIPSY enthalten. Dann müssen sie von außerhalb GIPSY aufgerufen werden. Die Übergabe graphischer Objekte als Parameter ist nicht möglich. In der REGENT-Option muß NOINIT angegeben werden.

Definition:

```
name: PROC(QQ,numerische Parameter)
      REGENT(NOINIT);
  ENTER GIPSY;
    GIPSY+PL/1-Anweisungen
    PL/1-Anweisungen
  END GIPSY;
END name;
```

Aufruf:

```
DCL name ENTRY(PTR,Parameterattribute)EXT;
CALL name(QQ, numerische Parameter);
```

Es ist auch möglich, in einer externen Prozedur ein ENTER GIPSY und am Ende ein END GIPSY LEAVE zu machen, in weiteren Prozeduren jeweils ein ENTER GIPSY REENTER und END GIPSY LEAVE und in einer zuletzt aufgerufenen Prozedur ein ENTER GIPSY REENTER und ein END GIPSY. Diese Methode ist im Abschnitt 7.4 näher beschrieben. Externe Prozeduren können auch Funktionsprozeduren sein. Die folgenden Beispiele demonstrieren die Verwendung externen Prozeduren.

```
//IRE147EG JOB (0147,330,POH1A),ENDERLE,REGION=480K,TIME=(,30)
// EXEC QQPCL                                  JOB-Karte
//P.SYSIN DD *                                 Modulgenerator-Aufruf
*SUBSYSTEM GIPSY;
*ROUTINE GGG1 LIB(MYLIB);                      "name"=GGG1, "ddname"=MYLIB
*PROCESS REGENT;
 GGG1:PROC(XXX,YYY,P) REGENT(SUB=GIPSY);
 DCL (XXX,YYY) DEC FLOAT(6);                                  externe
 DCL P POINT2 PARAMETER;       GIPSY-Parameter                Prozedur
 DCL A POLY2(2);                                              zum Aufruf
 SET A=LINE(POINT2(XXX,YYY),P);                               aus GIPSY
 OPEN PLOT DIN A(4);           GIPSY-Anweisungen
 PLOT(A);
 PLOT (LINE(POINT2(0.02,0.02),POINT2(0.1,0.12)));
 END GGG1;
*ROUTINE GGG2 LIB(MYLIB);
*PROCESS REGENT;
 GGG2: PROC(XXX,YYY) REGENT(NOINIT);
 DCL (XXX,YYY)(*) DEC FLOAT(6);
 ENTER GIPSY;
 DCL I BIN FIXED(15);
 I=HBOUND(XXX,1);
 BEGIN;                                                       externe
    DCL A POLY2(I);                                           Prozedur
    DCL K COLL;                                               zum Aufruf
    DO J=1 TO I;                                              von ausserhalb
       SET K=COLL(K,POINT2(XXX(J),YYY(J)));                   GIPSY
    END;
    SET A=POLY2(K);
    OPEN PLOT DIN A(4);
    PLOT(A);
 END ;
 END GIPSY;
 END GGG2;
//P.MYLIB DD DSN=TSO147.FLUSHLIB.LOAD,DISP=SHR    Bibliothek für
// EXEC REGENT                                    Prozeduren
//P.SYSIN DD *                                    Aufruf REGENT
 GGG3: PROC OPTIONS(MAIN) REGENT(NODA,PLOT=STATOS,MPOOL=100000);
 ENTER GIPSY;                                    Deklaration GGG1
 DCL GGG1 ENTRY(PTR,DEC FLOAT(6),DEC FLOAT(6),PTR) EXT;
 DCL PP POINT2;
 SET PP=POINT2(0.03,0.03);
 CALL GGG1(QQ,0.18,0.01,PP);                     Aufruf GGG1
 END GIPSY;
 DCL GGG2 ENTRY(PTR,(*)DEC FLOAT(6),(*)DEC FLOAT(6)) EXT;
 DCL (XXX,YYY)(10) DEC FLOAT(6);                 Deklaration GGG2
 DO I=1 TO 10;                                   ausserhalb GIPSY
    XXX(I)=I*0.01E0+2.0E0;
    YYY(I)=(I<=5)*(0.08E0+I*0.01E0)+(I>5)*(0.16E0-I*0.01E0);
 END;
 CALL GGG2(QQ,XXX,YYY);                          Aufruf GGG2
 MESSAGE ACTIVE DEBUG;
 FINISH;
END GGG3;                                        Bibliothek für
//G.SYSLIB DD                                    Routinen hinter
// DD DSN=TSO147.FLUSHLIB.LOAD,DISP=SHR          SYSLIB verkettet
//G.PLOTTAPE DD UNIT=T0800,VOL=SER=Y147EG,DCB=DEN=2,LABEL=(,NL)
```

Externe GIPSY-Prozeduren können auch hinter dem Hauptprogramm in der REGENT-Prozedur liegen. Sie werden dann durch '*PROCESS;' voneinander getrennt. Die Regeln für die Definition und den Aufruf sind die gleichen wie bei der Prozedurübersetzung mit dem Modulgenerator.Beispiel:

```
// EXEC REGENT
//P.SYSIN DD *
    MAIN: PROC OPTIONS(MAIN) REGENT(NODA);
    :
    END MAIN;
*PROCESS;
    SUB1: PROC(A,B,C) REGENT(SUB=GIPSY);
    :
    END SUB1;
*PROCESS;
    SUB2: PROC REGENT(SUB=GIPSY);
    :
    END SUB2;
//
```

7.1.8 Module

Module sind externe Prozeduren, die nicht statisch in das Anwenderprogramm gebunden werden (durch den Linkage Editor oder Loader), sondern erst beim Aufruf dynamisch in den Arbeitsspeicher geladen werden. Die folgende Anleitung zum Erzeugen und Aufrufen von Moduln ist nur für den fortgeschrittenen GIPSY-Anwender gedacht. Module sind nur dann externen Routinen vorzuziehen, wenn mehrere von ihnen in einem Anwendungsprogramm nicht sehr häufig (bis einige 1000 mal) abwechselnd aufgerufen werden und der verfügbare Speicherplatz für alle Module nicht ausreicht. Module werden durch den REGENT-Modulgenerator erzeugt und durch die REGENT-Modulverwaltung während des Programmlaufes verwaltet. Das Laden von einer Bibliothek, der Aufruf und das Festhalten oder Überschreiben im Arbeitsspeicher werden von der Modulverwaltung übernommen. Der etwas umständliche Modulaufruf in GIPSY, der im folgenden beschrieben wird, wird durch eine REGENT-Erweiterung in Zukunft vereinfacht. Auch Module können entweder von außerhalb ENTER-END GIPSY oder von innerhalb gerufen werden, je nachdem wozu sie vorgesehen sind.

Module zum Aufruf innerhalb GIPSY:

Definition:

```
//    JOB
// EXEC QQPCL
//P.SYSIN DD *
*SUBSYSTEM GIPSY;
*MODULE name LIB(ddname);
*PROCESS REGENT;
 name:PROC(parameter) REGENT(SUB=GIPSY);
    .
    .
    GIPSY- und PL/1-Anweisungen
    .
    .
 END name;
```

Aufruf:

```
ENTER GIPSY;
DCL 1 name_CB,
  2 name_P PTR INIT(ADDR(name_P)),
  2 name ENTRY(PTR,parameterattribute) INIT(QQMLK1),
  2 name_E CHAR(8) INIT('name'),
  2 name_M CHAR(8) INIT('GIname');
CALL name(name_P,parameter);
END GIPSY;
```

Module zum Aufruf von außerhalb GIPSY:

Definition:

```
//    JOB
// EXEC QQPCL
//P.SYSIN DD *
*SUBSYSTEM GIPSY;
*MODULE name LIB(ddname);
*PROCESS REGENT;
 name: PROC(parameter) REGENT(NOINIT);
 ENTER GIPSY;

    GIPSY- und PL/1-Anweisungen

 END GIPSY;
 END name;
```

Aufruf:

```
M: PROC OPTIONS(MAIN) REGENT(NODA,PLOT=...,MPOOL=100000);
DCL 1 name_CB,
  2 name_P PTR INIT(ADDR(name_P)),
  2 name ENTRY(PTR,parameterattribute),
  2 name_E CHAR(8) INIT('name'),
  2 name_M CHAR(8) INIT('GIname');
name=QQMLK1;
CALL name(name_P,parameter);
FINISH;
END M;
```

"name" ist dabei der Modulname, er darf höchstens 6 Zeichen lang sein. "ddname" ist der DD-Name der Datei, auf die der übersetzte und gebundene Modul geladen werden soll (entspricht SYSLMOD in einem normalen Linkage-Editor-Lauf). Diese Datei muß bei der Anwendung des Moduls im REGENT-G-Step hinter STEPLIB verkettet werden. Die folgenden Beispiele demonstrieren die Erzeugung und Anwendung von Modulen, die GIPSY-Anweisungen enthalten:

```
//IRE147EG JOB (0147,330,P0H1A),ENDERLE,REGION=980K,TIME=(,30)
// EXEC QQPCL                                          Modulgenerator-
//P.SYSIN DD *                                        aufruf
*SUBSYSTEM GIPSY;
*MODULE MMM1 LIB(MYLIB);                              eigene Bibliothek
*PROCESS REGENT;
 MMM1:PROC(XXX,YYY,P) REGENT(SUB=GIPSY);
 DCL (XXX,YYY) DEC FLOAT(6);
 DCL P POINT2 PARAMETER;        GIPSY-Parameter
 DCL A POLY2(2);                                      Modul MMM1
 SET A=LINE(POINT2(XXX,YYY),P);                       zum Aufruf in
 OPEN PLOT DIN A(4);            GIPSY-Anweisungen     GIPSY
 PLOT(A);
 PLOT (LINE(POINT2(0.02,0.02),POINT2(0.1,0.12)));
 END MMM1;
*MODULE MMM2 LIB(MYLIB);
```

```
*PROCESS REGENT;
 MMM2: PROC(XXX,YYY) REGENT(NOINIT);
 DCL (XXX,YYY)(*) DEC FLOAT(6);
 ENTER GIPSY;
 DCL I BIN FIXED(15);
 I=HBOUND(XXX,1);                                    Modul MMM2
 BEGIN;                                              zum Aufruf
    DCL A POLY2(I);                                  von ausserhalb
    DCL K COLL;                                      GIPSY
    DO J=1 TO I;
       SET K=COLL(K+POINT2(XXX(J),YYY(J)));
    END;
    SET A=POLY2(K);
    OPEN PLOT DIN A(4);
    PLOT(A);
 END ;
 END GIPSY;
 END MMM2;
//P.MYLIB DD DSN=TSO147.FLUSHLIB.LOAD,DISP=SHR     eigene Bibliothek
// EXEC REGENT                                     Aufruf REGENT
//P.SYSIN DD *
 MMM3: PROC OPTIONS(MAIN) REGENT(NODA,PLOT=STATOS,MPOOL=100000);
 ENTER GIPSY;
 DCL 1 MMM1_CB,
   2 MMM1_P PTR INIT(ADDR(MMM1_P)),
   2 MMM1 ENTRY(PTR,DEC FLOAT(6),DEC FLOAT(6),PTR) INIT(QQMLK1),
   2 MMM1_E CHAR(8) INIT('MMM1'),
   2 MMM1_M CHAR(8) INIT('GIMMM1');              Deklaration MMM1
 DCL PP POINT2;
 SET PP=POINT2(0.03,0.03);
 CALL MMM1(MMM1_P,0.18,0.01,PP);                 Aufruf MMM1
 END GIPSY;
 DCL (XXX,YYY)(10) DEC FLOAT(6);
 DCL 1 MMM2_CB,
   2 MMM2_P PTR INIT(ADDR(MMM2_P)),
   2 MMM2 ENTRY(PTR,(*)DEC FLOAT(6),(*)DEC FLOAT(6)),
   2 MMM2_E CHAR(8) INIT('MMM2'),                Deklaration MMM2
   2 MMM2_M CHAR(8) INIT('GIMMM2');
 MMM2=QQMLK1;
 DO I=1 TO 10;
    XXX(I)=I*0.01E0+2.0E0;
    YYY(I)=(I<=5)*(0.08E0+I*0.01E0)+(I>5)*(0.16E0-I*0.01E0);
 END;
 CALL MMM2(MMM2_P,XXX,YYY);                      Aufruf MMM2
 MESSAGE ACTIVE DEBUG;
 FINISH;
 END MMM3;
//G.STEPLIB DD                                    Eigene Bibliothek
// DD                                             hinter STEPLIB
// DD DSN=TSO147.FLUSHLIB.LOAD,DISP=SHR           verkettet
//G.PLOTTAPE DD UNIT=T0800,VOL=SER=S147EG,DCB=DEN=2,LABEL=(,NL)
```

7.1.9 Fehlermeldungen

Außer den Fehlernachrichten des Subsystems GIPSY selbst (siehe Abschnitt 10), können auch Fehlermeldungen von anderen Quellen bei einer GIPSY-Anwendung auftreten:

a) Bei der REGENT-Übersetzung:

1) Fehlermeldungen des REGENT-Übersetzers

Die Fehlermeldungen des REGENT-Übersetzers liegen auf einer Ebene tiefer als die GIPSY-Meldungen und sind daher für den GIPSY-Anwender nicht immer verständlich.
Meldungen wie:

"SUBSYSTEM NOT FOUND"

nach der Anweisung

"ENTER DIPSY"

sind selbsterklärend. Ansonsten muß die betreffende GIPSY- oder PL/1-Anweisung auf Fehlerfreiheit überprüft werden.

2) Fehlermeldungen des PL/1-Laufzeitsystems

(Der REGENT-Übersetzer ist selbst ein PL/1-Programm)

Dies sollte nicht auftreten. Wenn es doch geschieht und der Fehler ist vom Anwender nicht zu beseitigen, siehe 10).

3) OS-Completion-Codes

Bei I/O-Error Job nochmal abgeben, wenn der Fehler bleibt, muß evtl. die betreffende Datei kopiert werden (Backup der REGENT- und GIPSY-Bibliotheken auf Band ist eine gebotene Maßnahme).

Bei Zeitüberschreitung, die auch bei CPU-Zeit-Verlängerung bleibt: SYSPRINT entblocken und die letzte ausgegebene Anweisung und die folgenden auf Fehlerfreiheit überprüfen. Bei anderen nichtbehebbaren Fehlern: siehe 10).

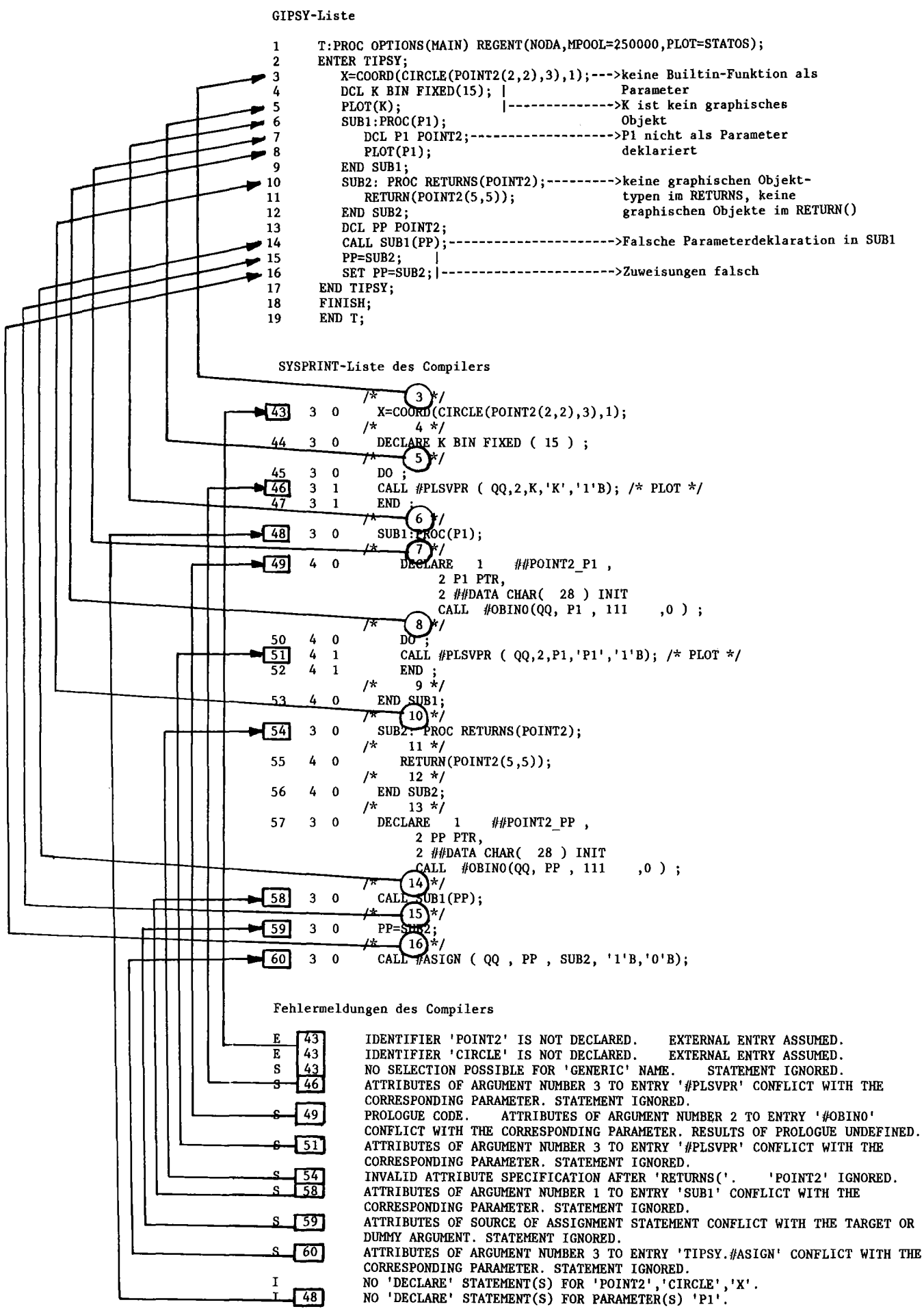


Abb. 7.1: Fehlermeldungen des Compilers

b) Bei der PL/1-Compilation

4) Fehlermeldungen des PL/1-Compilers

Außer vom Benutzer durch falsche PL/1-Anweisungen produzierte Compilermeldungen können auch fehlerhafte GIPSY-Anweisungen, die weder REGENT noch GIPSY bei der Übersetzung beanstanden, zu fehlerhaftem PL/1 führen. Die SYSPRINT-Liste des Compilers enthält in Kommentaren die Statement-Nummern der GIPSY-Liste, so daß leicht zurückverfolgt werden kann, welche GIPSY-Anweisung den Fehler verursacht. Diese Anweisung muß auf Fehlerfreiheit überprüft werden. Einige typische Beispiele siehe Abb. 7.1

5) OS-Completion-Codes

Sehr selten, dann meist ein Fall für den zuständigen Systemprogrammierer. In Ausnahmefällen: siehe 10).

c) Bei der Ausführung (REGENT-G-Step)

6) Fehlermeldungen des OS-Loaders

OS-Loader-Meldungen "UNRESOLVED EXTERNAL REFERENCE" können Folgefehler von falsch angewandten GIPSY-Builtin-Funktionen sein. Oder es werden externe Prozeduren (GIPSY oder Nicht-GIPSY) benutzt, die es nicht gibt oder deren Bibliothek nicht korrekt hinter SYSLIB verkettet ist. Bei unlösbaren Problemen siehe 10).

7) Fehlermeldungen des REGENT-Kernes

Die Fehlermeldungen des REGENT-Systemkernes sind im allgemeinen selbsterklärend. Sie sind im REGENT-Handbuch noch weiter erläutert. Die Fehlerbehebung richtet sich nach der Meldung. Siehe auch 10).

8) Fehlermeldungen des PL/1-Laufzeitsystems

Dies sollte nicht auftreten, wenn doch, siehe 10).

9) OS-Completion-Codes

Verfahren je nach dem auftretenden Code. Bei Zeitüberschreitung, die auch durch CPU-Zeit-Verlängerung nicht behebbar ist,

durch Test-PUTS die betreffende Anweisung herausfinden und vorher MESSAGE ACTIVE DEBUG einschalten. Ursache kann evtl. ein unendlich ausgedehnter Körper sein. In jedem Fall 10).

10) Unbehebbare Fehler

Die Gruppe, die REGENT und GIPSY wartet, bittet alle Anwender im Interesse einer laufenden Verbesserung des Systems um folgende Informationen:

- Fälle in denen trotz korrekter Anwendung von GIPSY laut Handbuch Fehler auftreten (außer sie wurden von außerhalb REGENT/GIPSY verursacht).
- Fälle in denen bei fehlerhafter Anwendung von GIPSY nicht sinnvolle, unklare oder katastrophale Fehlerreaktionen auftraten. (Beispiel: beim Zeichnen eines Kegels, dessen Spitze und Mittelpunkt auf dem gleichen Punkt gelegt wurden, tritt beim Zeichnen ZERODIVIDE auf).
- In allen Fällen, in denen Fehler vom Anwender nicht behoben oder Fehlerursachen nicht gefunden werden, ist die Gruppe REGENT/GIPSY nach den gegebenen Möglichkeiten zur Unterstützung bereit.

In allen diesen Fällen bitte mitliefern:

- Komplette Listen aller angesprochenen Programme einschließlich der SYSPRINT-Liste des Compilers mit den Optionen SOURCE, A, AG, XREF.
- Liste aller anfallenden Dumps(SYSUDUMP, PLIDUMP). Bei PL/1-Laufzeitfehlern im REGENT-G-Step einen PLIDUMP erzeugen durch ON ERROR CALL PLIDUMP('TFCB');
- Bei Fehlern im G-Step wenn möglich die GIPSY Objekte vorher mit PRINT und den Systemzustand mit STATUS ausdrucken. Vor dem fehlerhaften Statement mit TRACE den Kontrollausdruck einschalten.
- Programmeingabe in maschinenlesbarer Form (Karten, Band, Datei).

7.2 Die REGENT-Graphik-Schnittstelle

Die Ausgabe von Zeichnungen durch GIPSY und REGENT ist naturgemäß abhängig von der Art der vorhandenen Hardware und der dazugehörenden Software. Jedoch gibt es eine Reihe von Herstellern von Zeichenmaschinen, deren Grundsoftware weitgehend standardisiert ist. Diese Zeichenroutinen werden durch CALL aufgerufen. Namen und Argumentlisten sind einander angepaßt. Daher werden die Routinen der Calcomp-Basis-Software von REGENT unterstützt und auch von GIPSY bentutzt. Dieses Kapitel bezieht sich in den Einzelheiten auf die REGENT-Version, die im KfK implementiert ist.

7.2.1 REGENT-Option PLOT und FINISH-Anweisung

Die Ausgabe von Zeichnungen bei der Subsystemanwendung wird durch den PLOT-Parameter der REGENT-Option auf der PROCEDURE-Anweisung des Anwenderprogramms gesteuert.

PLOT=CALCOMP oder PLOT

bedeutet Ausgabe auf Calcomp-Plotter.

$$\text{PLOT} = \left\{ \begin{array}{l} \text{XYNETICS} \\ \text{STATOS} \\ \text{TEKTRONIX} \end{array} \right\}$$

bedeutet Ausgabe auf Xynetics, Statos-Plotter oder Tektronix-Display. TEKTRONIX ist nur sinnvoll für Programme, die interaktiv auf einem Tektronix-Bildschirm laufen. Die Liste der Geräte kann erweitert werden.

NOPLOT

erlaubt keine Zeichenausgabe, ein Aufruf von Zeichenroutinen bewirkt eine Fehlermeldung.
NOPLOT ist der Standardwert. Wenn lediglich auf den Plotfile ausgegeben werden soll, ist die Option PLOT nicht erforderlich.

Alle Anwenderprogramme, die Zeichnungen erzeugen, müssen als letzte Anweisung

FINISH;

enthalten.Erst dann erfolgt der Abschluß der PLOT-Dateien. Das Fehlen der FINISH-Anweisung führt zu folgenden Fehlern: Die Zeichnung ist unvollständig (das letzte Bild fehlt) oder gar nicht vorhanden. Es erscheint dann nur der Jobname und die Zeit auf der Zeichnung. Der Plotfile wird nicht korrekt abgeschlossen, der letzte Satz fehlt, was zu einer fehlerhaften oder unvollständigen Verarbeitung des letzten Bildes beim Interpretieren führen kann.

7.2.2 Mehrere Schichten für die Graphik in REGENT

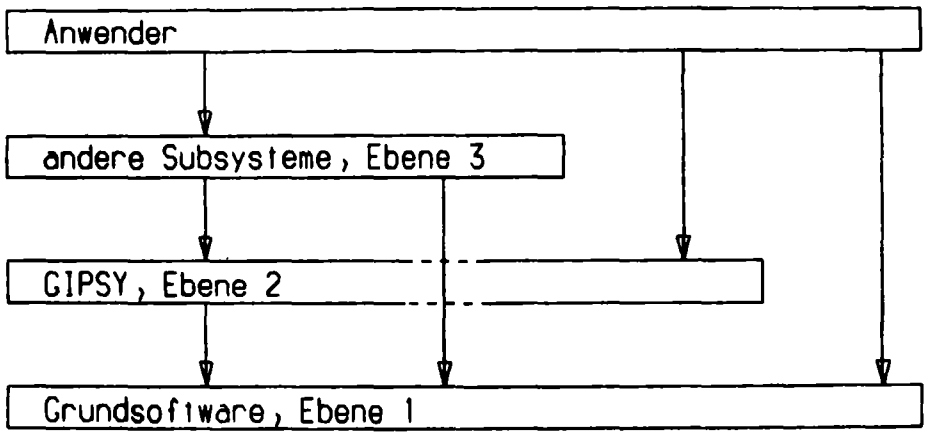
REGENT unterstützt die graphische Datenverarbeitung auf drei Ebenen:

Ebene 1: REGENT bietet eine definierte Schnittstelle zur elementaren Treiber-Software der graphischen Ausgabegeräte. Ohne Änderung der Module kann der Subsystem-Anwender das für ihn geeignete graphische Ausgabegerät auswählen (PLOT=...).

Ebene 2: GIPSY. Zur Ausgabe von Zeichnungen bedient sich GIPSY der durch die Ebene 1 bereitgestellten Fähigkeiten.

Ebene 3: Andere REGENT-Subsysteme, die graphische Fähigkeiten beanspruchen, benutzen hierzu die Möglichkeiten des Subsystems GIPSY. Dies ist dadurch möglich, daß die Module eines Subsystems in der Sprache eines anderen Subsystems (hier: GIPSY) geschrieben werden können.

Der Anwender hat die Möglichkeit, auf alle Ebenen zuzugreifen, der Subsystemhersteller arbeitet entweder mit Ebene 1 oder 2, um für sein Subsystem eine neue Ebene 3 zu schaffen:



Schichten für die Anwendung graphischer Fähigkeiten

7.2.3 Für GIPSY erforderliche REGENT-Optionen

Da GIPSY die REGENT-Datenverwaltung für dynamische Datenfelder nicht benutzt, sollte zum Sparen von Speicherplatz die REGENT-Option NODA angegeben werden. Die PLOT-Option wurde bereits behandelt. Zur Speicher- oder Zeit-optimalen GIPSY-Anwendung ist außerdem die Angabe der Modul-Poolgröße durch MPOOL= von großer Wichtigkeit. Große Modulpoolgröße bedeutet großen Speicherplatzbedarf und geringe Rechenzeit, kleine Poolgröße geringere Speicherplatzanforderungen und größere Rechenzeit. Die folgende Tabelle gibt die minimal und maximal erforderliche Modulpoolsize für 2D-GIPSY-Anwendungen, 3D-Anwendungen und gewünschte Anwendungen an.

	Poolgröße	
	minimale	maximale
2D	50 K	364 K
3D	100 K	525 K
gemischt	100 K	616 K

Werden zur Rechenzeitoptimierung die Anweisungen LOAD 2D, LOAD 3D oder LOAD ALL benutzt, wird stets die maximale Poolgröße benötigt. Die oben angegebenen minimalen Poolgrößen

werden dann benötigt, wenn alle 2D oder 3D Fähigkeiten oder alle GIPSY-Fähigkeiten angesprochen werden. Im einzelnen Anwendungsfall können sie also noch darunter liegen. Die aktuell benutzten Poolgrößen werden ausgedruckt, wenn vor der Anweisung FINISH; die Anweisung MESSAGE ACTIVE DEBUG; angegeben wird (siehe auch REGENT-Handbuch). Die benötigte REGION-Angabe kann nach der folgenden Formel abgeschätzt werden:

$$R \approx M + ISA + P + 110K \quad ;$$

M : Maximal benutzte Modulpoolgröße
ISA: Größe des PL/1-Speicherbereiches, siehe dazu Abschnitt 7.5, ISA.
P : Größe des Anwenderprogrammes. Wird vom Loader auf SYSLOUT ausgegeben und muß von hexadezimaler in dezimale Darstellung umgerechnet werden,
R : erforderliche REGION in Bytes.

Der konstante Wert von 110K gilt für die KfK-Installation mit großen Blockgrößen aller Dateien und 5 Puffern pro Datei. Durch Reduktion der Blockgrößen und Verminderung der Pufferanzahl auf 2 kann dieser Wert erheblich gesenkt werden.

Die erste Karte eines GIPSY-Anwenderprogrammes kann also folgendermaßen aussehen:

```
P  :  PROC OPTIONS(MAIN)
       REGENT(NODA,MPOOL=200000,PLOT=STATOS);
```

bei Ausgabe auf Plotter, oder

```
P  :  PROC OPTIONS(MAIN)
       REGENT(NODA,MPOOL=200000);
```

bei Ausgabe auf den Plotfile.

7.3 Der Plotfile

Der Plotfile ist ein Mittel zum langfristigen Speichern und zum Transportieren graphischer Information. Er wird nicht nur in GIPSY benutzt, sondern ist eine standardisierte Schnittstelle zwischen verschiedenen Systemen und Geräten innerhalb der AGF / 10 /

(Arbeitsgemeinschaft der Großforschungszentren). Vom Plotfile ausgehend können Bilder auf Terminals, Plotter, Filme oder einen graphischen Arbeitsplatz ausgegeben werden.
Das Beschreiben des Plotfiles durch SAVE und PLOT in einem GIPSY-Programm wird in Abschnitt 6.5 beschrieben. Dieser Abschnitt enthält

- Dateiformat des Plotfiles
- Aufbau des Plotfiles
- Anleitung zur Darstellung des Plotfiles an einem Tektronix-Terminal
- Anleitung zur Erzeugung von Filmen
- Ausgabe des Plotfiles auf Plotter
- Editieren des Plotfiles mit dem graphischen Arbeitsplatz T4081

7.3.1 Dateiformat

Im Anwenderprogramm:

```
DCL file FILE;
ENTER GIPSY FILE(file);
```

Dateiformat:

Der Plotfile hat das Record-Format FB. Eine normale TSO-DATA-Datei mit DCB=(BLKSIZE = 3120, LRECL = 80, RECFM = FB) kann verwendet werden. Jedes andere FB-Format ist zulässig. Der Standard-Plotfile sieht vor: DCB = (BLKSIZE = 960, LRECL = 80, RECFM = FB).

Beispiel für das Beschreiben des Plotfiles:

```
T : PROC REGENT(NODA,PLOT=STATOS)
        OPTIONS(MAIN);"PLOT=" ist nicht erforderlich bei SAVE
                                                        alleine
DCL FILE1 FILE;
```

```
 ENTER GIPSY FILE(FILE1);
 .....
 .....
  DO WHILE(WEITER);
   GET LIST(PLOT_ODER_SAVE);
   IF PLOT_ODER_SAVE=1 THEN
      SAVE END;                          NUR PLOTTER-AUSGABE
   ELSE IF PLOT_ODER_SAVE=2 THEN
      SAVE;                              NUR PLOTFILE-AUSGABE

   ELSE                                  GLEICHZEITIG PLOTTER-
      SAVE PLOT;                         UND PLOTFILE-AUSGABE
   PLOT(.....);
  END;

 END PROG;
//G.FILE1 DD DSN=TSO147.PLOTLIB.DATA,DISP=SHR
                                        DD-KARTE FUER BESTEHENDE
                                        TSO-DATEI
//G.FILE1 DD DSN=PLOTLIB.DATA,UNIT=TAPEX,VOL=SER=BANDNAME,
//        DISP=(NEW,KEEP),LABEL=....,DCB=(LRECL=80,
//        BLKSIZE=3120,DEN=3,RECFM=FB)
                                        DD-KARTE FUER PLOTFILE
                                        AUF BAND.SOWOHL LABEL=(,NL)
                                        ALS AUCH LABEL=(N,SL) IST
                                        MOEGLICH.FUER DIE DCB-PARA-
                                        METER IST LEDIGLICH RECFM=FB
                                        FEST VORGESCHRIEBEN.
```

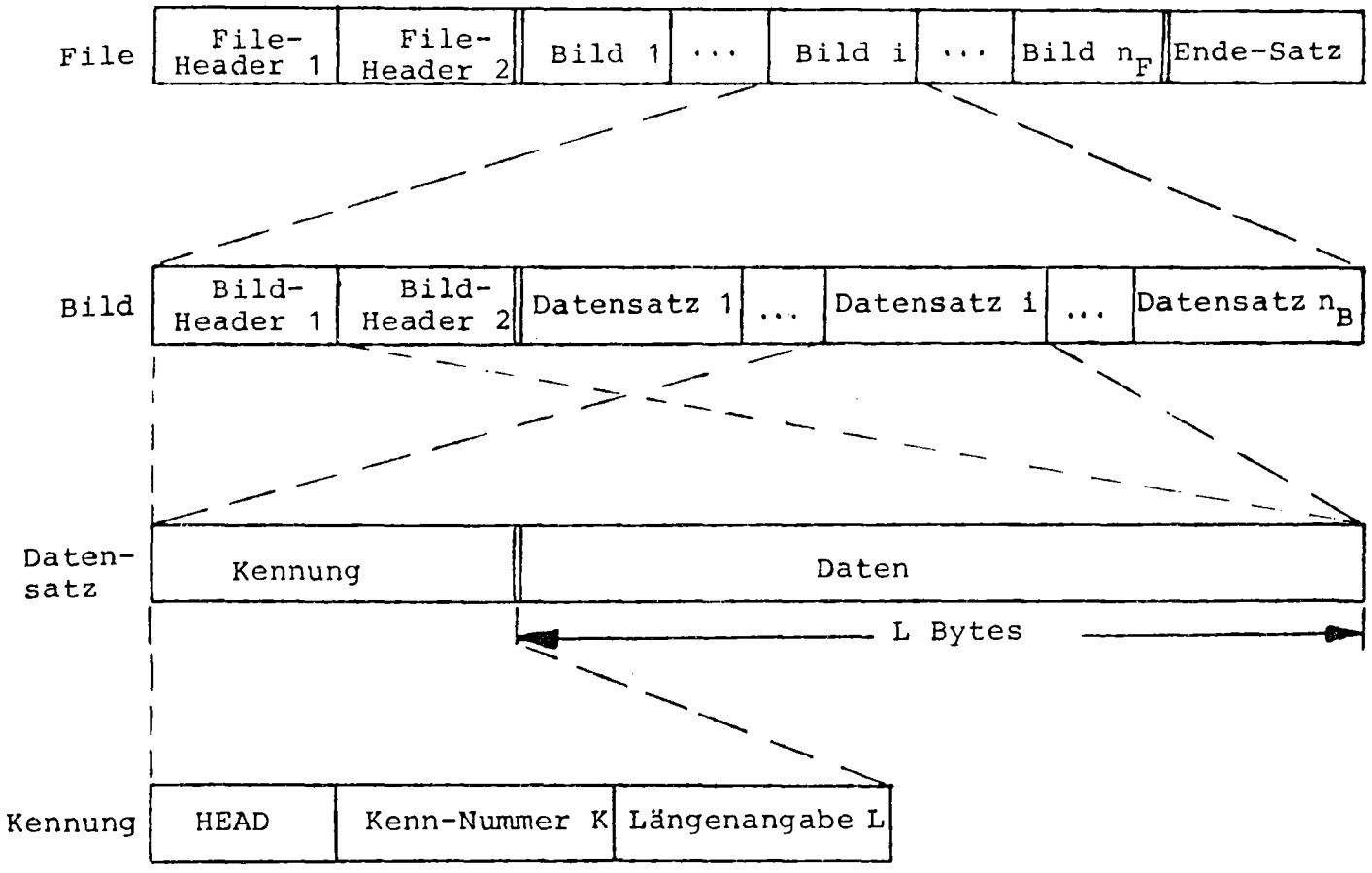
7.3.2 Plotfile-Aufbau

Die Daten (Koordinaten, Attribute) werden auf dem Plotfile wahlweise in einer formatierten Darstellung (das ist die Standard-Darstellung) oder in interner Codierung abgespeichert. Das ist für unsere Zwecke innerhalb KfK rationeller. Das Umschalten erfolgt durch:

SAVE START { FORMATTED / UNFORMATTED } ;

Bei FORMATTED werden Integerzahlen auf dem Plotfile im Format F (6) dargestellt. Realzahlen im Format F (10,5). Bei UNFORMATTED werden Integerzahlen als BIN FIXED (15), Realzahlen als BIN FLOAT (21) in 2 oder 4 Bytes im Maschinencode ausgegeben. Die unformatierten Daten beanspruchen so nur etwa 40% des Platzes auf der Datei.

Übersicht:



Die Besonderheiten bei der Beschreibung des Plotfiles mit GIPSY sind jeweils bei den einzelnen Sätzen vermerkt. Die Angaben (i) und (r) in Klammern bedeuten: Integerzahlen bzw. Realwerte. Tabelle 7.1 gibt eine Übersicht über die Datensätze des Plotfiles. In GIPSY werden nur die mit G bezeichneten Datensätze verwendet.

Datensatztyp	Typangabe	Kurzbeschreibung
File-Header	HEADOO11 G HEADOO12	Header 1: Standard-Angaben Header 2: Zusatztext
Bild-Header	HEADOO21 G HEADOO22	Header 1: Standard-Angaben Header 2: Zusatztext
Objekt-datensatz	HEADOO31 G HEADOO32 HEADOO33 G HEADOO34 G	Polygonzug Vektorschar Punkthaufen Text
Attribut-information	HEADOO41 G HEADOO42 G HEADOO43 HEADOO44 G HEADOO45 G HEADOO46 G	Stiftdefinition Stiftauswahl Farbe, Strichstärke Symboltyp Linientyp Schriftform
Rasterdatensatz	HEADOO51	Ein Rastergraphik-Bild
Nachrichten-datensatz	HEADOO61 HEADOO62 G	Operator-Nachricht Bild-Struktur
Unbenutzt	HEADOO71 bis HEADOO79 und HEADiikk ii≠OO	Reserviert für spätere Erweiterungen (z.B. Flächengraphik)
Benutzerinformation	HEADOO81 bis HEADOO89	Anwendungsspezifische Daten
Ende-Satz	HEAD 9999 G	Ende des Plotfiles

Tabelle 7.1: Datensätze des Plotfiles.

File Header 1:

HEADOO11	L=8O	N	A	T	F	V	D	Z	M	E	J	R	O

N(8∝): Name der Installation, 8 Zeichen

A(8∝): Name des Autors, 8 Zeichen

T(8∝): Datum(Jahr-Monat-Tag), 8 Zeichen (z.B.'77-10-27')

F(i4) : File-Nummer

V(i4) : Versions-Nr. Dieser Bericht
beschreibt Version 1

D(i4) : Koordinaten-Dimension (für alle Koordinaten und Bilder)
2 oder 3

Z(i4) : Zahltyp der Koordinaten und Längen (mit (r) bezeichnet)
1: Realzahlen (d.h. mit Dezimalpunkt)
2: Integerzahlen

M(i4) : Mantisse des Maßstabes

E(i4) : Exponent des Maßstabes
Maßstab M*10**E Meter

J(i4) : Formatangabe für Integerzahlen (mit (i) bezeichnet)

J > 0 : Formatierte Darstellung: IJ (FORTRAN) bzw. F(J) (PL/1)

J < 0 : Unformatierte Darstellung mit |J| Bytes

R(i4) : Formatangabe für Realzahlen (mit (r) bezeichnet)

R > 0 : Formatierte Darstellung:
Z=1: FR.O (FORTRAN) bzw. F(R,O) (PL/1)
Z=2: IR (FORTRAN) bzw. F(R) (PL/1)

R < 0 : Unformatierte Darstellung mit |R| Bytes, sowohl für Z=1 (Realzahl) als auch für Z=2 (Integerzahl)

O(24α) : reserviert für Erweiterungen

Die Umrechnung von Plotfile-Koordinatenwerten in logische Bildkoordinaten erfolgt nach der Regel

$$X_{Bild} = X_{Plotfile} \cdot M \cdot 10^{E}.$$

Für maßstäbliche Zeichnungen haben die logischen Bildkoordinaten und alle Längenangaben die Einheit Meter.

Im File-Header 1 wird die Länge L im Format I4 dargestellt, in allen folgenden Datensätzen im Format IJ , Die Typangabe (HEAD mit Kenn-Nummer) wird stets im Format 8α angegeben (z.B.HEAD0031).

Für GIPSY gilt:

N =	'KfK/IRE'	Z =	1
A =	'PLOTFILE'	M =	1
T =	Datum des Jobs	E =	0
F =	1	J =	6 für FORMATTED
			-2 für UNFORMATTED
V =	1	R =	10 für FORMATTET
D =	2		-4 für UNFORMATTED

File-Header 2:

HEADOO12	L	N	T

N(i) : Anzahl der Zeichen im Text T

T(Nα) : Text mit N Zeichen

Hier können Angaben, die den gesamten File betreffen, in Textform geschrieben werden, z.B. Angaben über das graphische System oder Unterprogrammpaket, das die Bilder erzeugt hat; Angaben über das ursprünglich verwendete Ausgabegerät; Angaben über das gewünschte Ausgabegerät. Der File-Header 2 entfällt bei GIPSY.

Bild-Header 1:

HEADOO21	L	B	U	R	P	M	I	F	S	D

B(i) : Bild-Nummer

U(2r/3r), R(2r/3r) : Window, x-, y-, bzw. x-, y-, z-Koordinaten der linken unteren (vorderen) Ecke des Bildbereiches und Koordinaten der rechten oberen (hinteren) Ecke des Bildbereiches.
Für Rastergraphik wird man i.a. U=1,1 und R=RX, RY setzen, wobei das Bild RX Zeilen und RY Punkte pro Zeile besitzt.

P(i) : Maximale Anzahl von Stiftnummern für dieses Bild. Für P=-1 wird nicht der Attribut-Satz"Stift-Definition" und "Stiftaufruf", sondern der Attributsatz "Farbe/Strichstärke" verwendet.

M(i) : Modusparameter:

M=1: Die Intensitäten und Farben werden über die drei Parameter: Intensität (I), Farbton (F) und Farb-Sättigung (S) definiert.

M=2: Die Intensitäten und Farben werden über die drei Parameter: Rotanteil (I), Grünanteil (F) und Blauanteil (S) definiert (Farbdreieck mit additiver Farbmischung)

I(i) : Maximale Intensität (Helligkeit)
F(i) : Maximale Farbton-Nummer
S(i) : Maximale Farbsättigungs-Nummer
} für M=1

Für M=2 sind I, F und S die Zahlen, die 100% rot, grün und blau entsprechen.

D(r) : Maßstab für Strichstärken in Längeneinheiten. Für D=0 wird kein Maßstab angegeben.

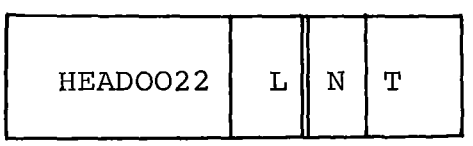
Für GIPSY gilt:

U und R sind (0.0, 0.0) und die OPEN-PLOT-Werte. M=2,P=0.

S, und I sind 0, F wird durch CHANGE MAXCOLOR festgelegt.

D = 0.00001, d.h. Strichstärken werden in Hunderstel mm angegeben.

Bild-Header 2:



N(i) : Anzahl der Zeichen im Text T

T(Nα) : Text mit N Zeichen

Hier können Angaben, die das gesamte Bild betreffen, geschrieben werden.

Der Bild-Header 2 entfällt bei GIPSY.

Graphische Grundelemente (Primitives)

Der Plotfile enthält die graphischen Objekte: Polygonzug, Vektorschar, Punkthaufen und Text. Höhere graphische Elemente wie Kreis und Kreisbogen oder Koordinatenachse wurden aus Gründen der Einfachheit nicht aufgenommen. Die Aufnahme von Vektorschar (zusätzlich zu Polygonzug) und Punkthaufen (statt Einzelpunkt) ist zur Darstellung von Strichbildern nicht notwendig. Diese Objekte vereinfachen aber für eine Reihe von Anwendern, die mit Punkthaufen (z.B. zwei- und dreidimensionale Verteilungsfunktionen) und mit Vektoren arbeiten, den Umgang mit dem Plotfile und führen zu einer Reduktion des zur Speicherung nötigen Platzes. Der Mehraufwand für einen Interpretierer, neben Punkten auch Punkthaufen und neben Polygonzügen auch Vektorscharen lesen zu können, ist dagegen vernachlässigbar.

In den Objektdatensätzen sind lediglich die geometrischen Angaben der Grundelemente enthalten, ihre Attribute wie Strichstärke, Farbe oder Texthöhe werden durch die Attributdatensätze beschrieben.

Polygonzug:

HEADOO31	L	N	P_1	...	P_N

N(i) : Anzahl der Punkte (Elemente) des Polygonzuges

P_j(2r/3r) : Koordinaten des J-ten Punktes

$1 \leq J \leq N$

Vektorschar:

HEADOO32	L	N	P_1	Q_1	...	P_N	Q_N

N(i) : Anzahl der Vektoren (Elemente) der Vektorschar

P_J(2r/3r) : Koordinaten des Anfangspunktes des J-ten Vektors

Q_J(2r/3r) : Koordinaten des Endpunktes des J-ten Vektors

$1 \leq J \leq N$

Prinzipiell kann dieser Datensatz durch N Polygonzug-Datensätze ersetzt werden; dies kann aber ca. 50% mehr Speicherplatz erforderlich machen. Er entfällt in GIPSY.

Punkthaufen:

HEADOO33	L	N	P_1	...	P_N

N(i) : Anzahl der Punkte (Elemente) des Punkthaufens

P_J(2r/3r) : Koordinaten des J-ten Punktes

$1 \leq J \leq N$ GIPSY: Für Punkte N=1,
für Polygonzüge mit LINETYPE=DOTTED,N=NMAX.

Text:

HEADOO34	L	P	A	N	T

P(2r/3r) : Koordinaten des Anfangspunktes des Textes (linke untere Ecke des virtuellen Rechteckes um das erste Zeichen).

A(i) : Art der Anfangspunkt-Koordinaten
1 : absolute Angabe
2 : relative Angabe, d.h. bezüglich des augenblicklichen Standes der Feder bzw. des Strahls. Dabei ist der Feder- bzw.Strahlstand nach der Zeichnung eines Textes in der linken unteren Ecke des virtuellen Rechteckes des (theoretischen) (N+1)-ten Zeichens

N(i) : Anzahl der Zeichen im Text T

T(Nα) : Zu zeichnender Text mit N Zeichen
(alle EBCDIC-Zeichen einschließlich Kleinbuchstaben)

Für A=2 wird der Text relativ zur augenblicklichen Position plaziert. Dies ist die einzige relative Angabe im Plotfile. Sie ist z.B. erforderlich,wenn man Texte geringer Textqualität hintereinander zeichnen möchte, deren Höhe und Breite/Höheverhältnis man nicht genau kennt. A=2 ist nur gültig, wenn das vorausgegangene Grundelement ebenfalls ein Text war. In GIPSY ist stets A=1.

Attributinformation

Die Attributinformation beschreibt die Darstellungsattribute für alle nachfolgenden Objekt-Datensätze. Am Anfang jeden Bildes sind alle Darstellungsattribute auf ihre Standardwerte gesetzt. Es gibt die Datensätze: Stiftdefinition, Stiftauswahl, Farbe/Strichstärke, Symboltyp, Linientyp und Schriftform. Sie wirken sich nach Tabelle 7.2 auf die Grundelemente aus.

Attribut	Wirkung auf			
	Polygon-zug	Vektor-schar	Punkt-haufen	Text
Stiftdefinition + Stiftauswahl	x	x	x	x
Farbe/Strichstärke	x	x	x	x
Symboltyp	x	x	x	
Linientyp	x	x		
Schriftform				x

Tabelle 7.2: Auswirkung der Darstellungsattribute auf Objekte.

Die Bedeutung der einzelnen Werte für ein Attribut (z.B.2=rot oder 4=gestrichelte Linie) sind festgelegt, diese Zuordnung kann aber jederzeit durch Eingabe an den Interpretierer geändert werden. Zur Beschreibung der Attribute Farbe und Strichstärke sind zwei Möglichkeiten vorgesehen. Bei der ersten Methode wird eine Stift-Nummer mit bestimmten Werten für Farbe, Helligkeit und Strichstärke assoziiert. Dies geschieht durch den Attributsatz "Stift-Definition".

Anschließend kann einer der vorher definierten Stifte durch den Satz "Stift-Auswahl" ausgewählt werden. Diese Methode ist von Vorteil für Geräte mit einer begrenzten Anzahl von Stiften. Mit der zweiten Möglichkeit können Farbe, Helligkeit und Strichstärke ohne Benutzung einer Stift-Nummer direkt angegeben werden. Dies geschieht durch den Attributsatz Farbe/Strichstärke. Diese Methode ist für Farbdisplays angemessen. Welche der beiden Beschreibungsmöglichkeiten gewählt wird, ist durch den Parameter M in Bild-Header 1 festgelegt.

Stiftdefinition

HEADOO41	L	T	I	F	S	B

T(i) : Stiftnummer, T ≤ P (P im Bildheader 1)

I(i) : } Helligkeit, Farbton, Farbsättigung für

F(i) : } M=1 (M im Bildheader 1), Anteil

S(i) : } von rot, grün und blau für M=2.

B(i) : Strichstärke, Vielfaches der Strichstärkeneinheit D (im Bildheader 1).

Dieser Datensatz kann eine vorhergehende Stiftdefinition mit gleicher Stiftnummer M überschreiben.

Stiftauswahl:

HEADOO42	L	M

M(i) : Stiftnummer.

Ist der Stiftauswahl keine Stiftdefinition vorangegangen, so wird eine Standardtabelle benutzt.

Wird kein Stift ausgewählt (mit Datensatz 42 oder 43), so wird mit normal üblicher Farbe (z.B.schwarz) und Intensität gezeichnet (geräteabhängige Werte).

Stiftdefinition/Stiftauswahl werden in GIPSY verwendet, wenn die Stift-Nr. oder andere Attribute sich ändern.

Farbe/Strichstärke

HEADOO43	L	I	F	S	D

I(i): Intensitätsnummer (bei M=1, Bildheader 1)
bzw. Rot-Anteil (bei M=2, Bildheader 1)
F(i): Farbtonnummer (bei M=1, Bildheader 1)
bzw. Grün-Anteil (bei M=2, Bildheader 1)
S(i): Farbsättigungsnummer (bei M=1, Bildheader 1)
bzw. Blau-Anteil (bei M=2, Bildheader 1)
D(i): Vielfaches der Strichstärkeneinheit (vergl. Bildheader 1) bzw. allgemeine Strichstärkennummer

Dieser Datensatz entspricht dem Datensatzpaar 41/42, es wird jedoch nicht mit Stiften, die vorher definiert werden müssen, gearbeitet, sondern die entsprechende Angabe wird direkt gemacht. Im Bildheader 1 wird durch den Parameter P angegeben, nach welcher Methode Farbe und Strichstärke angegeben werden. In GIPSY nicht verwendet.

Symboltyp:

HEADOO44	L	S	M	G

S(i): Symbolnummer. Symbole sind zentrierte Zeichen (Marker) zur Kennzeichnung von Punkten und zur Unterscheidung von Linien.
Zuordnung: 1: kein Symbol(d.h.kleinster Punkt bei Punkthaufen
2:
3:
4:
5: +
6: ×
7:
8:
9:
10:
≥ 11: geräte-oder systemabhängig

M(i): Angabe, jedes wievielte Element der jeweiligen Objektdatensätze markiert werden soll

Polygonzug: 1 Element = 1 Eckpunkt (Anfangspunkt einer Teillinie)

Vektorschar: 1 Element = 1 Vektor mit beiden Endpunkten

Punkthaufen: 1 Element = 1 Punkt

G(r): Größe des Symbols

Angabe in Längeneinheiten (siehe Maßstabsdefinition im File-Header 1)

Bei Übergabe von 0.0 wird ein Geräte- oder System-Standardwert benutzt.

Festlegung:

Standardwert des Symbols,wenn keine explizite Symbolangabe erfolgt: Symbol 1 (Punkthaufen: kleinster Punkt, Polygonzug und Vektorschar: kein Symbol)

Die Symbolnummer 1 (Standardwert) dient zum Zurücksetzen eines gewählten Symbols, wenn danach kein anderes Symbol (mit S>1) gewählt wird, d.h. kein Symbol gezeichnet werden soll.

Die Form und Lage der Symbole ist fest, sie wird nicht durch die Schriftform-Attribute beeinflußt. GIPSY erzeugt diesen Satz,wenn sich Symbolnummer, Symbolhöhe oder EVERY von einem zum nächsten Objekt ändern.

Linientyp:

HEAD0045	L	T	N	S_1	G_1	...	S_N	G_N

T(i): Linientyp-Nummer

0: eigene Linientyp-Definition in S_1/G_1 bis S_N/G_N

1: ——— (durchgezogen)

2: ..... (gepunktet)

3: —·—·— (strichpunktiert)

4: – – – – (gestrichelt)

≥ 5: geräte- oder systemabhängig

N(i): 0 oder Anzahl der Teilstriche für eigene Linientyp-Definition (Periodenlänge)

$S_J(r)$: Länge des J-ten Teilstriches für eigene Linientyp-Definition

1 J Angabe in Längeneinheiten (siehe Maßstabsdefinition im
J N File-Header 1)
Bei Übergabe von 0.0 wird ein Punkt gezeichnet

$G_J(r)$: Länge des J-ten Gaps für eigene Linientyp-Definition

1 J Angabe in Längeneinheiten (siehe Maßstabsdefinition im
J N File-Header 1)

Festlegung:
Standardwert des Linientyps, wenn keine explizite Linientypangabe erfolgt: Linientyp 1 (durchgezogen).

Bei N=0 entfallen die S_J- und G_J-Parameter. Dies ist nur bei T> 0 möglich.

GIPSY erzeugt für

THROUGH: T = 1, N = 0
DASHED : T = 0, N = 1, S(1) = G(1) = LENGTH/2.
MARKED : T = 1, N = 0
CENTER : T = 3, N = 2, S(1) = LENGTH/2, S(2) = 0,
G(1) = G(2) = LENGTH/4.

Für DOTTED wird dieser Satz nicht erzeugt.

Schriftform:

HEAD0046	L	F	Q	H	B	S	P

F(i): Font (Zeichensatz)
1: Standard
2: Romanisch
3: Italic
4: Griechisch
5: Mathematische Symbole

Q(i): Qualität
1: Einfache Qualität
2: Mittlere Qualität
3: Hohe Qualität
Kleine Qualität bedeutet: Nur waagerechter Text einer Höhe, bei Text hoher Qualität kann Neigung, Winkel, Höhe und Breite der Zeichen vorgegeben werden.

H(r) : Höhe eines Zeichens (virtuelles Rechteck um einen normalen Großbuchstaben, z.B. das M)
Angabe in Längeneinheiten.

B(r) : Breite eines Zeichens (virtuelles Rechteck um einen normalen Großbuchstaben, z.B. das M) in Längeneinheiten-
Siehe Abb.7.2

S(2r/3r): Abstandspunkt-Angabe
Anfangspunkt des nächsten Zeichens relativ zum vorhergehenden Zeichen, i.a. Abstand der jeweiligen linken unteren Eckpunkte (x-, y-, bzw. x-, y-, z- Koordinaten). S definiert also die Schreibrichtung und den Endpunkt, d.h. den Federstand /Strahlstand nach der Beendigung des Textaufrufes; die Differenz zur Breite gibt die Zwischenraum-Breite an.

P(2r/3r): Ebenenpunkt-Angabe
Definition der Ebene, in der der Text liegen soll, durch Angabe eines Punktes oberhalb der Schriftachse (Schreibrichtung).
Diese Angabe ist erforderlich für Texte im Raum.
Angabe in x-, y-, bzw. x-, y-, z- Koordinaten relativ zum Anfangspunkt des Textes (linker unterer Eckpunkt des ersten Zeichens).

Die Länge von P, die prinzipiell als Höhe des Zeichens (bzw. der Projektion auf die Normale) angesehen werden könnte, wird nicht interpretiert (aus Gründen der Klarheit und Übersichtlichkeit). Der Winkel zwischen P, dem Anfangspunkt des Textes und S kann als Neigung interpretiert werden, jedoch erst bei hoher Textqualität.

Für GIPSY gilt: F = 1, Q = 3. Intern werden in GIPSY die Punkte P und S gespeichert. H wird als Abstand OP ausgerechnet, B als 0.8 mal Abstand OS.

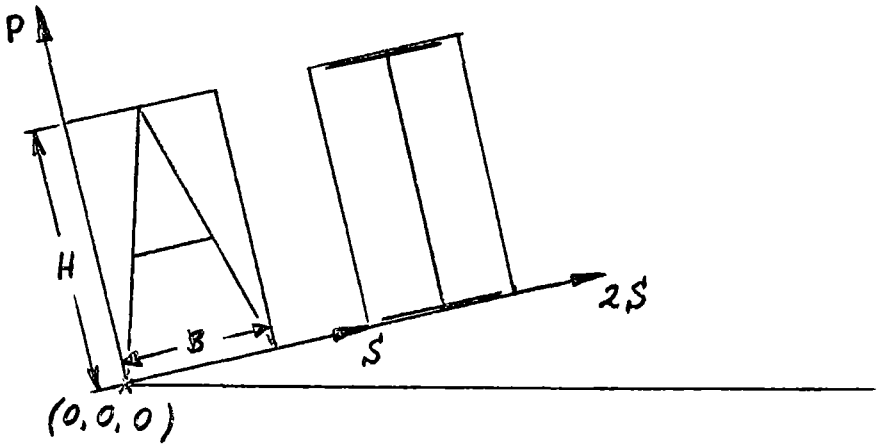


Abb.7.2: Angaben zur Schriftform

Rasterdatensatz

HEADO051	L	C	I_1^1	...	I_1^C	...	I_N^1	...	I_N^C

C(i) : Anzahl Kanäle

$I_J^M(r)$: Intensitätsangabe, Graustufen oder Farbanteil des M-ten Kanals des J-ten Pixels (Bildpunktes). Die Anzahl der Bildpunkte N=RX·RY ergibt sich aus dem R-Parameter von Bildheader 1. Für C=3 werden die Werte I normalerweise die Anteile der Grundfarben rot, grün und blau sein. Die Werte, die 100% einer Farbe entsprechen, sind in Bildheader 1 definiert. Für C=1 werden die Werte als Graustufen interpretiert.

$1 \leq M \leq C$

$1 \leq J \leq N$

Entfällt in GIPSY.

2.6 Nachrichtendatensätze, Benutzerinformation und Ende-Satz

Operator-Nachrichten können zur Ausgabe einer Nachricht an den Bediener eines graphischen Ausgabegerätes verwendet werden. Die Abarbeitung des Plotfiles kann dabei unterbrochen werden, um z.B. einen Stift zu wechseln. Die Bildstruktur kann durch Angabe von Teilbildnamen im Plotfile gespeichert werden. Sowohl eine einstufige als auch eine mehrstufige Bildstruktur ist damit beschreib-

bar. Die Benutzerdatensätze dienen dazu, für einen bestimmten Anwender wichtige Zusatzinformationen im Plotfile unterzubringen. Das können auch nichtgraphische Daten sein, die zu den graphischen Objekten gehören. Diese können i.a. nur durch bestimmte Interpretierer gelesen werden. Fremde Interpretierer überlesen die ihnen nicht bekannte Benutzerinformation. Durch ein Schlüsselwort wird sichergestellt, daß ein Interpretierer nur ihm bekannte Benutzerdaten verarbeitet.

Operator-Nachricht:

HEADOO61	L	S	N	T

S(i) : Stop-Parameter
- 1 : Ausgabe der Nachricht ohne darauffolgenden Stop
- 2 : Ausgabe der Nachricht mit darauffolgendem Stop; d.h. erst bei entsprechender Reaktion soll weitergemacht werden (z.B. Federwechsel)

N(i) : Anzahl der Zeichen im Text T

T(N∝): Text der Message mit N Zeichen

Entfällt in GIPSY

Segmentname:

HEADOO62	L	S	N	T

S(i) : Stufennummer einer Baumstruktur, die den Bildaufbau beschreibt

N(i) : Steuerparameter
- N=O : Festlegung des Endes des letzten Segmentes der Stufe S (d.h. kein rechter Bruder mehr)
- N≥1 : Festlegung des Beginns eines Segmentes der Stufe S (Sohn oder Bruder); dabei ist N die Anzahl der Zeichen im Segmentnamen T

T(N∝): Segmentname T mit N Zeichen (entfällt bei N=O)

In GIPSY ist S = 1, N = LENGTH (name), T = name = Name des GIPSY-Objektes. Die Namen aller im PLOT-Statement stehenden namentlich identifizierten Objekte werden mit auf den Plotfile geschrieben.

Ende-Satz

HEAD9999	L = 0

Ende des Plotfiles

7.3.2 Darstellung des Plotfiles an einem T4014-Terminal

Der Aufruf des Plotfile-Interpretierers am Terminal erfolgt durch

TEK plotfilename

plotfilename: Dataset-Name der Datei, die den Plotfile enthält. Da sich die Kommandoprozedur in der IRE-CMDLIB befindet, ist vorher einmal das Kommando IRECMD erforderlich. Das TEK-Kommando ruft folgende CLIST-Prozedur auf:

```
PROC 1 LIB

ATTN OFF

CONTROL MAIN NOFLUSH NOMSG

ALLOC F(T4010) DA(*)

ATTR A1 BLKSIZE(3120) LRECL(80) RECFM(F B)

ALLOC F(BILDFIL) NEW SPACE(50 20) TRACK USING(A1)

CONTROL MSG

ALLOC F(INFILE) DA(&LIB) SHR

PROF NOINTERCOM

CALL 'TSO147.POST.LOAD(PFTEKT)'

PROF INTERCOM

LISTB

FREE F(INFILE)

END
```

Der Filename INFILE ist dem Plotfile zugeordnet. BILDFIL ist eine temporäre Datei, auf die das jeweils gerade betrachtete Bild kopiert wird. BILDFIL wird nur beim ersten Aufruf von TEK allokiert und bleibt so bis zum Ende der Sitzung. Während der Bearbeitung des Plotfiles wird die Annahme von Messages unterbunden, damit sie nicht die Graphik stören. Aufgelaufene Messages werden vor Rückkehr in den READY-Modus ausgegeben.

Nach dem Aufruf zeichnet PFTEKT einen Rahmen für die Bildinformation und einen Menü-Bereich.

Kommandoeingabe:

Das System unterscheidet zwischen einer alphanumerischen und graphischen Ein-/Ausgabe. Dabei läuft die alphanumerische im wesentlichen im Menübereich ab, während die graphische Ein-/Ausgabe nur im Zeichenbereich stattfindet.

Durch Erscheinen eines Doppelpunktes im Menübereich wird angezeigt, daß das System zur Eingabe bereit ist. Nun kann ein Kommando eingegeben werden, wobei folgendes zu beachten ist:

- jede Eingabe darf nur ein Befehlswort enthalten; dieses muß am Anfang der Kommandoliste stehen,
- jede Eingabe kann in Groß- und/oder Kleinbuchstaben erfolgen,
- jedes Kommando wird durch Drücken der Returntaste abgeschlossen,
- das Kommando wird bei Überschreiten des rechten Randes eines Menükästchens ohne Fehlermeldung abgeschnitten.

Um dem Benutzer bei Erkennen eines fehlerhaften bzw. falschen Kommandos die Möglichkeit zum Abbruch des Kommandos zu geben, wirkt die Eingabe eines "!" als simuliertes Attention. Aus diesem Grund ist das Ausrufezeichen aus dem gesamten Zeichenvorrat ausgeschlossen. Wirksam wird dieses simulierte Attention auch, falls das System einen fehlenden bzw. falschen Parameter in der Kommandoliste erkennt und sich nach entsprechender Fehlermeldung zum erneuten Einlesen des Parameters mit "REENTER:" gemeldet hat:

- durch Drücken der Returntaste,
- Eingabe eines "!"
- automatisch nach dreimaliger falscher Eingabe des entsprechenden Parameters.

Die Eingabe von Koordinatenwerten an das System erfolgt über die Position des Fadenkreuzes. Hierbei wird bei Erscheinen des Fadenkreuzes und dessen Positionierung durch Eingabe eines beliebigen Zeichens und Betätigen der Returntaste die Schnittpunktskoordinaten des Fadenkreuzes an das System zur Speicherung übergeben.

Das erste Kommando wird durch "ENTER COMMAND:" angefordert.

Kommandos

PICTURE, Bildauswahl

PICTURE { NEXT / NO/NR/NUMBER n / SKIP i }

P N : Das nächste Bild auf dem Plotfile wird gezeichnet, beim ersten Mal das erste Bild des Plotfiles.

P NO n : Bild Nr. n auf dem Plotfile wird gezeichnet.

P S i : i Bilder auf dem PLOTFILE werden überlesen, das dann folgende Bild wird gezeichnet.

Das gezeichnete Bild wird zum aktuellen Bild. Beim Zeichnen gelten die aktuellen Skalierungswerte.

DISPLAY, Neuzeichnen des Bildes

DISPLAY

Das aktuelle Bild wird mit den aktuellen Skalierungsparametern neu gezeichnet.

ALL, Zeichnen des aktuellen Bildes mit maximaler Ausdehnung

ALL

Das aktuelle Bild wird in maximal möglicher Größe so auf den Bildschirm gebracht, daß alle Zeichnungsinformationen sichtbar werden. Auch außerhalb des Bildfensters liegende Objekte werden gezeichnet. Damit kann bei GIPSY-Zeichnungen die Lage von graphischen Objekten und Objektteilen erkannt werden, die außerhalb des durch OPEN PLOT und VIEW definierten Fensters liegen: Bei der Ausgabe auf ein Zeichengerät wird diese Information dagegen abgeschnitten. Die aktuellen Skalierungsparameter werden durch ALL nicht verändert und kommen bei einem nachfolgenden PICTURE- oder DISPLAY-Befehl wieder zur Wirkung.

OVERLAY, kein Löschen des Bildschirmes

OVERLAY

Dieses Kommando unterdrückt das Löschen des Bildschirmes vor dem ersten nachfolgenden Zeichenbefehl (PICTURE, DISPLAY oder ALL).

INVISIBLE, Ändern der Darstellung unsichtbarer Körperkanten

INVISIBLE { OMITTED | THROUGH | DASHED }

Die Darstellung unsichtbarer Körperkanten von GIPSY-Zeichnungen kann geändert werden. Durch das INVISIBLE-Kommando können gestrichelte Linien entweder weggelassen werden (OMITTED), durchgezogen (THROUGH) oder gestrichelt dargestellt werden (DASHED). Die Körper müssen mit INVISIBLE LINETYPE DASHED erzeugt sein.

MENUE, Ändern der Menüfeldhöhe

MENUE, n

n: Anzahl der Zeilen für das Menüfeld. Der Standardwert ist 5. Ein Wert von n zwischen 2 und 21 ist zulässig.

Skalierung

Zur Abbildung der Bildinformation auf den Bildschirm sind zwei Arten von Abbildungstransformationen möglich, die WINDOW-VIEWPORT-Transformation und die WINDOW-SCALE-Transformation.

Ein WINDOW (Anwenderfenster) ist ein rechteckiger Ausschnitt aus dem Bild in Plotfile-Koordinaten, d.h. in Meter. Das WINDOW ist gegeben durch den linken unteren Eckpunkt und den rechten oberen Eckpunkt des Bildfensters. Zum Setzen des Window dient das WINDOW-Kommando.

Ein VIEWPORT (Gerätefenster) ist ein rechteckiger Ausschnitt aus der realen Zeichenfläche, also im vorliegenden Fall ein Bereich auf dem Bildschirm. Der linke untere Endpunkt und der rechte obere Endpunkt werden angegeben. Der VIEWPORT wird durch das VIEWPORT-Kommando definiert.

Bei der WINDOW-VIEWPORT-Transformation wird das Bild in x- und y-Richtung so verschoben und vergrößert bzw. verkleinert, daß das WINDOW auf den VIEWPORT genau abgebildet wird. Der linke untere Eckpunkt des WINDOW wird also auf den linken unteren Eckpunkt des VIEWPORTS abgebildet, entsprechend für den rechten oberen Eckpunkt (siehe Abb.7.3).

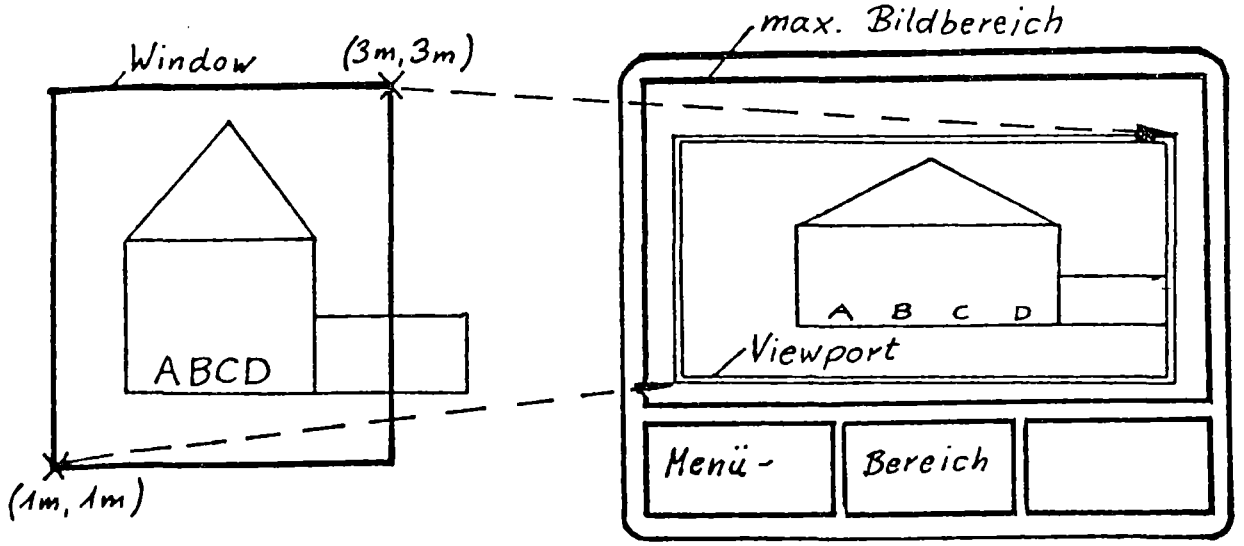


Abb.7.3: WINDOW-VIEWPORT-Transformation

Die WINDOW-SCALE-Transformation gestattet die maßstäbliche Abbildung eines Bildes. Der Maßstab kann dabei angegeben werden. Außerdem ist die linke untere Ecke des Gerätefensters anzugeben. Die Abbildung erfolgt so, daß der linke untere Eckpunkt des WINDOW auf den angegebenen linken unteren Eckpunkt des VIEWPORT fällt. Alle anderen Punkte werden so abgebildet, daß für alle Koordinatenabstände der angegebene Maßstab eingehalten wird. Die Skalierung kann mit dem SCALE-Befehl festgelegt werden (siehe Abb.7.4).

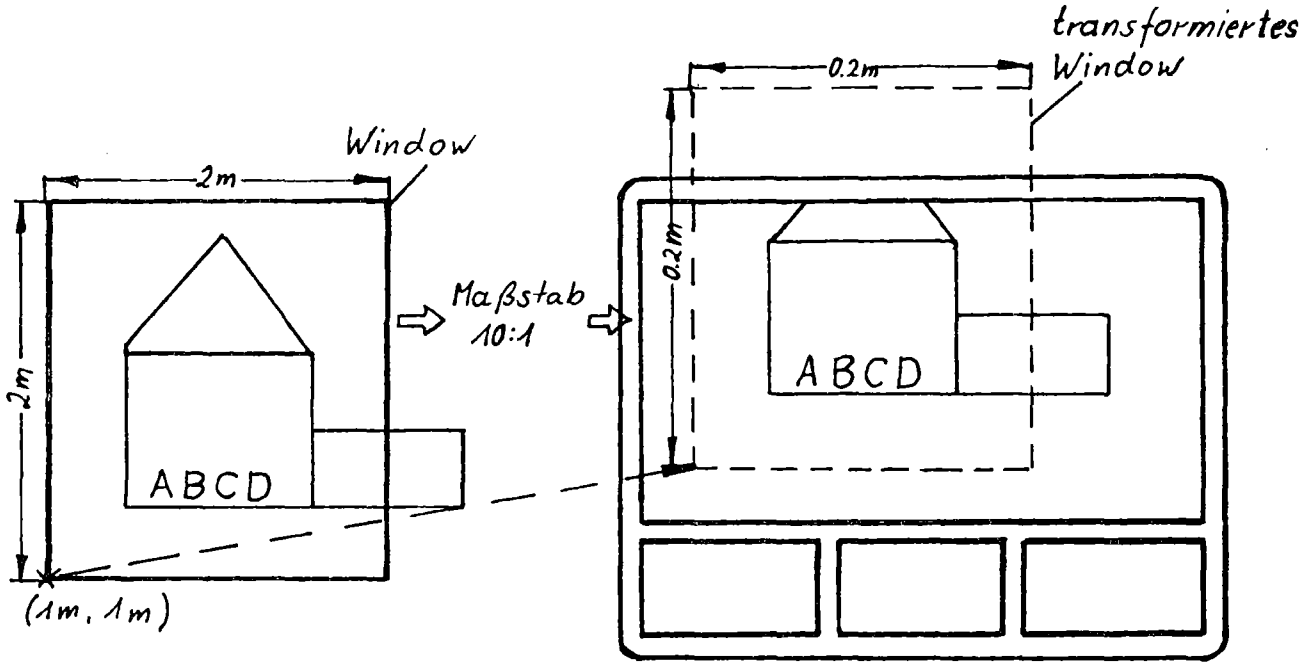


Abb.7.4: WINDOW-SCALE-Transformation

WINDOW, setze Anwenderfenster

$$\text{WINDOW} \left\{ \begin{array}{l} \underline{\text{MAX}} \\ \underline{\text{PLOTFILE}} \\ \underline{\text{SIZE}}\ x_1\ y_1\ x_2\ y_2 \end{array} \right\}$$

W M: Das WINDOW wird so gewählt, daß alle Koordinaten des Bildes darin enthalten sind. Dabei ist der GIPSY-OPEN-PLOT-Rahmen stets enthalten.

W P : Das Window wird auf den im Plotfile angegebenen Wert gesetzt. Bei GIPSY ist das der OPEN-PLOT-Wert.

W S : Das Window wird auf die angegebenen Werte gesetzt. (x_1,y_1) = linke untere Ecke, (x_2,y_2) = rechte obere Ecke. Alle Werte in Meter.

Beispiel:

W S -0.05 -0.05 0.4 0.3

definiert ein Window der Größe 45 x 35 cm mit dem linken unteren Eckpunkt bei (-5 cm, - 5 cm).

VIEWPORT, setze Gerätefenster

VIEWPORT { MAX / SIZE + + }

V M : Der maximal mögliche Zeichenflächenbereich wird als Viewport benutzt.

V S : Es erscheint zweimal das Fadenkreuz. Damit wird nacheinander die linke untere und die rechte obere Ecke des Viewports angefahren. Nach dem Positionieren des Fadenkreuzes muß mindestens ein Zeichen eingegeben und danach die RETURN-Taste gedrückt werden.

Bei dem Kommando ALL wird eine Abbildung erzeugt, die den Fenstern WINDOW MAX und VIEWPORT MAX entspricht.

SCALE, setze Maßstab

SCALE sx sy +

sx und sy sind die Maßstabsfaktoren in x- und y-Richtung. Danach wird durch "POSITION ORIGIN" und Erscheinen des Fadenkreuzes der Benutzer aufgefordert, den linken unteren Eckpunkt des Viewport zu setzen.

Beispiel:

```
S  0.5   0.5
```

Eine Abbildung, die der auf dem Zeichengerät genau entspricht, erhält man durch WINDOW PLOTFILE und SCALE 1 1.

Der Viewport wird nach SCALE automatisch errechnet. Er entspricht dem nach SCALE transformierten maximalen Window. Bildteile, die über den maximalen Bildbereich hinausgehen, werden dabei abgeschnitten.

Durch Kombination der Befehle OVERLAY und VIEWPORT oder OVERLAY und SCALE lassen sich mehrere Bilder nebeneinander auf dem Bildschirm darstellen.

END, Ende der Plotfile-Bearbeitung

```
END
```

Es wird in den TSO-READY-Modus zurückgekehrt.

Bei der Abbildung auf dem Bildschirm gilt:

- Alles, was über den VIEWPORT hinausgeht, wird abgeschnitten (das entspricht im Anwenderbereich einem Abscheiden am WINDOW).
- Der VIEWPORT wird durch eine doppelte Linie dargestellt.
- Das Plotfile-Fenster (OPEN-PLOT-Rahmen in GIPSY) wird durch eine einfache Linie dargestellt.

Abb.7.5 verdeutlicht die Zusammenhänge.

Beispielsitzung:

Die folgenden Abbildungen 7.6 bis 7.9 zeigen eine Beispielsitzung mit Kommandoeingabe, graphischer Ausgabe und Erläuterungen.

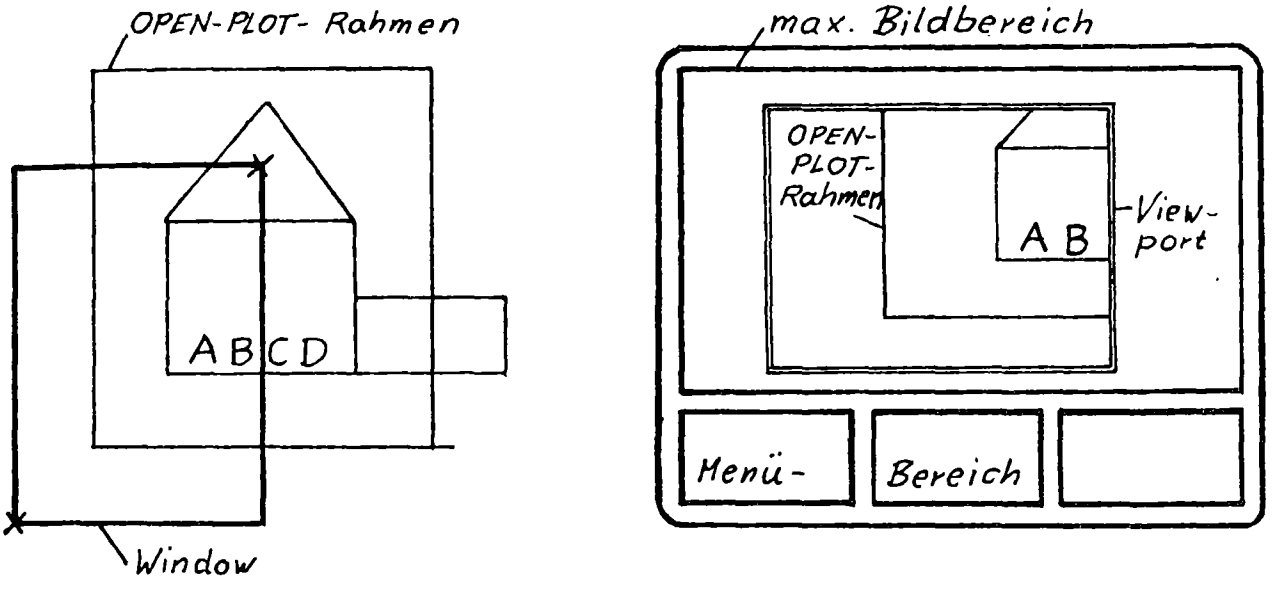


Abb.7.5: Darstellung von VIEWPORT und OPEN-PLOT-Rahmen.

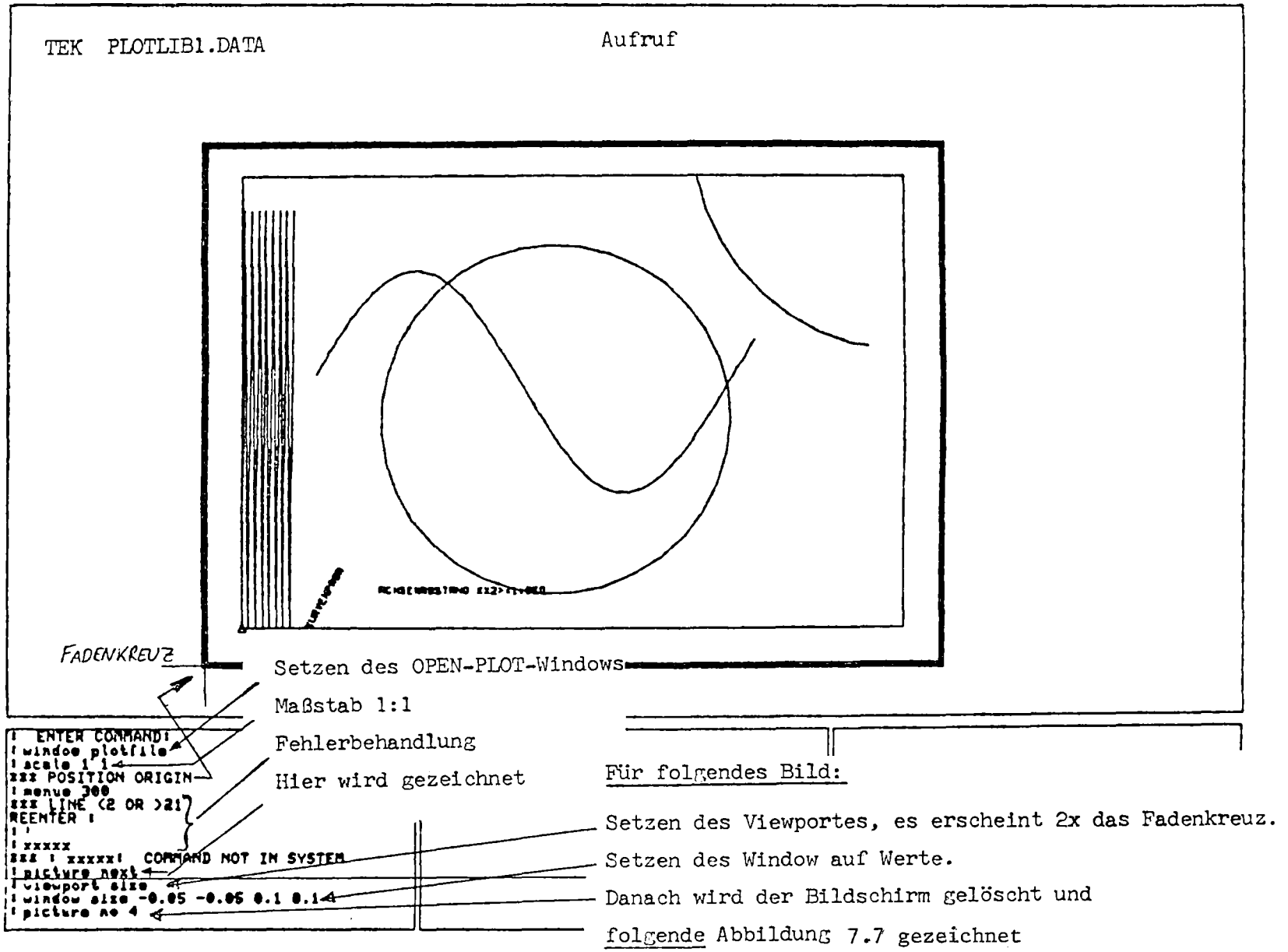


Abb. 7.6: Eröffnen der Sitzung, erstes Bild

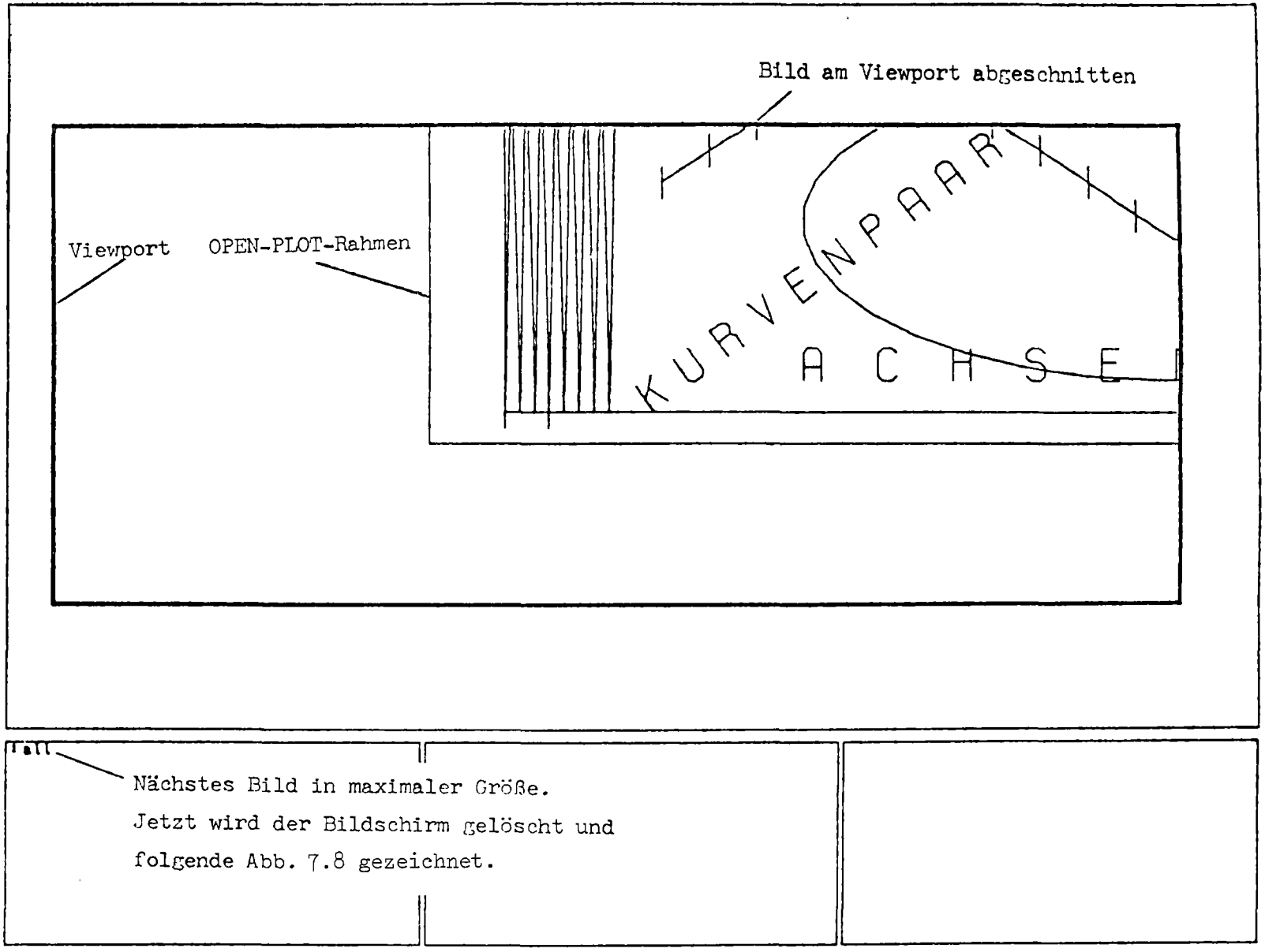


Abb.7.7: Anderes Bild mit veränderter Transformation

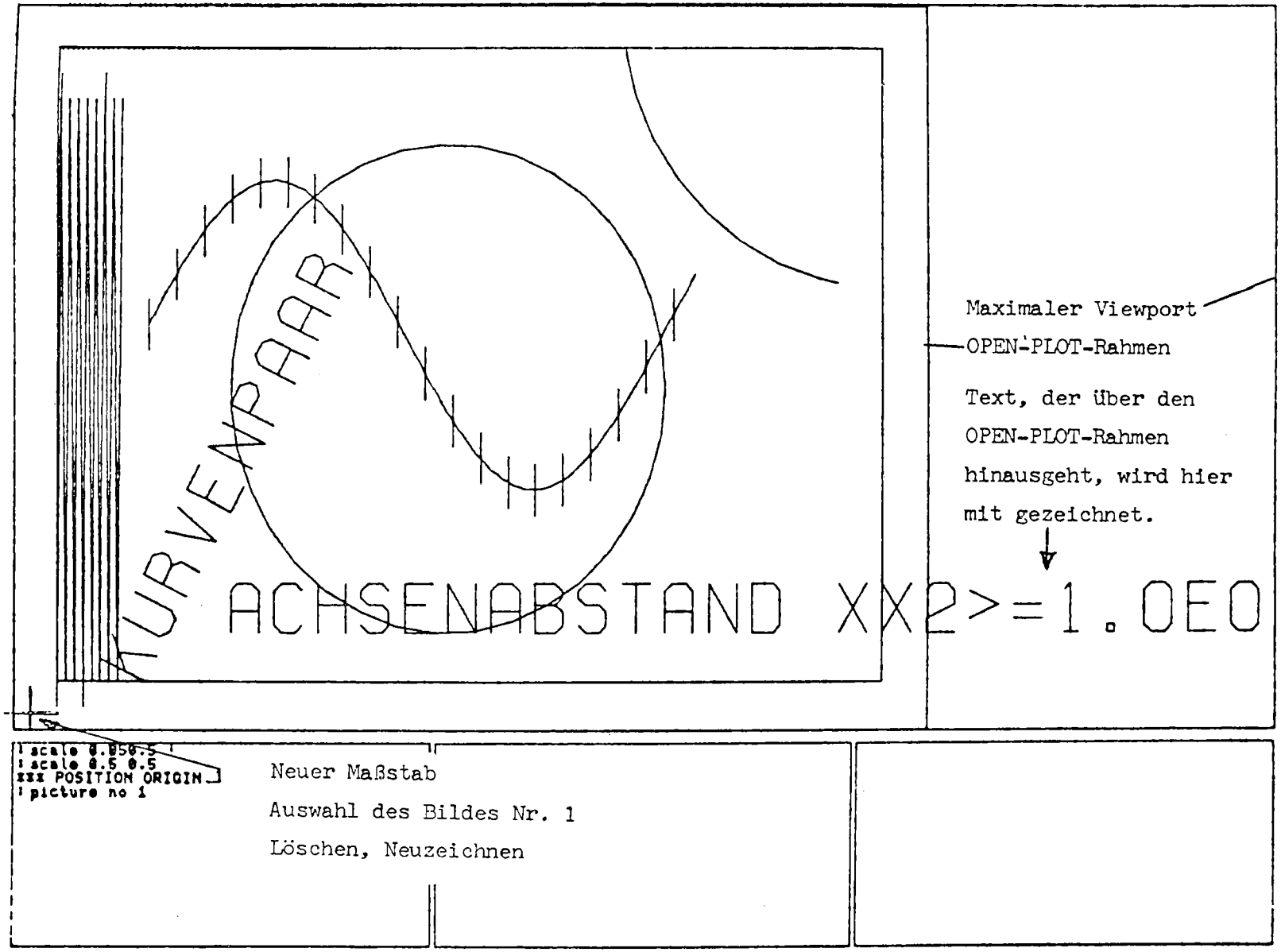
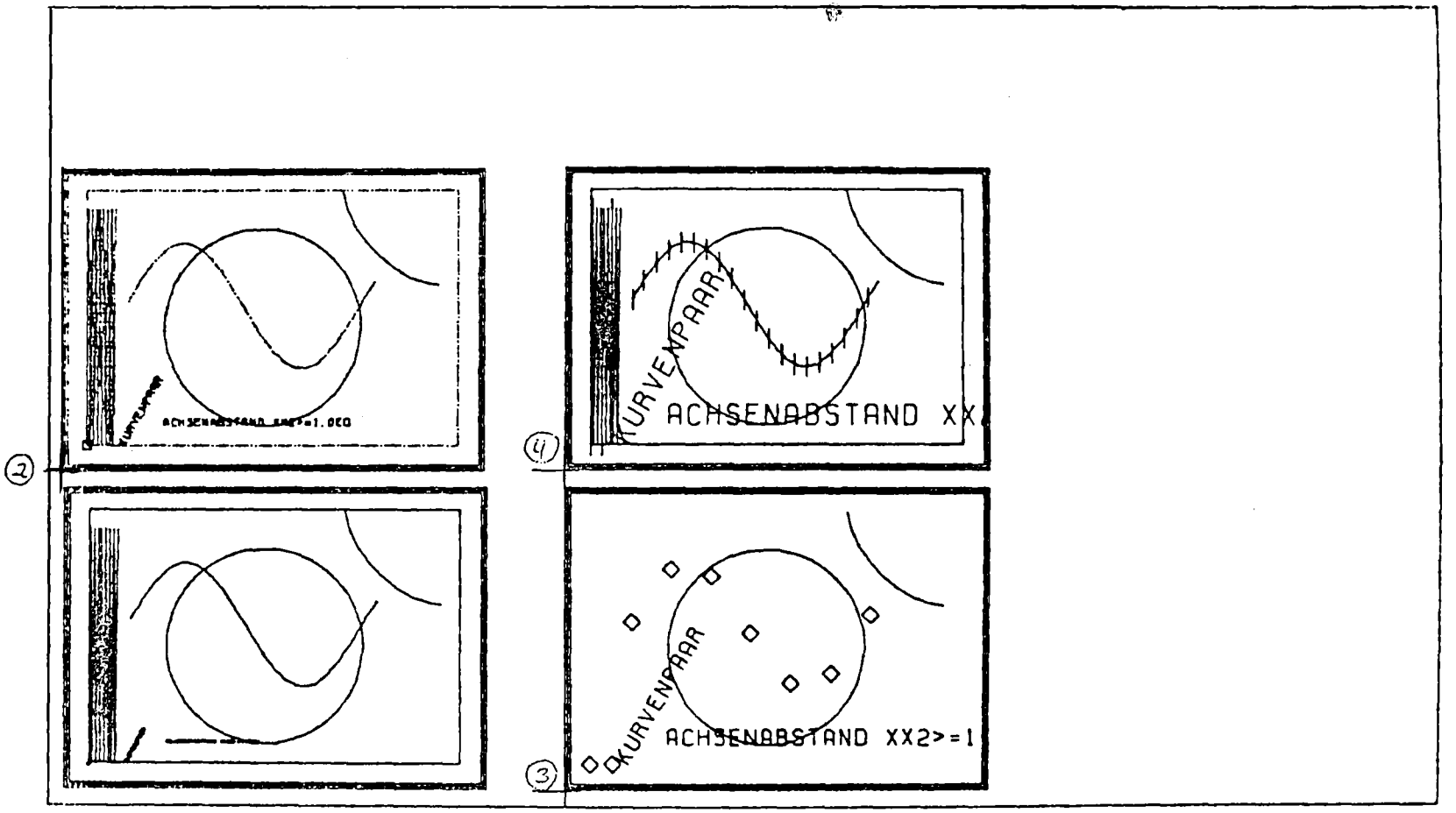


Abb.7.8: Gleiches Bild in maximaler Größe




```
! overlay
! scale 0.5 0.5
### POSITION ORIGIN
! picture next
! overlay
! scale 0.5 0.5
### POSITION ORIGIN
! p skip 1
! b
! o
! s 0.5 0.5
### POSITION ORIGIN
! p n
!
```

Mit OVERLAY und SCALE wird jedes neue Bild neu positioniert und ohne Löschen auf den Bildschirm gezeichnet. Dadurch können vier verschiedene Bilder gleichzeitig betrachtet werden.

Abb. 7.9: Mehrere Bilder gleichzeitig auf dem Bildschirm.

Die Fehlermeldungen bei der Plotfile-Interpretation sind im Abschnitt 10 aufgeführt.

7.3.4 Anleitung zur Erzeugung von Filmen

Mit einem Interpretier-Programm wird die graphische Information umgeschrieben,indem die Microfilmplotter-Software (Unterprogramm-Aufrufe) benutzt wird. Jedes OPEN PLOT beendet ein Filmbild. Das Programm wird nachfolgend aufgelistet.

```
10      //IRE147TM JOB (0147,330,P4125),ENDERLE,NOTIFY=IRE147,
20      //  TIME=(,30),MSGCLASS=A,REGION=600K
30      //  EXEC PGM=PFMOVIE
40      //STEPLIB DD DSN=TSO147.POST.LOAD,DISP=SHR
45      //SYSPRINT DD SYSOUT=A
50      //PLIDUMP DD SYSOUT=A
55      //SYSIN DD *
60        '16','GIPSY',1,28,28,1
70      //GIPSY  DD DSN=PFTEST1,UNIT=T1600,VOL=SER=PFTEST,
80      //  DISP=(NEW,PASS),LABEL=(1,SL),
90      //  DCB=(BLKSIZE=4000,LRECL=80,RECFM=FB,DEN=3)
100     //PLOTLIB DD DSN=TSO147.PFILE4.DATA,DISP=SHR
```

Das Programm kann sowohl unformatierte Plotfiles als auch formatierte beliebigen Formates lesen. Das verwendete Format wird in einem Kontrollblock des Plotfiles selbst angegeben.

Karte 10 ist die Jobkarte, auf Karte 30 wird das Interpretierprogramm aufgerufen. Karte 60 enthält die Eingabedatei für Kameratyp und Filmintensität, ab Karte 70 wird das Ausgabe-Band beschrieben. Karte 100 ist die Plotfile-Datei, die mit GIPSY beschrieben wurde.

Die DCB-Parameter des Plotfiles brauchen nicht angegeben zu werden. Wenn sie angegeben sind, müssen sie mit denen beim Beschreiben des Files übereinstimmen.

Zu Zeile 60:

//G.SYSIN Ausgabe-Dataset (6 Daten) im Beispiel:

'16','GIPSY', 1, 28, 28, 1

- '16' KAMERATYP 16 mm-Film
- 'GIPSY' Output-DD-Name GIPSY
- 1 Anzahl der Filmbilder pro Plot
- 28 Intensität der Symbole
- 28 Intensität der Linien

(Empfehlung: Intensität nicht unter 20 wählen)

- 1 Ablaufmeldung nach jedem Bild (0:keine Meldung)

Zu Zeilen 70 - 90:

//G.GIPSY

DD-Karte des Bandes, das zur Abarbeitung nach Jülich geschickt wird.

Unbedingt beibehalten:

Standard-Label SL

DCB-Parameter wie angegeben

Das Filmformat hat ein Höhe-zu-Breite-Verhältnis von 0.725. Hat das OPEN-PLOT-Fenster nicht das gleiche Verhältnis, so wird es auf das Filmformat so abgebildet, daß es verzerrungsfrei vollständig dargestellt werden kann.

7.3.5 Ausgabe des Plotfiles auf Plotter

Das Programm PFPLOT gibt Plotfiles auf verschiedene Plotter aus. Die Bildauswahl und Bildgröße kann durch Steuerparameter noch festgelegt werden. Außerdem kann ein Protokoll der Datensätze des Plotfiles erzeugt werden.
Beispiel für einen kompletten Job:

```
(1)  //IRE147P JOB (0147-330-P0H1A),ENDERLE,TIME=(,30)
     //*
(2)  //PLOT PROC PLOTTER=S,PLOTFIL=DUMMY,PLOTVOL=NULLFI
(3)  //G EXEC PGM=PFPLOT&PLOTTER.,COND=(9,LT)
(4)  //STEPLIB DD DSN=TS0147.POST.LOAD,DISP=SHR
(5)  //PLOTF  DD DSN=&PLOTFIL,DISP=SHR
(6)  //PLOTWK01 DD UNIT=SYSDA,DISP=(NEW,DELETE),SPACE=(TRK,(50,50))
(7)  //PLOTWK02 DD UNIT=SYSDA,DISP=(NEW,DELETE),SPACE=(TRK,(50,50))
(8)  //PLOTTAPE DD UNIT=T0800,LABEL=(,NL),VOL=SER=&PLOTVOL,
     //            DISP=(,PASS),DCB=DEN=2,DSN=&PLOTVOL.LE
(9)  //FT06F001 DD SYSOUT=A,DCB=(BLKSIZE=133,LRECL=133,RECFM=FA)
     //PLIDUMP  DD SYSOUT=A,DCB=(LRECL=125,BLKSIZE=2004,RECFM=VBA)
(10) //SYSPRINT DD SYSOUT=A,DCB=(LRECL=125,BLKSIZE=2004,RECFM=VBA)
(11) //INPUT    DD SYSOUT=A,DCB=(LRECL=125,BLKSIZE=2004,RECFM=VBA)
(12) // PEND
     //*
(13) // EXEC PLOT,PLOTFIL='TS0147.PF1.DATA'
(14) //G.SYSIN DD DSN=TS0147.INPUT.DATA,DISP=SHR
     //*
(15) // EXEC PLOT,PLOTFIL='TS0147.PF2.DATA',PLOTVOL=S147P
(16) //G.SYSIN DD DUMMY
     //*
(17) // EXEC PLOT,PLOTFIL='TS0147.PF3.DATA',PLOTVOL=X147P,PLOTTER=X
(18) //G.SYSIN DD *
      BILD=2,DRUCK=3;
      BILD=10,DRUCK=3,MASST='1'B,HOEHEVI=0.15;
     /*
     //
```

Karte (1) ist die JOB-Karte, von Karte (2) bis (12) wird die JCL-Prozedur PLOT definiert, die zur Ausgabe von PLotfiles auf Plotter benutzt werden kann. Die Prozedur besitzt die drei Parameter PLOTFIL, PLOTVOL und PLOTTER, die zur Angabe von Plotfile-Dataset-Name, VOL=SER-Name des Plotbandes und Plotter-Kennbuchstaben dienen.
In Karte (3) wird das Interpretier-Programm aufgerufen, dessen Name je nach verwendetem Plotter einen anderen Endbuchstaben trägt.
Karte (4) gibt die Bibliothek an, auf der das Interpretierprogramm gespeichert ist. In Karte (5) wird die Datei, die den Plotfile enthält, mit dem DD-Namen PLOTF verknüpft. Die Karten (6) bis (8) dienen der Steuerung des Plotters, auf die in Karte (9) angegebene Druck-Datei werden Fehler- und Ablaufmeldungen der Plottersoftware gedruckt. Karte (10) ist die Ausgabedatei für die Protokolle des Interpretierers, Karte (11) eine Datei, auf die ein Echo der Steuereingabe gedruckt wird.
Die Karten (13) bis (18) zeigen drei Aufrufe der Prozedur PLOT zur Ausgabe der Plotfiles TS0147.PF1.DATA, TS0147.PF2.DATA, TS0147.PF3.DATA. In Karte (13) wird der Plotfile nur gedruckt, da PLOTVOL nicht angegeben wurde. In Karte (15) wird der Plotfile auf den Statos-Plotter ausgegeben, hier muß PLOTTER nicht angegeben werden, jedoch mit PLOTVOL die VOL=SER-Angabe für das Plotband. In Karte (17) wird der Plotfile auf den Xynetics-Plotter ausgegeben. Die Karten (14), (16) und (18) geben den Eingabedataset für die Steuerparameter an. Der Eingabedataset ist entweder eine gespeicherte Datei (14), eine Primäreingabedatei(10) oder der Dataset ist DUMMY (16).

Steuerparameter

Durch das Setzen von bestimmten Steuerparametern ist es für den Benutzer möglich, den Ablauf des Programms zu beeinflussen.

Für folgende Größen ist dies vorgesehen:

- Bildnummer
- Druckausgabe (Protokoll)
- Setzen eines Viewports
- Liniendefinition. Festlegen von Strich-und Zwischenraumlängen

Im folgenden bedeuten:
i = Festkommazahl r = Gleitkommazahl B = Binärzahl
die Steuerparameter sind zur Darstellung in // eingeschlossen.

Ein Beispiel soll den Gebrauch der Steuerparameter veranschaulichen.

(1) DRUCK=2, VIEW='1'B, BREITEVI=0.4, HOEHEVI=0.3;

(2) BILD=3, DRUCK=3, S_4=10., G_4=5.;

(3) MASST='1'B, HOEHEVI=0.2;

(4) BILD=6;

Die Steuerparameter stehen in einer nichtnumerierten Datei.

Das erste Bild eines AGF-Plotfiles wird gezeichnet und es werden die Steuerparameter der ersten Zeile zugeordnet. Dann wird das dritte Bild gezeichnet mit den Steuerparametern der zweiten Zeile. Das vierte Bild erhält die Parameter der dritten Zeile; das sechste und alle weiteren Bilder erhalten die Standardwerte zugeordnet. Im Druckprotokoll erscheinen für jedes Bild vor dem Ausdruck der Datensätze die den Steuervariablen zugewiesenen Werte. Außerdem werden die eingelesenen Steuervariablen noch einmal zusammenhängend ausgedruckt.

Die Angaben für einzelne Parameter werden durch "," getrennt, für einzelne Bilder durch ";".
Werden keine Steuerparameter gesetzt, werden alle Bilder eines AGF-Plotfiles mit den angegebenen Standwerten gezeichnet.
Die Steuerparameter werden in folgenden näher erläutert.

Transformation

Bevor ein Bild gezeichnet werden kann, müssen die Koordinaten des Problemraumes (Window) umgesetzt werden in Koordinaten eines Zeichenbereiches (Viewport).

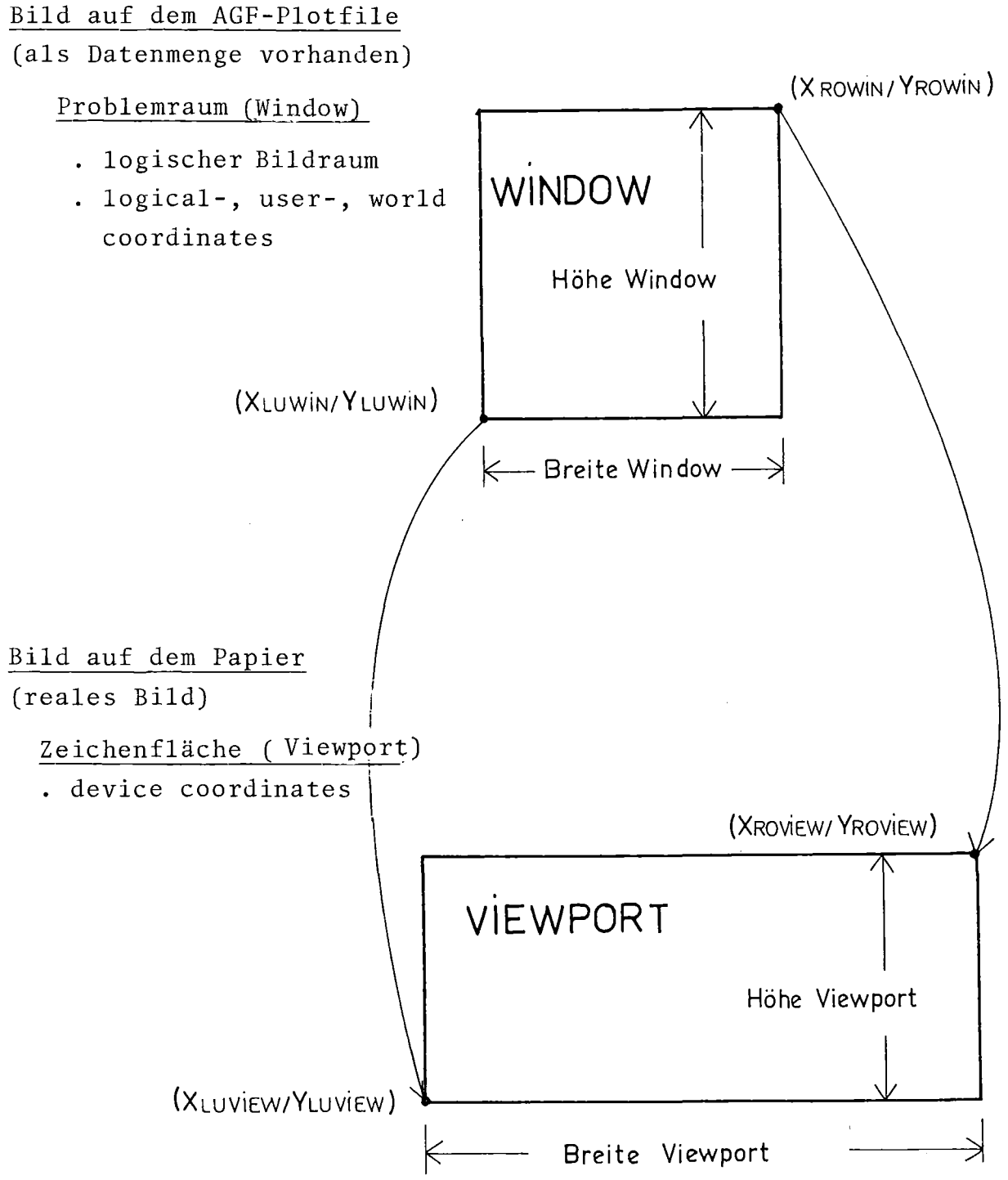


Abb.7.10: Transformation bei Ausgabe auf Plotter

(Xluwin/Yluwin) : = Koordinaten linke untere Ecke Window

(Xrowin/Yrowin) : = Koordinaten rechte obere Ecke Window

(Xluview/Yluview) : = Koordinaten linke untere Ecke Viewport

(Xroview/Yroview) : = Koordinaten rechte obere Ecke Viewport

Durch diese Wertepaare sind Höhe und Breite von Window und Viewport eindeutig festgelegt.
Vor dem Zeichnen werden alle Koordinaten und alle Längen transformiert.

Die beiden Eckpunkte des Windows erhält man aus dem Bild-Header1. Der linke untere Eckpunkt des Viewport hat immer die (relativen) Koordinaten (0/0). Der rechte obere Eckpunkt kann vom Benutzer gesetzt werden, er gibt somit Breite und Höhe des Viewport an. Damit ist es möglich, ein Bild vergrößert, verkleinert oder in x- und y-Richtung unterschiedlich verzerrt zeichnen zu lassen. Wird kein Viewport gesetzt, erfolgt eine Abbildung Window-Viewport im Verhältnis 1:1.

Setzen des Viewportes:

Es kann ein Viewport gesetzt werden (Abmessungen des Bildes):

/ VIEW = '1'B /

Als nächste Eingabe wird dann die Viewportbreite

/ BREITEVI = r /

und die Viewporthöhe

/ HOEHEVI = r /

in Metern erwartet.

Werden diese Werte nicht eingegeben, erfolgt eine Fehlermeldung (Fehler Nr.21) und Viewportbreite und Viewporthöhe werden gesetzt (BREITEVI=0.4, HOEHEVI=0.25).

Soll ein Bild maßstäblich abgebildet werden, wird ein Maßstab gesetzt:

/ MASST = '1'B /

Danach wird nur die Eingabe der Viewporthöhe

/ HOEHEVI = r /

in Metern erwartet.
Mit dieser gegebenen Viewporthöhe wird dann ein Maßstab vom Programm berechnet. Es ergibt sich aus dem Verhältnis von Windowhöhe zu Viewporthöhe.
Mit der Windowbreite und dem Maßstab wird dann die Viewportbreite vom Programm berechnet. Sie verhält sich zur Windowbreite wie die Viewporthöhe zur Windowhöhe.
Wird die Viewporthöhe nicht angegeben, erfolgt eine Fehlermeldung (Fehler Nr.22) und die Höhe wird gesetzt (HOEHEVI=0.25).

Standardwert:

Werden weder VIEW noch MASST gesetzt, ergibt sich die Größe des Windows (Abbildung im Verhältnis 1:1).

Papierverwaltung

Die Papierverwaltung steuert die Plazierung von Bildern auf dem Papier der Zeichengeräte. Diese haben unterschiedliches Papierformat:

Plotter	Papierlänge in cm	Papierbreite in cm
Xynetics	144.8	83.8
Calcomp	805.2	73.7
Statos	805.2	53.3

Die ungeraden Werte ergeben sich aus der Umrechnung von inch in cm (1 inch = 2.54 cm).

Der verwendete Algorithmus für die Papierverwaltung ist zwar nicht optimal, aber er gestattet doch eine gute Ausnutzung des Papiers. Das erste Bild wird in der linken unteren Ecke des Papiers plaziert. Ist die restliche Papierbreite noch größer als die Höhe des zweiten Bildes, dann wird dieses über das erste Bild gesetzt. Ist die restliche Papierbreite kleiner, dann wird das Bild rechts neben das erste Bild plaziert. Bei allen weiteren Bildern wird entsprechend verfahren. Es wird versucht, möglichst viele Bilder übereinander zu zeichnen. Ist dies nicht mehr möglich, wird das nächste Bild rechts im Anschluß an die gezeichneten Bilder am unteren Papierrand gezeichnet. Die Ausdehnung in x-Richtung des breitesten Bildes (XMAX) bestimmt dann den x-Wert des neuen Nullpunktes (linke untere Ecke) für dieses Bild. YMAX ist jeweils die Ausdehnung von übereinander gezeichneten Bildern in Y-Richtung. Dies ist genau die in GIPSY standardmäßig vorhandene Papierverwaltung für die OPEN-PLOT-Rahmen.

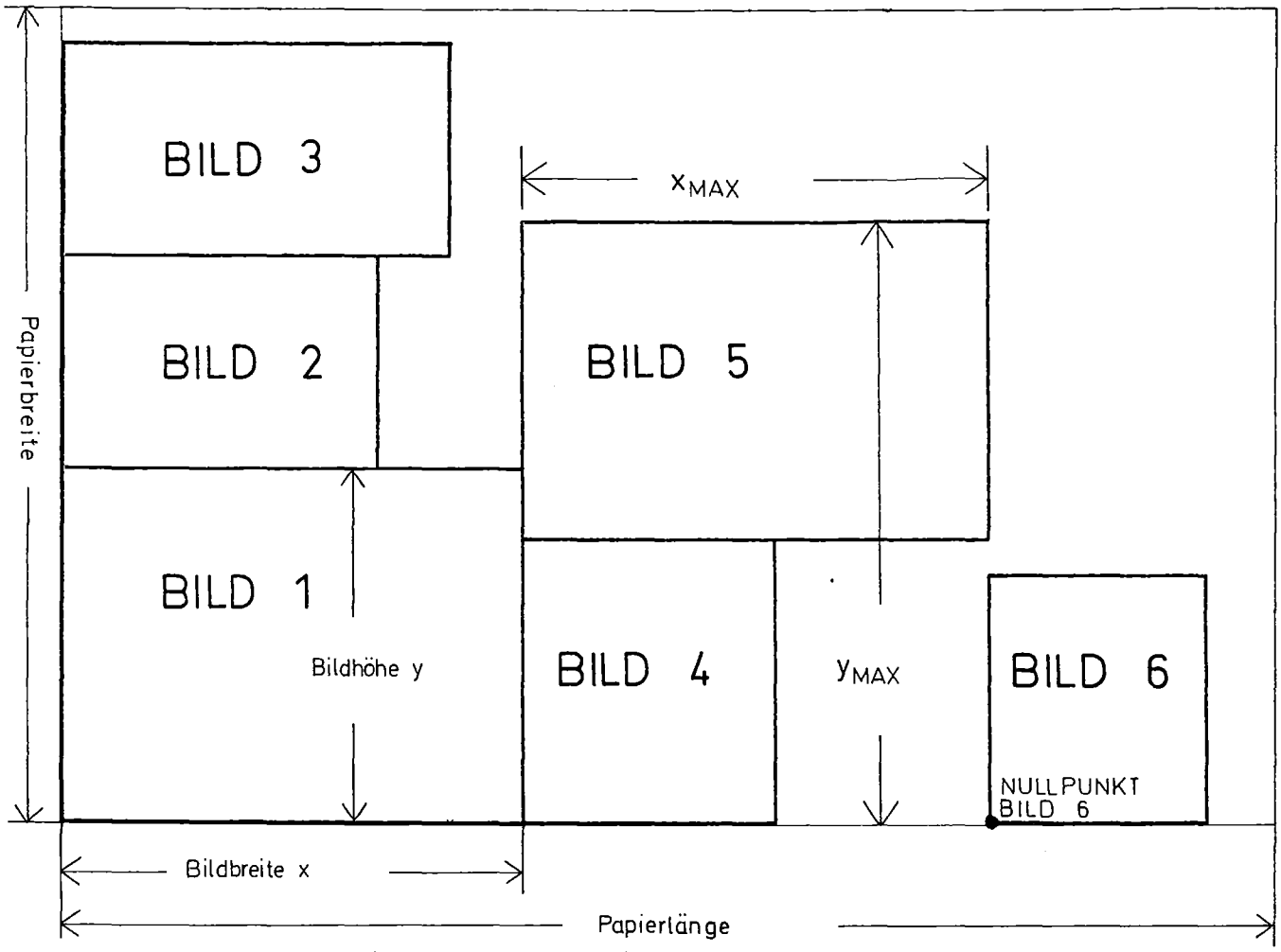


Abb. 7.11: Papierverwaltung bei Ausgabe auf Plotter

Es ist nicht möglich Bild 6 direkt rechts neben Bild 4 zu plazieren, da Bild 5 breiter ist und somit XMAX, das bedeutet den Nullpunkt für Bild 6, bestimmt (daher ist die Papierverwaltung auch nicht optimal).

Ist ein Bild höher als die Papierbreite, wird eine Fehlermeldung ausgegeben (Fehler 11), die Bildhöhe wird gleich der Papierbreite gesetzt und das Bild wird in y-Richtung gestaucht geplottet.
Ist ein Bild breiter als die noch vorhandene Papierlänge, wird eine Fehlermeldung ausgegeben (Fehler 12) und das Bild wird nicht geplottet. Die Papierverwaltung ist optimal, wenn alle Bilder gleich groß sind (gleicher Viewport).

Bildnummer

Durch das Setzen einer Bildnummer

/ Bild = i /

ist es möglich, ein bestimmtes Bild zu zeichnen und andere Bilder zu überspringen. Bildnummern dürfen aber nur in aufsteigender Reihenfolge angegeben werden, z.B. BILD=1; BILD=3; BILD=7; Soll das letzte Bild übersprungen werden, ist als letzte Bildnummer eine Zahl anzugeben, die größer ist, als die Anzahl der Bilder auf dem AGF-Plotfile.

Standardwert:

Wird bei den Parametern für ein Bild keine Bildnummer mit angegeben, so wird die Bildnummer des letzten gezeichneten Bildes um 1 erhöht und diesem neuen Bild zugewiesen; das auf dem AGF-Plotfile nächstfolgende Bild wird also gezeichnet.

Druckausgabe

Die Druckausgabe wurde vorgesehen, um die verarbeiteten Datensätze eines AGF-Plotfiles protokollieren zu können. Um den Umfang des Druckprotokolls zu steuern, ist es möglich für jedes Bild eine Drucksteuervariable zu setzen. Diese Variable kann die Werte 0, 1, 2, 3 annehmen.

Wert der Drucksteuervar.	Protokoll
0	kein Protokoll
1	File- und Bild-Header, Operator-Nachricht, Teilbildname
2	zusätzlich zu 1: Attributdatensätze und Länge und Anzahl der Elemente der Objekt-datensätze (bei Text werden die ersten 99 Zeichen gedruckt)
3	alle Datensätze mit allen Informationen

Die Drucksteuervariable

/ DRUCK = i / i=0,1,2,3

erlaubt es, die verarbeiteten Datensätze zu protokollieren. Der Umfang des Druckprotokolls kann gesteuert werden.

Standardwert:

DRUCK=1;

Behandlung der Attributangaben

Für alle Attributangaben gilt, daß sie in Abhängigkeit von der Drucksteuervariablen protokolliert werden.

Stiftangabe

- Stiftnummer

  nur beim Xynetics Plotter kann einer von 4 Stiften ausgewählt werden (CALL PEN(i)). Für die anderen Plotter ist der Aufruf CALL PEN nur ein dummy entry in der jeweiligen Plotter-Grundsoftware.

  Zuordnung:

gewünschte Stiftnummer	1 2 3 4	5 6 7 8	9 10 ...
ausgewählte Stiftnummer	1 2 3 4	1 2 3 4	1 2 ...

- Farbnummer

  wird nicht interpretiert. Eine bestimmte Farbe kann einer Stiftnummer zugeordnet und über diese ausgewählt werden (Xynetics); oder durch eine bestimmte Art der Strichlierung dargestellt werden (Calcomp, Statos).

- Intensitätsnummer

  wird nicht interpretiert.

- Strichdicke

  wird nicht interpretiert. Eine bestimmte Strichdicke kann einer Stiftnummer zugeordnet und über diese ausgewählt werden (Xynetics); oder durch eine bestimmte Art der Strichlierung dargestellt werden (Calcomp, Statos).

Symboltyp

- Symbolnummer

  wird umgerechnet in die jeweilige Nummer, die im Plotter-Software-Aufruf benötigt wird, um das richtige Symbol zu plotten.

- Markierung der Elemente der Objektdatensätze

  das gewünschte Element wird markiert; z.B. werden für Markierung = 2 das erste, dritte, fünfte ... Element markiert.

- Größe des Markierungssymbols

  das Symbol wird in etwa der 0.6-fachen Größe gezeichnet. (Die Größe ergibt sich aus dem gegebenen Raster. Symbole werden dabei in der Größe von Kleinbuchstaben gezeichnet).

Linientyp

- Linientypnummer

  eine der fünf definierten Linientypen wird in Abhängigkeit von der Typnummer gezeichnet. Für Typnummer = 1 (durchgezogene Linie) wird lediglich die Plotter-Software-Routine PLOT aufgerufen und die Linie wird durchgezogen gezeichnet. Dagegen wird die Typnummer = 0 (eigene Linientypdefinition), für Typnummer = 2 (gepunktete Linie), Typnummer = 3 (strichpunktierte Linie) und Typnummer = 4 (gestrichelte Linie) die Prozedur LINETYP aufgerufen, die diese Strichlierung zeichnet. Die Strich- und Zwischenraum-(gap) Längen sind vorgegeben.

  Sie betragen für:

Linientyp	Strich- und Zwischenraum in mm		Darstellung
gepunktet	s = 0.5	g = 2	.....
strichpunkt.	s1= 5	g1= 2	— · — · —
	s2= 2	g2= 2	
gestrichelt	s = 5	g = 2.5	— — — —

Ist der Strichabstand größer als der Punktabstand, dann wird bei Polygonzügen der jeweilige Strich oder Zwischenraum über diesen Punkt hinaus in Richtung des nächsten Punktes gezeichnet.

Steuerparameter:

Linienart		Standardwerte
gepunktete Linie		
Strichlänge in mm	/ S_2 = r /	0.5 mm
Zwischenraum1.in mm	/ G_2 = r /	2 mm
strichpunkt. Linie		
Strichlänge 1 in mm	/ S_3(1) = r /	5 mm
Zwischenraum1.1 in mm	/ G_3(1) = r /	2 mm
Strichlänge 2 in mm	/ S_3(2) = r /	2 mm
Zwischenraum1. 2 in mm	/ G_3(2) = r /	2 mm
gestrichelte Linie		
Strichlänge in mm	/ S_4 = r /	5 mm
Zwischenraum1. in mm	/ G_4 = r /	2.5 mm

Schriftform

- Zeichensatz

  wird nicht interpretiert. Schrift wird mit dem GIPSY-Zeichengenerator erstellt.

- Qualität

  wird nicht interpretiert.

- Höhe und Breite eines Zeichens

  ein Zeichen wird in der angegebenen Höhe und Breite gezeichnet. Ist das Bild transformiert worden, so wird auch die Höhe transformiert.

- Spacingpunkt S (Xs/Ys)

  Schriftwinkel wird berechnet

  Schriftwinkel alpha=arctan (Ys,Xs)

Behandlung der Objektdaten

Für alle Objektdatensätze gilt, daß sie in Abhängigkeit von der Drucksteuervariablen protokolliert werden.

Polygonzug

Die Koordinaten eines Punktes werden transformiert, in Abhängigkeit vom Linientyp werden die Punkte eines Zuges verbunden und in Abhängigkeit von Symbolnummer, Markierung und Symbolgröße markiert.

Punkthaufen

Die Koordinaten eines Punktes werden transformiert und in Abhängigkeit von Symbolnummer, Markierung und Symbolgröße markiert.

Text

- Anfangspunktkoordinaten

  die Koordinaten werden transformiert und das erste Zeichen des Textes wird in diesem Punkt plaziert.

- Text

  der Text wird in der vorgesehenen (wenn nötig transformierten) Größe unter dem Winkel alpha (aus Spacingpunkt in Schriftform berechnet) gezeichnet.

Nachrichtendatensätze

Sie werden nicht interpretiert, sondern lediglich in Abhängigkeit von der Drucksteuervariablen protokolliert.

7.3.6 Editieren des Plotfiles mit dem graphischen Arbeitsplatz

Der Plotfile kann mittels des Programms TO4081 auf den graphischen Arbeitsplatz Tektronix T4081 übertragen werden, dort mit dem Graphic Function Manager GFM bearbeitet und mit dem Programm FROM4081 wieder in eine Datei zurück übertragen werden. Folgende Unterlagen, die für das Arbeiten mit der T4081 die notwendige Information enthalten, können GIPSY-Anwendern bereitgestellt werden:

Leinemann, K.: Ein Interpretierer des AGF-PLOTFILE-Formates für den interaktiven graphischen Arbeitsplatz TEKTRONIX 4081, Oktober 1978, unveröffentlicht.

Kühl, D.: Ein interaktiver Interpretierer zur Transformation des Picture-Data-Base-Formates (Tektronix 4081) in das AGF-PLOTFILE-Format, April 1979, unveröffentlicht.

7.4 Aufruf von GIPSY-Modulen aus anderen Subsystemen

Dieses Kapitel baut auf den Kenntnissen des Kapitel 6.1 dieses Handbuches auf.
In diesem Zusammenhang ist es wichtig, zuerst die Frage zu klären, ob ein GIPSY-Prozeß in einem Subsystemmodul abläuft oder ob er sich über mehrere Module hinweg erstreckt. Im ersten Fall ist dies wie in Kapitel 6.1 beschrieben möglich. Im anderen Fall müssen die Wiederaufnahme und die anschließende Unterbrechung eines GIPSY-Prozesses jeweils in einem Modul ausgeführt werden. Der Aufruf des GIPSY-Subsystems und der Abschluß kann sowohl im Hauptprogramm als auch in einem getrennten Modul erfolgen.

In den folgenden Beispielen werden diese beiden Techniken gezeigt. Das Subsystem, das GIPSY-Fähigkeiten benutzen möchte, ist das Subsystem HELP.

Beispiel 1: (Aufruf und Abschluß von GIPSY im Hauptprogramm, Wiederaufnahme und Unterbrechung in einem Modul)

```
TEST : PROC OPTIONS(MAIN) REGENT( ... REGENT OPTIONS ...);
        .
        .
       DCL GI_PTR POINTER;
        .
        .
        ENTER HELP;
          .
          .
          ENTER GIPSY;
          END GIPSY LEAVE(GI_PTR);
            .
            .
            CALL PLTMOD(GI_PTR);
            .
            .
        END HELP;
END TEST;
```

```
PLTMOD : PROC(GI_PTR) REGENT(NOINIT,SUB=HELP);
          .
          .
         ENTER GIPSY REENTER(GI_PTR);
          .
          .
          .   (ANWENDUNG DER GIPSY - FAEHIGKEITEN)
          .
          .
         END GIPSY LEAVE(GI_PTR);
          .
          .
END PLTMOD;
```

Beispiel 2 (Aufruf, Abschluß,Wiederaufnahme und Unterbrechung von GIPSY in Subsystemmodulen)

```
TEST : PROC OPTIONS(MAIN) REGENT( ... REGENT OPTIONS ... );
        .
        .
       ENTER HELP;
        .
        .  (ANWENDUNG DER FAEHIGKEITEN DES SUBSYSTEMS HELP)
        .
       END HELP;
END TEST;
INIT : PROC REGENT(NOINIT,SUB=HELP);
       DCL HLP_PTR POINTER;
       HELP_PTR = #GI_PTR;
       ENTER GIPSY;
       END GIPSY LEAVE(HELP_PTR);
       #GI_PTR = HELP_PTR;
END INIT;
PLTMOD : PROC REGENT(NOINIT,SUB=HELP);
          DCL HELP_PTR POINTER;
          HELP_PTR = #GI_PTR;
          ENTER GIPSY REENTER(HELP_PTR);
            .
            .
            .  (ANWENDUNG DER GIPSY FAEHIGKEITEN)
            .
            .
          END GIPSY LEAVE(HELP_PTR);
END PLTMOD;
ENDE : PROC REGENT(NOINIT,SUB=HELP);
       DCL HLP_PTR POINTER;
       HELP_PTR = #GI_PTR;
       ENTER GIPSY REENTER(HELP_PTR);
       END GIPSY;
       #GI_PTR = HELP_PTR;
END ENDE;
```

Bei Beispiel 2 wird vorausgesetzt, daß die Module INIT und ENDE am Anfang und Ende des Subsystems HELP, also bei den Anweisungen ENTER HELP und END HELP aufgerufen werden. Außerdem befindet sich der Pointer, der bei Unterbrechung und Wiederaufnahme anzugeben ist, in der Variablen ≠≠ GI_PTR im HELP Subsystem-Common.

Das zweite Beispiel zeigt außerdem nochmals die Tatsache, daß zwischen einem ENTER GIPSY und einem END GIPSY, gleich ob es sich um Aufruf, Abschluß, Unterbrechung oder Wiederaufnahme handelt, nur subsystemspezifische Daten zugänglich sind, also Daten, die im GIPSY-Common stehen. Will man nun auf Daten des eigenen Subsystem-Commons zugreifen, so kann dies nur dann geschehen, wenn diese Daten durch Umspeichern in andere Variablen vor Wiederaufnahme von GIPSY gebracht wurden. Im folgenden Beispiel wird in den COMMON-Variablen ≠≠ DRGX und ≠≠ DRGY der X und Y-Ursprung einer Zeichnung abgespeichert. In einem Modul soll nun anhand dieser Werte über eine teilweise erzeugte Zeichnung ein Text positioniert werden.

```
TXTPLT : PROC REGENT( ... REGENT OPTIONS ... );
          .
          .
         GIPTR = #GI_PTR;
         HELPX = #ORGX;
         HELPY = #ORGY;
         ENTER GIPSY REENTER(GIPTR);
           .
           .
           PLOT (SHIFT2(TEXT),HELPX,HELPY);
           .
           .
          END GIPSY LEAVE(GIPTR);
 END TXTPLT;
```

7.5 GIPSY und das Betriebssystem

7.5.1 Stapeljobs

REGENT und sein Subsystem GIPSY sind für das Betriebssystem ein ganz normales Anwenderprogramm. Daher ist auch der GIPSY-Anwender gezwungen, JCL(Job Control Language=Betriebssystem-Steueranweisungen) in gewissem Umfang zu verwenden. Die REGENT-Prozedur macht jedoch die GIPSY-Anwendung ebenso einfach wie die Übersetzung und Ausführung eines FORTRAN- oder PL/1-Programms. Folgendes Beispiel enthält alle möglichen Angaben die in der JCL erforderlich sind. Die meisten GIPSY-Läufe benötigen nur einige dieser Angaben.

```
IRE147P   JOB   ....  ,REGION=region                    (1)
// EXEC REGENT,PUNIT=punit,PVOL=pvol,
// PLIB=plib,CPARM=cparm,GPARM=gparm                    (2)
//P.SYSIN DD *
  PLOT: PROC OPTIONS(MAIN)                            (3)
             REGENT(NODA,PLOT=plot);
     ENTER GIPSY FILE(file);
     DCL FILE FILE;
       .
       .
       .
     END GIPSY;
  FINISH;
  END PLOT;
//G.PLOTTAPE DD UNIT=T0800,LABEL=(,NL),                (4)
// DCB=DEN=2,VOL=SER=S147P
//G.file DD DSN=dsname,DISP=SHR                        (5)
```

① Zur REGION-Abschätzung siehe 7.2.3

② Wenn das Subsystem GIPSY nicht auf den REGENT-Bibliotheken liegt, ist durch "punit, pvol, plib" die Einheit, der Volume-Name und der erste Teil des DS-Namens der GIPSY-Bibliotheken anzugeben.

"cparm" ist der Compiler-Parameter für den PL/1-Compiler, z.B.:CPARM='OPT(2),A,AG,XREF,SOURCE'; "gparm" ist der Parameter für den G-Step. Durch "/" getrennt, werden Loader-, PL/1-Laufzeit- und Programmparameter angegeben, z.B. GPARM='PRINT/ISA(100K),REPORT/SONSTIGES'. Der letzte Teil des Parameters kann in der PL/1-üblichen Weise vom Anwenderprogramm selbst verarbeitet werden (als Parameter des Hauptprogramms mit dem Attribut CHAR(100)VAR). Wichtig für die Laufzeiteffektivität ist der Parameter ISA(n K) für das PL/1-Laufzeitsystem.

ISA:

Die ISA-Größe (ISA=Initial Storage Area) gibt denjenigen Speicherplatz an, den das PL/1-Laufzeitsystem für AUTOMATIC- und BASED-Variable und für interne Zwecke benötigt. Wird mehr Platz als angegeben benötigt, wird der zusätzliche Platz vom Betriebssystem (mit GETMAIN/FREEMAIN) angefordert. Dies kann zu einer erheblichen Zunahme der Rechenzeit führen. Wird weniger Platz benötigt als angegeben, so ist der unbenutzte Platz trotzdem für die ganze Dauer des G-Steps belegt. Die richtige Wahl der ISA ist daher wichtig für optimale Effektivität.Für rechenzeitoptimales Rechnen:ISA so groß wählen, daß kein Platz außerhalb der ISA angefordert wird. Für speicherplatzsparendes Arbeiten kann die ISA kleiner als der maximal benötigte Platz angegeben werden, etwa 2/3 des maximalen Platzes ist ein sinnvoller Wert. Wieviel Platz tatsächlich in der ISA benötigt wird, kann erst nach einem Rechenlauf festgestellt werden. Durch die Option REPORT wird vom PL/1-Laufzeitsystem der maximal erforderliche und der innerhalb und außerhalb der ISA belegte Platz auf die Printdatei PLIDUMP ausgedruckt. In der Zeile:

"AMOUNT OF PL/1 STORAGE REQUIRED"

wird der maximal erforderliche Speicherplatz angedruckt.

Beispiel für GPARM:

```
EXEC REGENT,GPARM='PRINT/ISA(60K),REPORT'
```

③ ④ Für den in PLOT=.... angesprochenen Plotter muß auch eine Steuerkarte für das Plot-Band bereitgestellt werden. Diese richtet sich nach den installationsabhängigen Erfordernissen des Betriebssystems. Benötigt wird eine PLOTTAPE-DD-Karte für Stapel-Jobs.

Die PLOTTAPE-DD-Karte hat folgendes Aussehen:

```
//G.PLOTTAPE DD UNIT=T0800,LABEL=(,NL),
// DCB=DEN=2,VOL=SER=αjob
```

α :	T = Calcomp-Plot, Tusche P = Calcomp-Plot, Kuli	PLOT-CALCOMP
	S = Statos-Plot	PLOT=STATOS
	X = Xynetics-Plot, Kuli Y = Xynatics-Plot, Tusche	PLOT=XYNETICS
job:	Die Buchstaben 4-8 des Jobnamens	

Auf die Übereinstimmung von Plotterangabe in der REGENT-Option PLOT= und im Volume-Namen von PLOTTAPE durch den Kennbuchstaben ist zu achten. Im Fehlerfall erhalten Sie in den meisten Fällen eine Operator-Nachricht "Plotband nicht interpretierbar".

⑤ Wird durch SAVE auf den AGF-Plotfile geschrieben, muß eine DD-Karte für ihn vorhanden sein, siehe auch 7.3. Die DCB-Parameter müssen RECFM=FB sein. Die Standardwerte für Satz- und Blocklänge sind LRECL=80, BLKSIZE=960. Die REGENT-Option PLOT= ist nicht erforderlich, wenn Zeichnungen nur auf den Plotfile ausgegeben werden.

7.5.2 GIPSY-Programme im TSO

GIPSY-Programme können auch im TSO ausgeführt werden. Zur Ausgabe von Zeichnungen ist allerdings ein Terminal T4014 oder T4081 notwendig. Die Ausgabe von Plotfiles auf ein Terminal ist im Abschnitt 7.3 beschrieben. Im folgenden wird die Übersetzung und Ausführung eines GIPSY-Programms während einer TSO-Sitzung beschrieben.

Das Programm befindet sich in einer Datei name.PLI mit den DCB-Parametern RECFM=FB und LRECL=80. Die Datei darf keine Steuerkarten enthalten, sondern nur das GIPSY-Programm:

```
prog: PROC OPTIONS(MAIN)
           REGENT(NODA,PLOT=TEKTRONIX);
      ENTER GIPSY;
       :
      END GIPSY;
END prog;
```

Zuerst muß der REGENT-Übersetzer gerufen werden:

```
REGTRAN name.PLI /'pparms'/ ;
```

"name.PLI" ist der Name der Datei, die das GIPSY-Programm enthält. "pparms" sind Parameter für den REGENT-Übersetzer, z.B. kann die ISA-Size bei genügendem Speicherplatz groß gesetzt werden. Nach einem "/" können Parameter für den REGENT-Übersetzer angegeben werden. Wichtig ist der LIST-Parameter:

```
LIST    : Ausgabe des GIPSY-Programms auf SYSPRINT
LIST=ddname: Ausgabe des Programms auf eine Datei mit
          DD-Namen "ddname"
NOLIST  : Unterdrücken der Programmliste
```

Bei LIST=ddname kann die Liste auf einen Sysout-Dataset gelegt werden.

Beispiel:

```
ALLOC F(REGLIST) SYSOUT(A)
REGTRAN G.PLI 'ISA(100K)/LIST=REGLIST'
FREE F(REGLIST) SYSOUT(A)
```

Der REGENT-Übersetzer schreibt das erzeugte PL/1-Programm auf eine Datei mit dem Namen PLS.name.PLI. Diese Datei muß dem PL/1-Compiler zum Übersetzen übergeben werden:

```
PLI PLS.name.PLI PRINT(x) INC /¯cparms_7
```

Der Parameter INC bzw. INCLUDE ist unbedingt erforderlich. Sonstige Compilerparameter "cparms", z.B. SOURCE oder OPT(2) können zusätzlich angegeben werden. Der PL/1-Compiler erzeugt ein Objektprogramm in der Datei PLS.name.OBJ. Diese Datei wird durch das REGENT-Kommando zu einem ausführbaren Modul gefunden und ausgeführt:

```
REGENT PLS.name.OBJ /¯'gparms'_7
```

In "gparms" werden Loader-Parameter und PL/1-Laufzeitparameter angegeben wie bei den Stapeljobs. Auch hier kann die ISA den Erfordernissen angepaßt werden.

Beispiel:

```
REGENT PLS.G.OBJ 'PRINT/ISA(50K)'
```

Im TSO wird man i.a. die Programme auf geringen Speicherplatzbedarf auslegen und dafür höhere Rechenzeiten in Kauf nehmen müssen (siehe dazu Abschnitt 7.2).

Plotausgabe im TSO

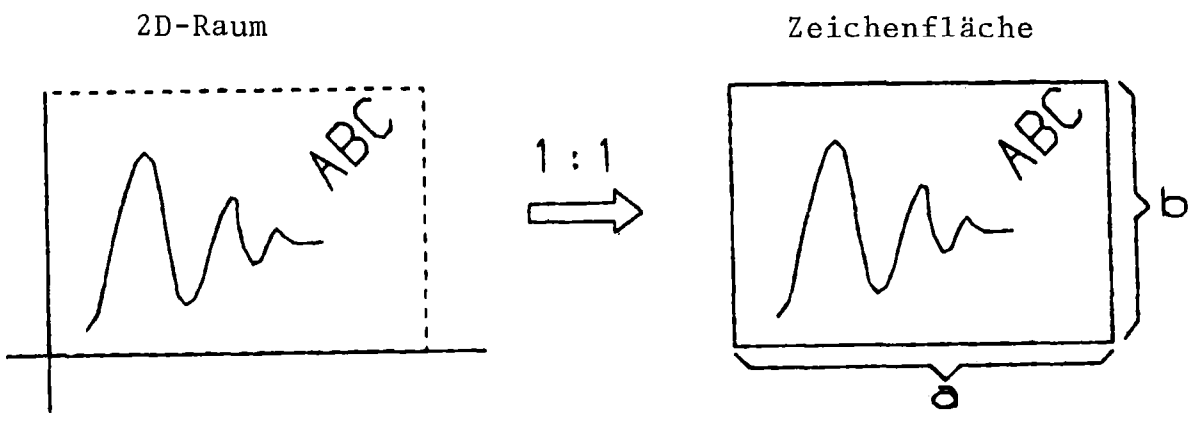
Als Ausgabegerät wird durch die REGENT-Option PLOT=TEKTRONIX ein graphisches Terminal ausgewählt. Bei Ein-/Ausgabe von Werten parallel zur Plot-ausgabe sind einige Besonderheiten zu beachten: Vor der Eingabe (mit GET) sollte das Terminal an eine Stelle positioniert werden, wo die Graphik nicht gestört wird, z.B. durch Zeichnen eines Textes mit Leerzeichen. Die Ausgabe von Werten und Texten sollte nicht durch PUT sondern durch PLOT(TEXT2(...)) erfolgen. Vor einem neuen OPEN PLOT und vor dem FINISH sollte als letztes ein Text mit Leerzeichen an eine freie Stelle der Zeichnung gezeichnet werden (z.B. unten links an die Position (0,0)).

8. Spezielle Probleme mit Lösungen

8.1 Transformation von 2D-Objekten ins Bildfenster

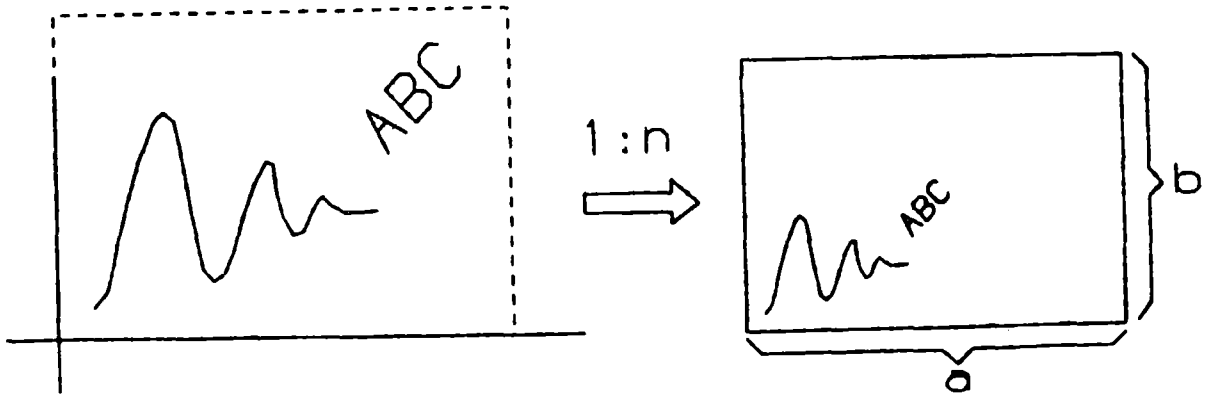
Für 10 verschiedene Transformationen vom 2D-GIPSY-Raum auf die Zeichenfläche werden hier die nötigen GIPSY-Anweisungen beschrieben. Die Abbildung wird durch die Anweisungen OPEN PLOT und VIEWING TRANSFORMATION gesteuert.

a) Abbildung 1:1 ohne Verschiebung




```
OPEN PLOT SIZE(a,b);
PLOT(....);
```

b) Abbildung 1:n ohne Verschiebung



```
OPEN PLOT SIZE(a,b);
VIEW SCALE(n);
```

oder

```
VIEW SCALE(n_x,n_y);
PLOT(....);
```

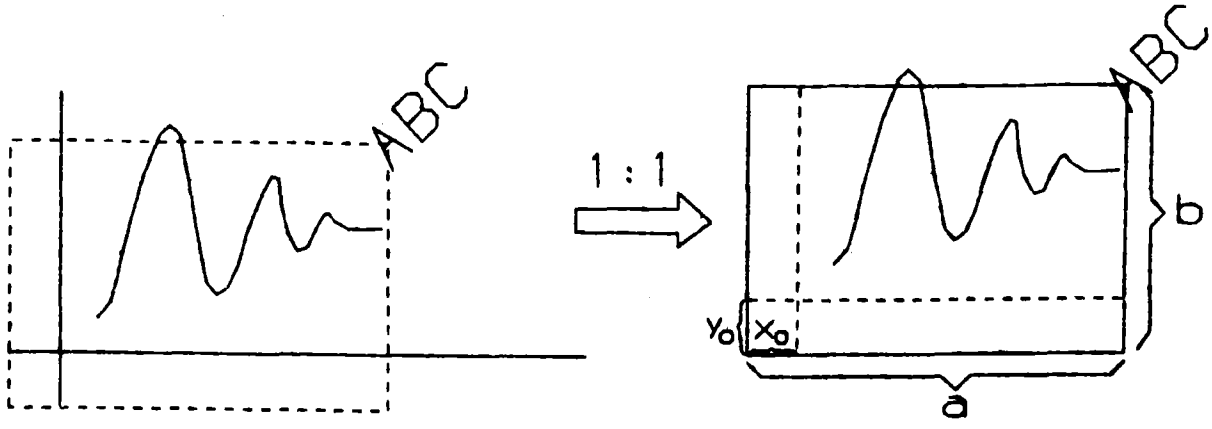
n : Gleichmäßige Skalierung in x- und y-Richtung

n_x, n_y : Skalierungsfaktoren in x- und y-Richtung sind unterschiedlich

Spiegelung um 2D-y-Achse : $n_x = -1$
Spiegelung um 2D-x-Achse : $n_y = -1$
Spiegelung um 2D-Ursprung: $n_x = n_y = -1$

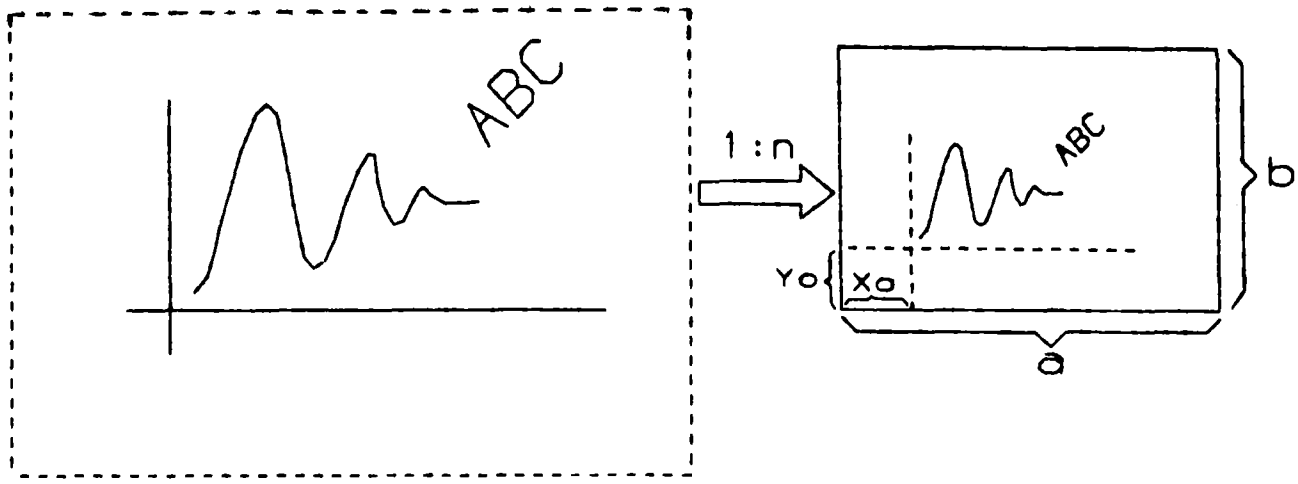
c) Abbildung 1:1 mit Verschiebung

Die Lage des 2D-Ursprungs auf der Zeichenfläche wird angegeben durch VIEW ORIGIN.




```
OPEN PLOT SIZE (a,b);
VIEW ORIGIN(x_o,y_o);
PLOT(....);
```

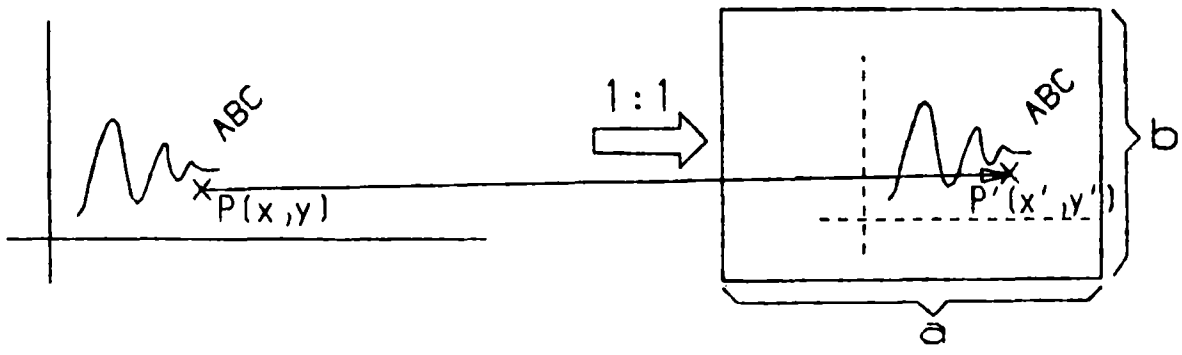
d) Abbildung 1:n mit Verschiebung




```
OPEN PLOT SIZE(a,b);
VIEW SCALE(n_x, n_y) ORIGIN(x_o, y_o);
PLOT( ... );
```

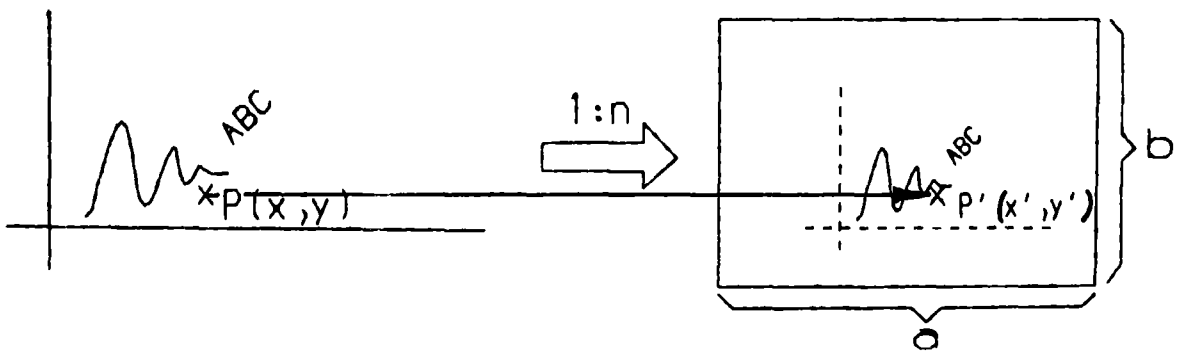
Achtung: (x_o, y_o) sind Zeichenflächenkoordinaten. Sie werden durch SCALE nicht beeinflußt.

e) Abbildung 1:1, so daß P auf P' liegt.




```
OPEN PLOT SIZE(a,b);
VIEW ORIGIN(x'-x, y'-y);
PLOT(....);
```

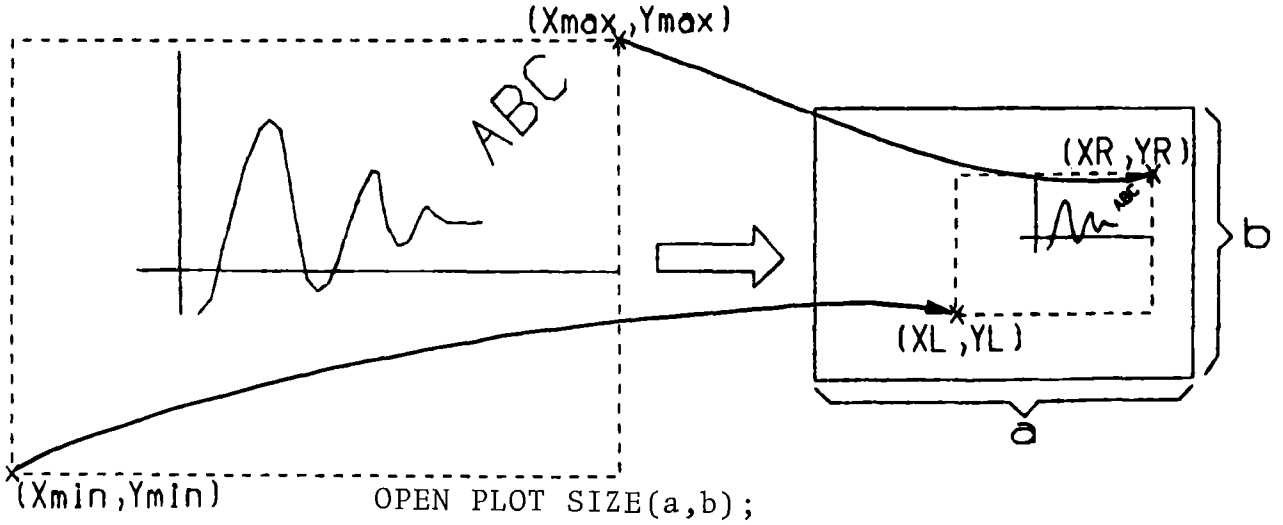
f) Abbildung 1:n, so daß P auf P' liegt.




```
OPEN PLOT SIZE(a,b);
VIEW ORIGIN(x'-n_x·x,y'-n_x·y)
     SCALE(n_x,n_y);
PLOT(....);
```

g) Window-Viewport-Transformation.

Der Ausschnitt im 2D-Raum x_{min} - x_{max} und y_{min} - y_{max} (Window) soll auf den Ausschnitt auf der Zeichenfläche x_L-x_R und y_L-y_R (Viewport) abgebildet werden:



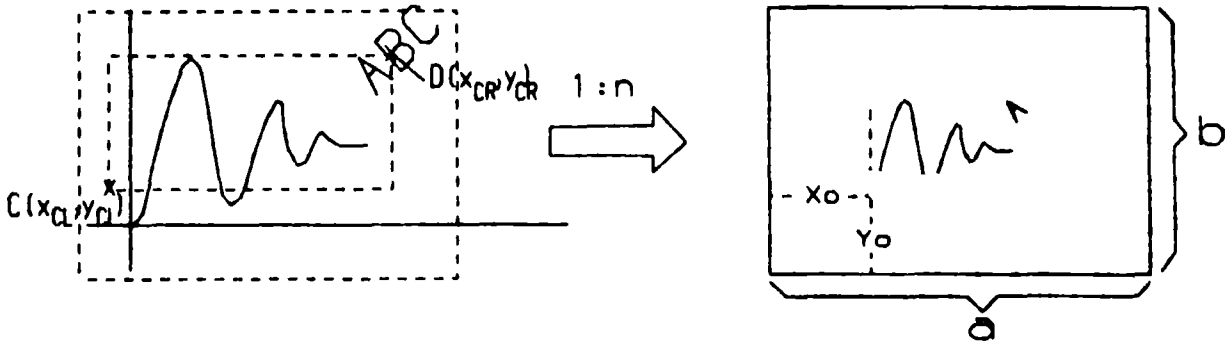

```
OPEN PLOT SIZE(a,b);
VIEW ORIGIN ((x_L * x_max-x_R * x_min)
/(x_max-x_min),(y_L * y_max-y_R * y_min)
/(y_max-y_min))
SCALE((x_R-x_L)/(x_max-x_min),
      (y_R-y_L)/(y_max-y_min));
PLOT(....);
```

Soll die Skalierung in x- und y-Richtung gleich sein, $n_x = n_y = n$, damit die Abbildung nicht verzerrt wird, so ist anzugeben:

```
N = MIN((xR-xL)/(xmax-xmin),
        (yR-yL)/(ymax-ymin));
VIEW ORIGIN(xL-N*xmin,yL-N*ymin)
     SCALE(N);
OPEN PLOT SIZE(a,b);
PLOT(... );
```

h) Abschneiden von Bildteilen.

Das Abschneiden von Teilen, die außerhalb einer angegebenen Rechteckfläche liegen, das Clippen, geschieht stets im 2D-GIPSY-Raum.
Der Clipp-Bereich wird im VIEW-Statement in 2D-GIPSY-Koordinaten angegeben. Das folgende Beispiel ist Fall d) mit zusätzlichem Clippen:

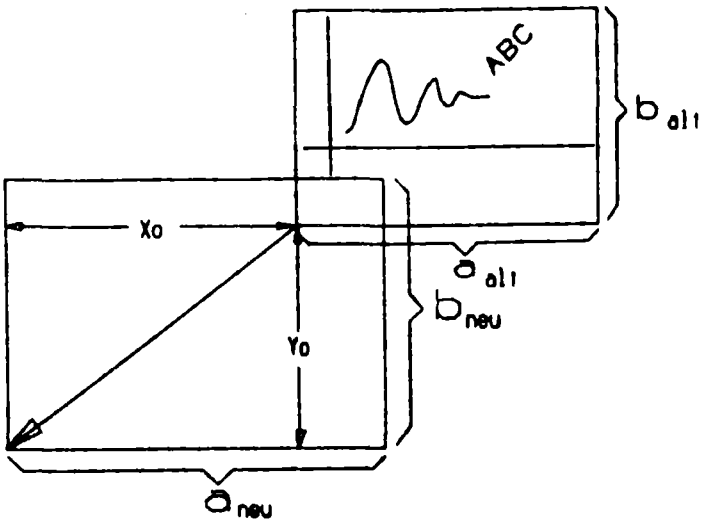


Auch bei allen anderen Beispielen ist die Option CLIPP zusätzlich möglich.

i) Positionieren des OPEN-PLOT-Rahmens auf der Zeichenfläche

Mit Hilfe der POSITION-Angabe im OPEN-PLOT-Statement kann die neue Zeichnung auf der Zeichenfläche beliebig positioniert werden. Im Standardfall übernimmt GIPSY die Anordnung auf dem Papier. Die Lage des linken unteren Expunktes des neuen Plotrahmens bezogen auf den gegenwärtigen Ursprung auf der Zeichenfläche wird angegeben.

Beispiel: Fall b) mit POSITION nach Fall c)



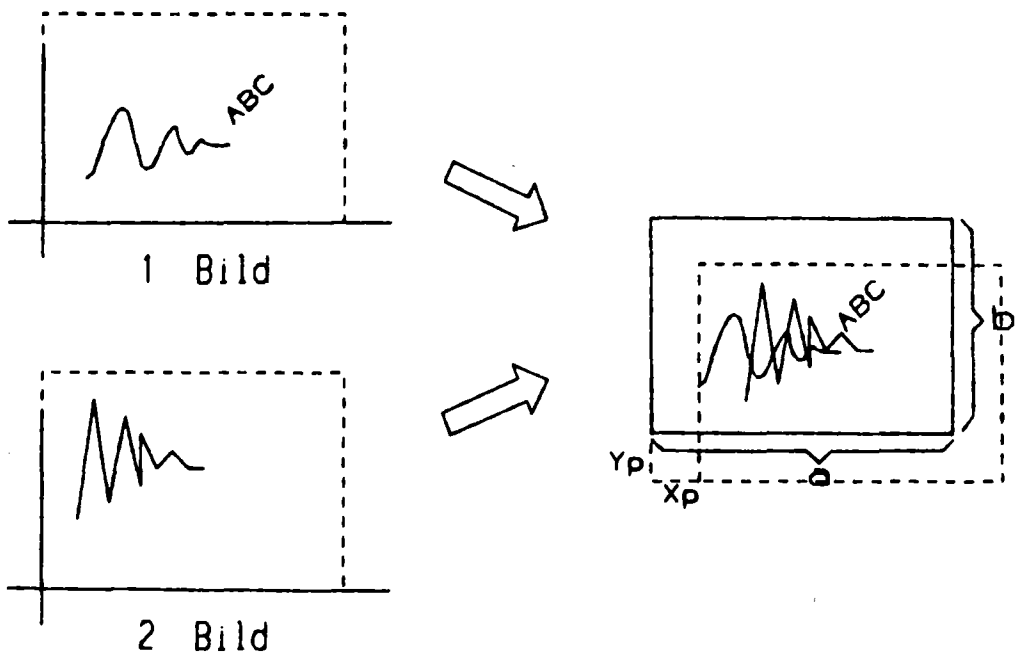
OPEN PLOT SIZE(a,b)POSITION(x_o,y_o);

VIEW.....usw. wie vorher

(Im Beispiel sind x_o und y_o negativ!)

k) Unterdrücken des Plotrahmens

Mit der NOFRAME-Option im OPEN-PLOT-Statement kann der Rahmen um das neue Bild unterdrückt werden.
Beispiel: Zweimal nacheinander Fall a)




```
OPEN PLOT SIZE (a,b);
PLOT(1.Bild);
OPEN PLOT SIZE (a,b) NOFRAME
    POSITION(-xp, -yp);
PLOT(2.Bild);
```

8.2 Transformation von 3D-Objekten ins Bildfenster

Die Transformation von 3D-Objekten erfolgt in zwei Stufen, zuerst eine Abbildung in den 2D-GIPSY-Raum und von dort weiter auf die Zeichenebene. Sind die Bildabmessungen im 2D-GIPSY-Raum bekannt, kann mit Hilfe von VIEW stets eine Abbildung auf die Zeichenfläche so erfolgen, daß alle Bildteile sichtbar werden.

a) Abbildung eines Körpers so, daß alle seine Teile im Bildfenster liegen. Die Perspektive ist dabei beliebig, eine maßstäbliche Abbildung wird nicht erreicht.
Ablauf: Es werden 8 3D-Punkte erzeugt, die einen den Körper umschließenden Quader beschreiben. Diese Punkte werden in den 2D-GIPSY-Raum abgebildet. Mit Hilfe der Window-Viewport-Transformation werden aus den Koordinatenwerten der transformierten Punkte die VIEW-Parameter so gesetzt, daß auch alle Teile des Körpers im OPEN-PLOT-Rahmen liegen müssen.

Vorgehen (Abb.8.1):

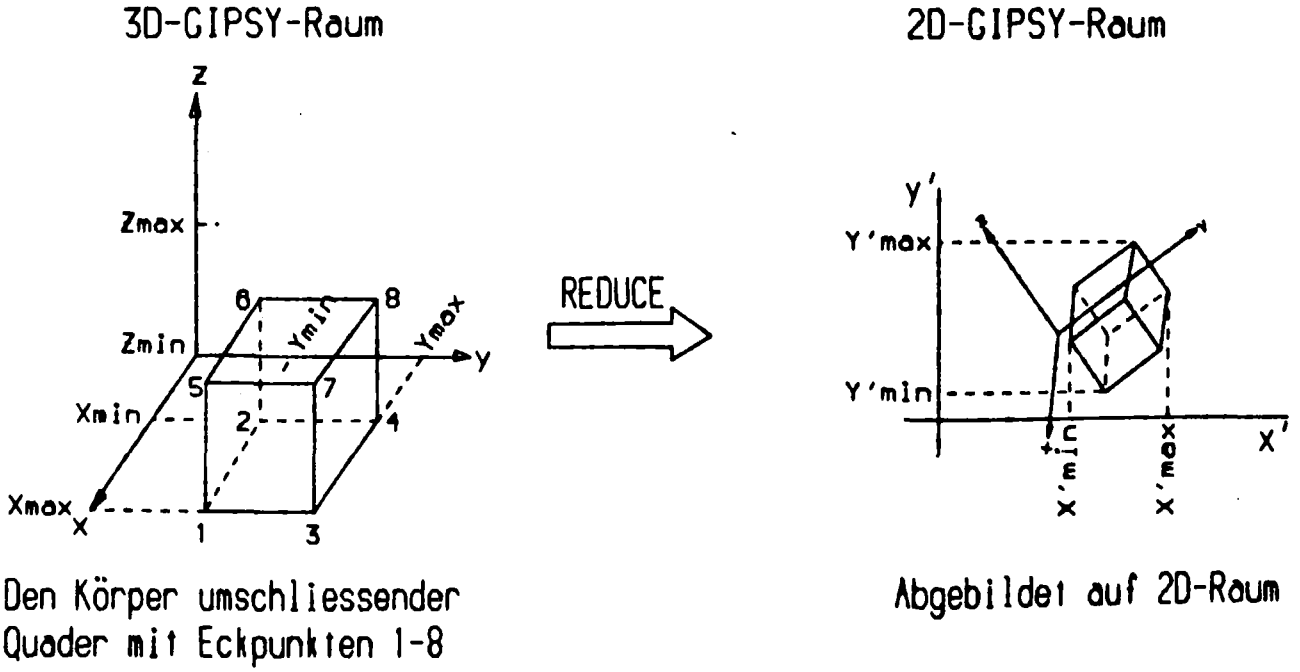
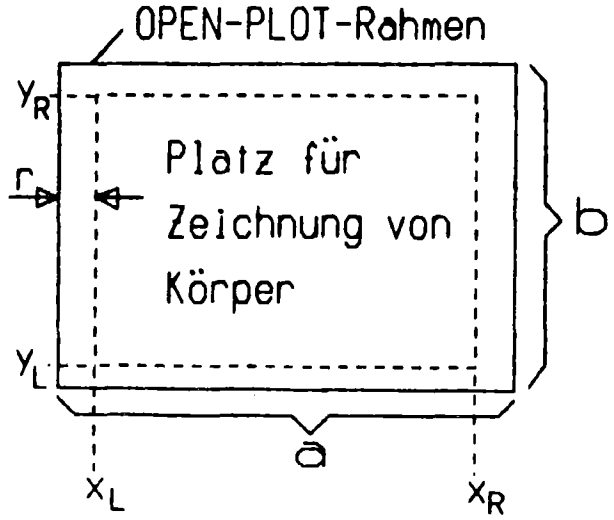


Abb.8.1: Lage von 3D-Objekt im 2D-Raum

GIPSY-Programm:

```
DCL PP POINT2;
DO I=1 TO 8;
   SET PP=REDUCE(POINT(
   Xmin+(MOD(I,2)=0)+Xmax*(MOD(I,2)=1),
   Ymin+(I=1|I=2|I=5|I=6)+Ymax*(I=3|I=4|I=7|I=8),
   Zmin*(I<=4)+Zmax*(Z>4)));
   W=COORD(PP,1);
   IF I=1 THEN X'min,X'max=W;
   ELSE DO;
      X'min=MIN(X'min,W);
      X'max=MAX(X'max,W);
   END;
   W=COORD(PP,2);
   IF I=1 THEN Y'min,Y'max=W;
   ELSE DO;
      Y'min=MIN(Y'min,W);
      Y'max=MAX(Y'max,W);
   END;
END;
/* Jetzt weiter bei 8.1, Fall g) */
OPEN PLOT SIZE(a,b);
VIEW ORIGIN(
     (XL*X'max-XR*X'min)/(X'max-X'min),
     (YL*Y'MAX-YR*Y'MIN)/(Y'MAX-Y'MIN) )
SCALE((XR-XL)/(X'max-X'min),
      (YR-YL)/(Y'max-Y'min) );
PLOT(...);
```

Die Werte (x_L,y_L) und (x_R,y_R), die die Lage des Viewports im OPEN-PLOT-Rahmen angeben, können dabei noch frei gewählt werden. Der Rahmen wird voll ausgenutzt, wenn $(x_L,y_L)=(0,0)$ und $(x_R,y_R)=(a,b)$ gewählt werden. Sinnvoller ist es, einen Rand freizulassen.



x_L-r , y_L-r , x_R-a-r , y_R-b-r

b) Abbildung eines 3D-Punktes P(x,y,z) auf einen Papierpunkt $P_p(x_p,y_p)$und festgelegte Skalierung S.
Mit dieser Transformation kann z.B. der Mittelpunkt eines Körpers auf die Mitte des OPEN-PLOT-Rahmens gelegt werden, die Skalierung S wird vorgegeben z.B. S = 1 oder S = 0.1.

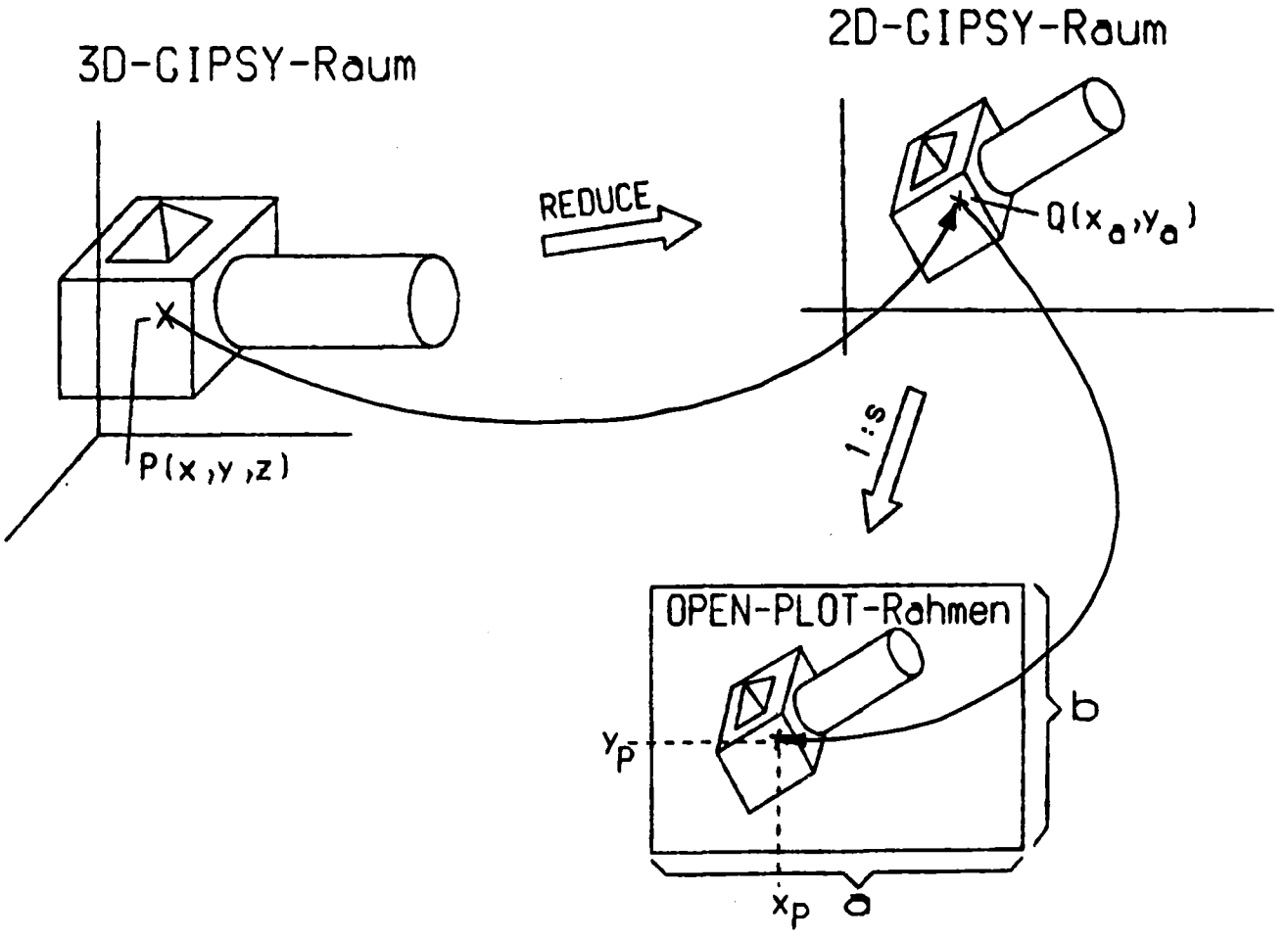


Abb. 8.2: Abbildung eines 3D-Punktes

```
DCL PP POINT2;
SET PP=REDUCE(POINT(x,y,z));
Xq=COORD(PP,1);
Yq=COORD(PP,2);
/* Jetzt weiter bei 8.1, Fall a) */
OPEN PLOT SIZE(a,b);
VIEW ORIGIN(Xp-s*Xq,Yp-s*Yq)
     SCALE (s);
PLOT(...);
```

8.3 Reliefdarstellung

Die perspektivische Darstellung eines Wertefeldes als Relief wird anhand von zwei Beispielen erläutert. Die Beispiele behandeln sowohl die Anwendungsvarianten der Reliefdarstellung als auch den Aufbau des Wertefeldes, um den Sichtbarkeitsalgorithmus bei der perspektivischen Darstellung richtig anzuwenden.

8.3.1 Anwendungsformen

In Abb.8.3 werden die 4 Darstellungsarten des Reliefs mit dem dazugehörigen Programm gezeigt.
Eine quadratische Matrix wird gefüllt mit x-, y- und z-Koordinaten, und zwar so, daß das Wertefeld im 2. Quadranten liegt ($x \leq 0$, $y \geq 0$). Die Blickrichtung wird so gewählt, daß die Laufrichtung des Feldindizes I und J mit der Blickrichtung zusammenfällt, d.h. daß der Sichtbarkeitsalgorithmus richtig angewendet wird.

Das 1.Relief, bei dem die Variation des 1. Index durchgeführt wird, besteht aus hintereinander gelegten Polygonzügen, die quer zur x-Richtung liegen. Entsprechendes gilt für Bild 2.

Bei der Standard-Anwendung des Reliefs werden die beiden ersten Bilder übereinander gelegt, so daß eine netzartige Oberfläche des Reliefs entsteht.

Im 4. Bild schließlich wird ein Relief gezeigt, bei dem nur der sichtbare Teil der Reliefoberseite abgebildet wird.

8.3.2 Test zum Sichtbarkeitsalgorithmus

Wie in Kapitel 6.7.1 schon ausgeführt, muß zur richtigen Anwendung des Sichtbarkeitsalgorithmus, der im GIPSY-Relief benutzt wird, die Datenmatrix nach einer bestimmten Regel gefüllt

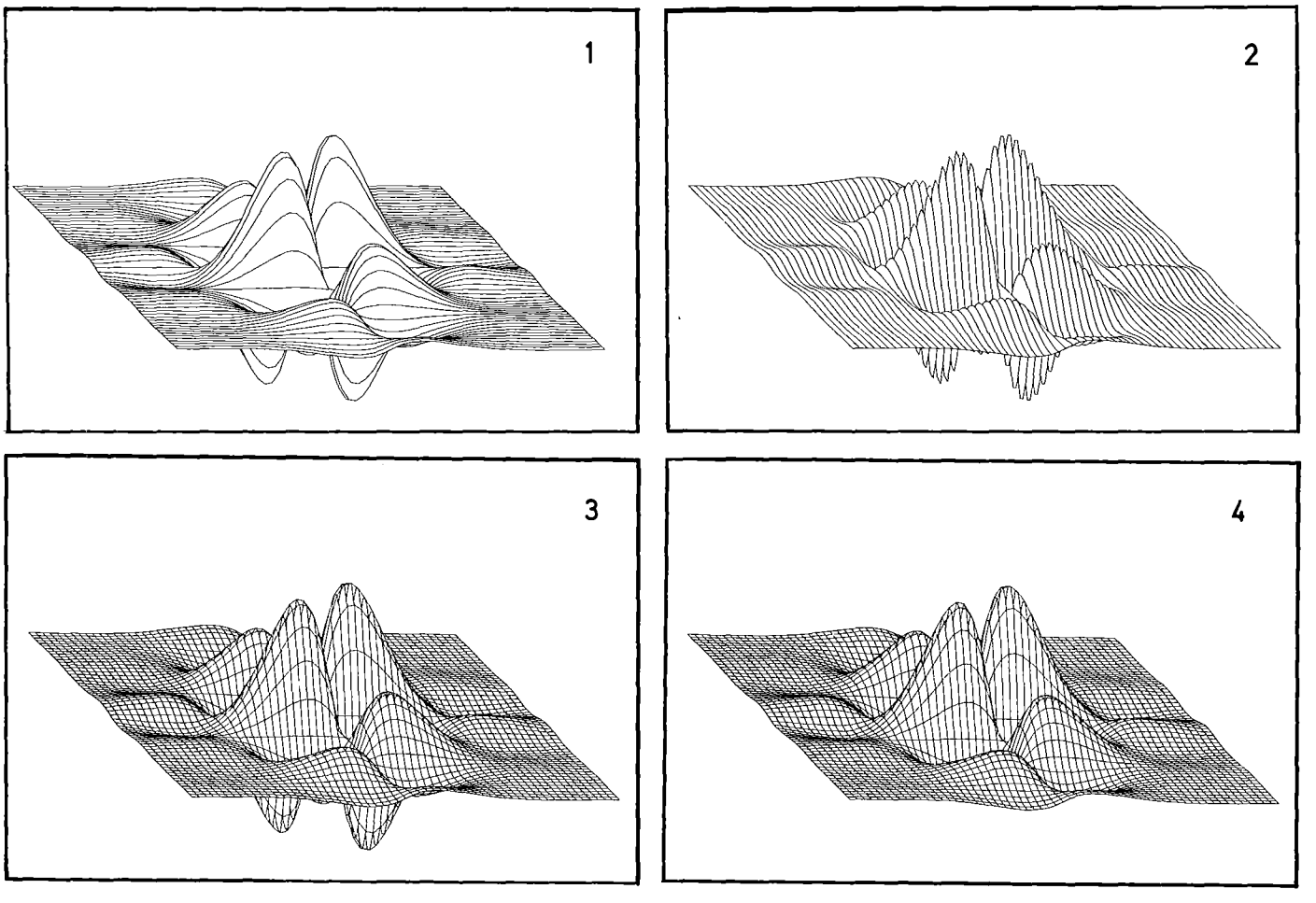


Abb.8.3: Die vier Darstellungsformen eines Reliefs

```
ALGO:PROC OPTIONS(MAIN)
     REGENT(INIT,NODA,MPOOL=300000,PLOT=STATOS);
ENTER GIPSY;
DCL FELD(61,61,3);
CHANGE PROJ ORIGIN (0.200, 0.090)
       PROJ PARA   (+1., -1.,+1.);
DO NP1=1 TO 61;
   BEAMV=SINDEXP(15.,NP1,2,30);
   FELD(NP1,*,1)=0. - 0.003 * (NP1-1);
   DO NP2=1 TO 61;
      FELD(NP1,NP2,2)=0.00254*3*(NP2-1);
      FELD(NP1,NP2,3)=0.5*BEAMV*SINDEXP(7.5,NP2,2,20);
   END;
END;
/***********    BILD  1    **********************************/
OPEN PLOT SIZE(0.750 , 0.480);
     PLOT REL OF (FELD) INDEXVARIATION (1);
/***********    BILD  2    **********************************/
OPEN PLOT SIZE(0.750 , 0.480);
     PLOT REL OF (FELD) INDEXVARIATION (2);
/***********    BILD  3    **********************************/
OPEN PLOT SIZE(0.750 , 0.480);
     PLOT REL OF (FELD);
/***********    BILD  4    **********************************/
OPEN PLOT SIZE(0.750 , 0.480);
     PLOT REL OF (FELD) LINES ('ABOVE');
/*************************************************************/
SINDEXP:PROC(A,N,M,L) RETURNS(DEC FLOAT(6));
    DCL (A,B) DEC FLOAT(6);
    DCL (K,L,M,N) BIN FIXED(15);
    K=3*N-93;
    B=SIN(A*SIND(K/M));
    B=B*EXP(-ABS(K/L));
    RETURN(B);
END SINDEXP;
/*************************************************************/
END GIPSY;
MESSAGE ACTIVE DEBUG;
FINISH;
END ALGO;
```

Programm zu Abb.8.3

werden. Dies wird anhand von Abb.8.4 verdeutlicht, wo man die Auswirkungen falscher Indexlaufrichtungen in den Reliefs erkennt.

Das Wertefeld liegt im 1. Quadranten (positive x- und y-Werte) und stellt einen Berg dar (z-Werte größer oder gleich Null). Die Datenmatrix X enthält größer werdende Koordinatenwerte mit ansteigenden Indizes I und J.

Die Bildebene ist die Ebene y-z, wobei die x-Achse aus der Bildebene zum Betrachter zeigt (PI_N(1., 0., 0.)). Als Projektionsrichtung (PARA(+1., +1., +1.)) ist die Blickrichtung von oben aus dem 1. Quadranten gewählt. Der Nullpunkt wird durch ein kleines Kreuz gekennzeichnet.

In Bild 1 laufen beide Indizes I und J entgegengesetzt der Blickrichtung, so daß die Verdeckung der Relieflinien völlig falsch ermittelt wird, nämlich von "hinten".

In den nachfolgenden Bildern 2 und 3 fallen abwechselnd die Indizes I oder J mit der Blickrichtung zusammen (Umdrehen der Laufrichtung beim Umspeichern in die Matrix Y). Auch hier werden falsche Reliefs gezeichnet.

In Bild 4 schließlich laufen die Indizes I und J in Blickrichtung, so daß hier der Sichtbarkeitsalgorithmus richtig angewendet wird.

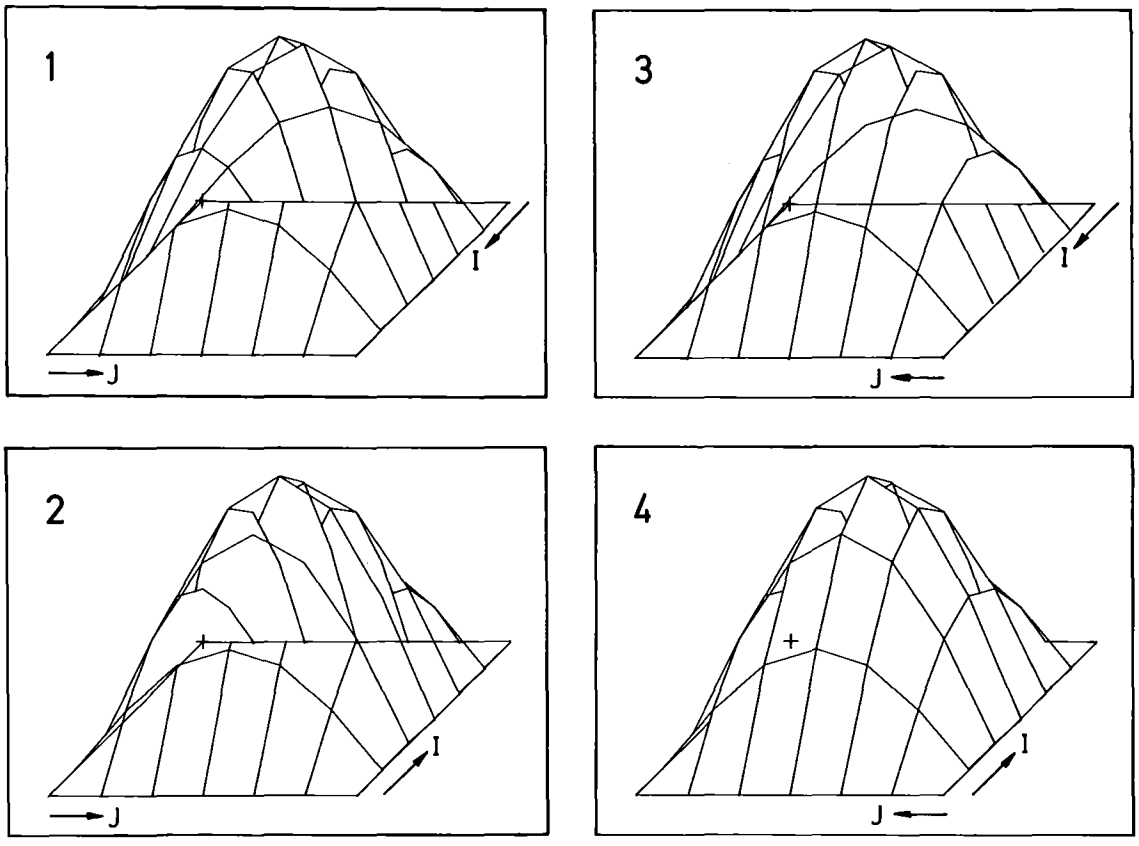


Abb.8.4: Ansicht eines Reliefs mit verschiedenen Indexlaufrichtungen.

```
RELI:PROC OPTIONS(MAIN)
      REGENT(INIT,NODA,MPOOL=100000,PLOT=STATOS);

   DCL M   BIN FIXED(15) INIT(7),
       N   BIN FIXED(15) INIT(7),
       (I  ,J  ) BIN FIXED(15);
   ENTER GIPSY;
     DCL X(M,N,3) DEC FLOAT(6);
     DCL Y(M,N,3) DEC FLOAT(6);
     DCL DA DEC FLOAT(6);
     DA = 2.*3.14159/12.;
     DO J=1 TO N;
        DO I=1 TO M;
              X(I,J,1) = (I-1)*DA;
              X(I,J,2) = (J-1)*DA;
              X(I,J,3) = SIN(X(I,J,1))*SIN(X(I,J,2));
              X(I,J,1) = X(I,J,1)*1.E-2;
              X(I,J,2) = X(I,J,2)*2.E-2;
              X(I,J,3) = X(I,J,3)*5.E-2;
        END;
     END;
     DCL NP POINT;
     SET NP = POINT(0.,0.,0.);
     EDIT SYMBOL (3) OF (NP);
     CHANGE PROJ PLAN (1., 0., 0.)
            PROJ PARA (+1., +1., +1.)
            PROJ ORIGIN (0.040, 0.040);
   /*************** BEISPIEL 1  ***************************/
   Y = X;
   CALL REP;
   /*************** BEISPIEL 2  ***************************/
   DO J=1 TO N;
      DO I=1 TO M;
         Y(M-I+1,J,*) = X(I,J,*);
      END;
   END;
   CALL REP;
   /*************** BEISPIEL 3  ***************************/
   DO J=1 TO N;
      DO I=1 TO M;
         Y(I,N-J+1,*) = X(I,J,*);
      END;
   END;
   CALL REP;
   /*************** BEISPIEL 4  ***************************/
   DO J=1 TO N;
      DO I=1 TO M;
         Y(M-I+1,N-J+1,*) = X(I,J,*);
      END;
   END;
   CALL REP;
   /*******************************************************/
   REP:PROC;
   OPEN PLOT SIZE (0.110 , 0.080 ) ;
   PLOT (NP);
   PLOT RELIEF OF (Y);
   END REP;
   /*******************************************************/
END GIPSY;
MESSAGE ACTIVE DEBUG;
FINISH;
END RELI;
```

Programm zu Abb.8.4

8.4 Körperdarstellung nach DIN 6

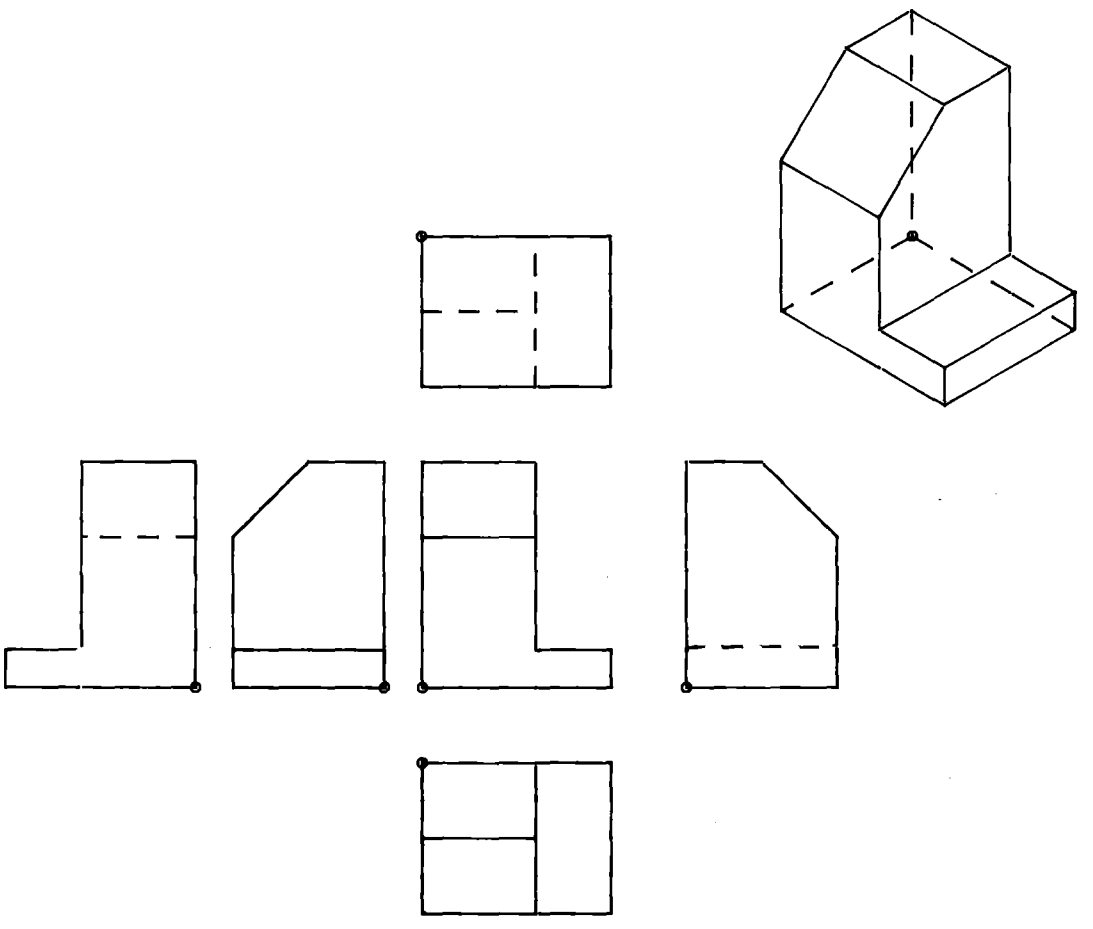
In diesem Beispiel wird ein Unterprogramm angegeben,mit dem ein beliebiger Körper B gemäß DIN6 dargestellt wird. Als Hilfsgrößen werden benutzt:

PREF	ein Referenzpunkt des Körpers
P(7)	Punkte der GIPSY-2D-Welt, die der Darstellung des Punktes PREF in Draufsicht, Ansicht von rechts, Vorderansicht, Ansicht von links, Rückansicht, Untersicht und Isometrie entsprechen.

Dabei wird angenommen:

'Draufsicht'bedeutet eine Betrachtung aus $+x_3$(oder z) Richtung
'Ansicht von rechts'bedeutet eine Betrachtung aus $+x_2$(oder y)Richtung
'Vorderansicht'bedeutet eine Betrachtung aus $+x_1$ (oder x) Richtung
'Isometrie'bedeutet eine Betrachtung aus der Raumdiagonalen, die von den positiven (x_1,x_2,x_3)-Achsen eingeschlossen wird, jedoch so verzerrt, daß Längen in allen (x_1,x_2,x_3)-Richtungen längentreu wiedergegeben werden.

In der Prozedur DIN6 werden die Projektionen ausgeführt. Bei Draufsicht und Untersicht sind wegen der Sonderregel für Projektionen in x_3-Richtung (siehe Kap.5.2.3)Drehungen in der Bildebene erforderlich. Die Isometrie wird aus einer normalen GIPSY-Projektion durch eine Dehnung um $\sqrt{3/2}$ in allen Richtungen erzeugt.



```
 PROGRAM:PROCEDURE OPTIONS(MAIN) REGENT(PLOT=STATOS,MPOOL=800000);
   DCL AGF FILE;
ENTER GIPSY FILE(AGF);
SAVE    PLOT   ;
DCL BEZ_P POINT;
DCL POSITION(7) POINT2;
DCL A COLLECTION;
DCL RESULT COLLECTION;
CHANGE UNIT LENGTH(CM);
OPEN PLOT SIZE(20,20);

/*EXAMPLE ERZEUGT DEN KOERPER "A" ALS KOLLEKTION UND PUNKT "BEZ_P" */
DCL  EXAMPLE ENTRY(PTR /*REGENT-BASIS           */,
                   PTR /*GIPSY-OBJEKT COLLECTION*/,
                   PTR /*GIPSY-OBJEKT POINT     */) EXTERNAL;
CALL EXAMPLE(QQ /*REGENTBASIS*/,A,BEZ_P);
/* BEI INTERNER PROCEDURE EXAMPLE WUERDE DIE DEKLARATION GANZ
   UND IM AUFRUF DAS "QQ" ENTFALLEN.ALSO:
   CALL EXAMPLE(A,BEZ_P);                                          */


CHANGE UNIT LENGTH(CM);
SET POSITION(1)=POINT2( 7.5, 5.5); /* FUER    DRAUFSICHT  */
SET POSITION(2)=POINT2( 4.5, 6.5); /* FUER  RUECKANSICHT  */
SET POSITION(3)=POINT2( 7. , 6.5); /* FUER RECHTSANSICHT  */
SET POSITION(4)=POINT2( 7.5, 6.5); /* FUER VORDERANSICHT  */
SET POSITION(5)=POINT2(11. , 6.5); /* FUER  LINKSANSICHT  */
SET POSITION(6)=POINT2( 7.5,12.5); /* FUER    UNTERSICHT  */
SET POSITION(7)=POINT2(14. ,12.5); /* FUER     ISOMETRIE  */
```

```
   EDIT HEIGHT(2 MM) FOR(BEZ_P);
   DCL  DIN6    ENTRY(PTR /*REGENT-BASIS           */,
                      PTR /*GIPSY-OBJEKT POINT     */,
                      PTR /*GIPSY-OBJEKT COLLECTION*/,
                  (*) PTR /*GIPSY-OBJEKT (*)POINT2 */,
                      PTR /*GIPSY-OBJEKT COLLECTION*/) EXTERNAL;
   /* BEI INTERNER PROCEDURE DIN6 WUERDE DIE DEKLARATION GANZ
      UND IM AUFRUF DAS "QQ" ENTFALLEN,ALSO:
      CALL DIN6(BEZ_P,A,POSITION,RESULT)                          */
   CALL DIN6(QQ /*REGENT-BASIS*/,BEZ_P,A,POSITION(*),RESULT);
   PLOT(RESULT);

    EMPTY(A      );
    EMPTY(RESULT);
   END GIPSY;
   FINISH;
   END PROGRAM;
```

Es folgen die externen Unterprogramme EXAMPLE und DIN6:

```
*PROCESS NS ;
   EXAMPLE:PROC(OBJEKT,BEZ_P) REGENT(SUB=GIPSY);
   DCL TEIL(2) SPACE(7);
   DCL (VORNE,HINTEN,LINKS,RECHTS,UNTEN,
        OBEN_L,OBEN_R,NEIGE,MITTE)PLANE;
   DCL OBJEKT COLLECTION PARAMETER;
   DCL BEZ_P  POINT PARAMETER;
   DCL P(0:3) POINT;
   DCL PHI BIN FLOAT(21);
   DCL ANG BIN FLOAT(21) INIT(45.);
   DCL N POINT;
   CHANGE UNIT LENGTH( CM );
   PHI= ANG*ACOS(0.E0)/90.E0;
   SET N   =POINT(-SIN(PHI),0,-COS(PHI));
   SET P(0)=POINT(0,0,0);
   SET P(1)=POINT(2,0,0);
   SET P(2)=POINT(0,2.5,0);
   SET P(3)=POINT(0,0,3);
   SET MITTE =PLANE(POINT(0,1.5,0)          ,POINT(0,-1,0)   );
   SET UNTEN =PLANE(P(0)                    ,P(3)            );
   SET LINKS =PLANE(P(0)                    ,P(2)            );
   SET HINTEN=PLANE(P(0)                    ,P(1)            );
   SET OBEN_L=PLANE(P(3)                    ,POINT(0,0,-1)   );
   SET OBEN_R=PLANE(POINT(0,0,0.5)          ,POINT(0,0,-1)   );
   SET VORNE =PLANE(P(1)                    ,POINT(-1,0,0)   );
   SET NEIGE =PLANE(POINT(2,0,2)            ,N               );
   SET RECHTS=PLANE(P(2)                    ,POINT(0,-2,0)   );
   BUILD TEIL(1)=SPACE(LINKS+OBEN_L+NEIGE+UNTEN+HINTEN+VORNE
                      +MITTE);
   BUILD TEIL(2)=SPACE(RECHTS+OBEN_R+UNTEN+HINTEN+VORNE
                      -MITTE);
   SET BEZ_P=POINT(-13,+17,-90);
  /*DIESER BEZUGS-PUNKT IST WILLKUERLICH GEWAEHLT*/
   SET OBJEKT=MOVETO(COLLECTION(BODY(TEIL(1)+TEIL(2))),BEZ_P);
   EMPTY(TEIL(1));
   EMPTY(TEIL(2));
   END EXAMPLE;
```

```
*PROCESS NS ;
   DIN6:PROC(BEZ_P,AIN,POSITION,RESULT) REGENT(SUB=GIPSY);
   DCL BEZ_P POINT PARAMETER;
   DCL (AIN,RESULT) COLLECTION PARAMETER;
   DCL A COLLECTION;
   DCL POSITION(7) POINT2 PARAMETER;
   DCL XPOS(7,2) DEC FLOAT(6);
   DCL J,I;
   DO J=1 TO 7;DO I=1 TO 2;
   XPOS(J,I)=COORD(POSITION(J),I);
   END; END;
   SET A=SHIFT(AIN,-COORD(BEZ_P,1),-COORD(BEZ_P,2),-COORD(BEZ_P,3));
 SET BEZ_P=SHIFT(BEZ_P,-COORD(BEZ_P,1),-COORD(BEZ_P,2),-COORD(BEZ_P,3));
   CHANGE PROJECTION PI_P(0,0,0);
   /* DRAUFSICHT       */
        CHANGE PROJECTION PI_N(0,0,1);
        CHANGE PROJECTION PARALLEL(0,0,1);
        CHANGE PROJECTION ORIGIN(XPOS(1,1),XPOS(1,2));
        SET RESULT=COLLECTION(RESULT,REDUCE(BEZ_P),
      ROTATE2(REDUCE(A),-90. /* GRAD */,POINT2(XPOS(1,1),XPOS(1,2))));
   /*  RUECKANSICHT          */
        CHANGE PROJECTION PI_N(-1,0,0);
        CHANGE PROJECTION PARALLEL(-1,0,0);
        CHANGE PROJECTION ORIGIN(XPOS(2,1),XPOS(2,2));
        SET RESULT=COLLECTION(RESULT,REDUCE(BEZ_P),REDUCE(A));
   /*  RECHTSANSICHT        */
        CHANGE PROJECTION PI_N(0,1,0);
        CHANGE PROJECTION PARALLEL(0,1,0);
        CHANGE PROJECTION ORIGIN(XPOS(3,1),XPOS(3,2));
        SET RESULT=COLLECTION(RESULT,REDUCE(BEZ_P),REDUCE(A));
   /*  VORDERANSICHT         */
        CHANGE PROJECTION PI_N(1,0,0);
        CHANGE PROJECTION PARALLEL(1,0,0);
        CHANGE PROJECTION ORIGIN(XPOS(4,1),XPOS(4,2));
        SET RESULT=COLLECTION(RESULT,REDUCE(BEZ_P),REDUCE(A));
   /*  LINKSANSICHT        */
        CHANGE PROJECTION PI_N(0,-1,0);
        CHANGE PROJECTION PARALLEL(0,-1,0);
        CHANGE PROJECTION ORIGIN(XPOS(5,1),XPOS(5,2));
        SET RESULT=COLLECTION(RESULT,REDUCE(BEZ_P),REDUCE(A));
   /*  UNTERSICHT            */
        CHANGE PROJECTION PI_N(0,0,1);
        CHANGE PROJECTION PARALLEL(0,0,-1);
        CHANGE PROJECTION ORIGIN(XPOS(6,1),XPOS(6,2));
        SET RESULT=COLLECTION(RESULT,REDUCE(BEZ_P),
      ROTATE2(REDUCE(A),-90. /* GRAD */,POINT2(XPOS(6,1),XPOS(6,2))));
   /*  ISOMETRIE             */
        CHANGE PROJECTION PI_N(1,1,1);
        CHANGE PROJECTION PARALLEL(1,1,1);
        CHANGE PROJECTION ORIGIN(XPOS(7,1),XPOS(7,2));
        SET RESULT=COLLECTION(RESULT,REDUCE(BEZ_P),
              SCALE2(REDUCE(A),SQRT(1.5E0),SQRT(1.5E0),
                                  POINT2(XPOS(7,1),XPOS(7,2))));
    EMPTY(A);
   END DIN6;
```

8.5 Kurven mit Achsen

Die Darstellung von Diagrammen mit GIPSY wird in Abb.8.5 gezeigt. Dazu werden die Achsenfunktionen (AXIS2, XAXIS und YAXIS) zur Beschreibung von Abszisse und Ordinate, sowie CURVE2 zur Zuweisung und Skalierung der Kurve benötigt.

Im 1.Bild werden die Sinus- und Cosinus-Funktionen zwischen 0 und 360 Grad dargestellt. Während die Abszisse explizit beschrieben wird (mit AXIS2), wird für die Ordinate die Kurzform YAXIS benutzt. Danach erfolgt die Generierung der Kurvenzüge und der Plotbefehl.

Die Bilder 2 und 3 zeigen eine beliebige Kurve in einem orthogonalen und einem nicht orthogonalen Achsensystem.

Zwei versetzte e-Funktionen werden in Bild 4 dargestellt. Die Abszisse wird mit der Kurzform XAXIS generiert. Die Ordinate hat eine logarithmische Unterteilung.

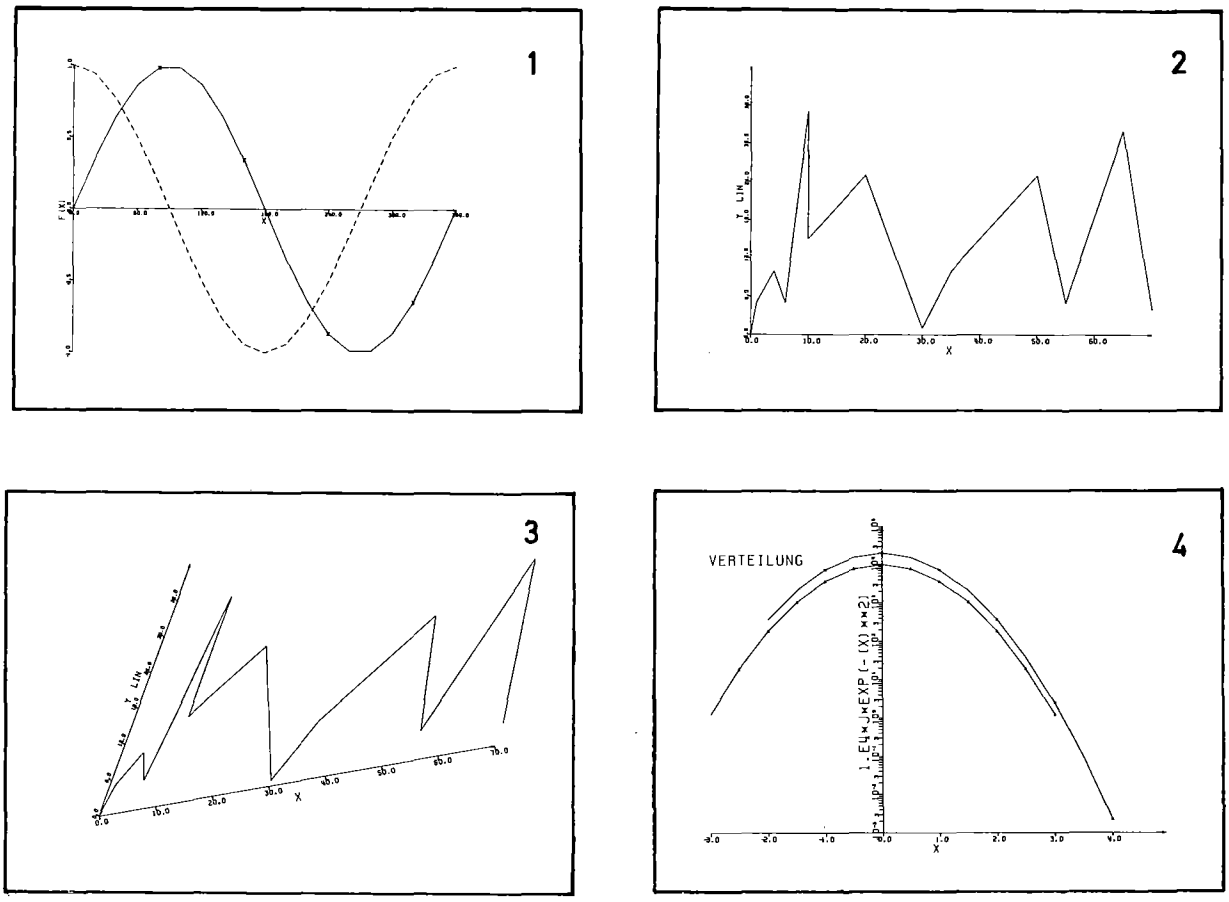


Abb. 8.5: Kurven mit Achsen.

```
ACHSEN:PROC OPTIONS(MAIN)
    REGENT(INIT,NODA,MPOOL=300000,PLOT=STATOS);
  ENTER GIPSY;
  /************  BILD  1   ***************************************************/
    DCL DEGREE DEC FLOAT(6),
        XCOS(19) DEC FLOAT(6),
        YCOS(19) DEC FLOAT(6),
        YSIN(19) DEC FLOAT(6),
      DCL COSIN(2) POLY2(19),
        (XA,YA) AXIS2(6);
    OPEN PLOT DIN A(4) BROAD;
    DO I=1 TO 19;
      DEGREE=20.*(I-1);
      XCOS(I)=DEGREE;
      YCOS(I)=COSD(DEGREE);
      YSIN(I)=SIND(DEGREE);
    END;
    SET XA=AXIS2(POINT2(30 MM,0.105),0.,360.,200. MM,0.,200./6. MM,
    'X',3 MM,'LINRIGHT');
    SET YA=YAXIS(YCOS,37.5 MM,'F(X)',3 MM,'LINLEFT');
    SET COSIN(1) = CURVE2(XCOS,YCOS,XA,YA);
    SET COSIN(2) = CURVE2(XCOS,YSIN,XA,YA);
    EDIT LINETYPE (DASHED) OF (COSIN(1));
    EDIT LINETYPE(MARKED),SYMBOL(12),EVERY(4),HEIGHT(3MM) OF (COSIN(2));
    PLOT(XA,YA,COSIN(1),COSIN(2));
  /************  BILD  2 UND 3   ***************************************************/
    DCL ZAEHNUNG POLY2(13),
        LIN(2)   AXIS2(6);
    DCL WINKEL  DEC FLOAT(6);
    DCL XYCOR(13,2) DEC FLOAT(6) INIT
    (0.,0.,1.,5.,4.,10.,6.,5.,10.,35.,10.,15.,20.,25.,
     30.,1.,35.,10.,50.,25.,55.,5.,65.,32.,70.,4.);
    DO WINKEL=0.,10.;
      OPEN PLOT DIN A(4) BROAD;
      SET LIN(1)=AXIS2(POINT2(50. MM,40. MM),0.,70.,21. CM,WINKEL,
                       3 CM,'X',4 MM,'');
      SET LIN(2)=AXIS2(POINT2(0.050,0.040),0.,40.,14. CM,
           (90.-2*WINKEL),0.020,'Y  LIN',3 MM,'LINLEFT');
      SET ZAEHNUNG = CURVE2(XYCOR(*,1),XYCOR(*,2),LIN(1),LIN(2));
      PLOT(LIN(1),LIN(2),ZAEHNUNG);
    END;
  /************  BILD  4   ***************************************************/
    DCL XEF(13,2) DEC FLOAT(6),
        YEF(13,2) DEC FLOAT(6),
        XMINMAX(2) DEC FLOAT(6);
    DCL X_AX AXIS2(1),
        Y_AX AXIS2(19),
        EFUNC(2) POLY2(13);
    OPEN PLOT DIN A(4) BROAD;
    DO J=1,2;
      DO I=-6 TO 6;
        XEF(I+7,J)=(I/2.)+(J-1);
        YEF(I+7,J)=1.E4*J*EXP(-(I/2.+(J-1))**2);
      END;
    END;
    XMINMAX(1) = XEF(1,1);
    XMINMAX(2) = XEF(13,2);
    SET X_AX=XAXIS(XMINMAX,30 MM,'X',4 MM,'LIN');
    SET Y_AX=AXIS2(POINT2(0.120,0.030),1.E-3,1.E5,16 CM,90 DEG,
                   20 MM,'1.E4*J*EXP(-(X)**2)',5 MM,'LOGLEFT');

    DO J=1 TO 2;
      SET EFUNC(J) = CURVE2(XEF(*,J),YEF(*,J),X_AX,Y_AX);
      EDIT SYMBOL(J*2),EVERY(J),LINETYPE(MARKED),HEIGHT(0.002)
           FOR (EFUNC(J));
    END;
    PLOT(X_AX,Y_AX,EFUNC(1),EFUNC(2));
    PLOT (TEXT2(POINT2(0.030,0.170),0.,'VERTEILUNG'));
  /*******************************************************************************/
  END GIPSY;
  FINISH;
  END ACHSEN;
```

Programm zu Abb.8.5

8.6 Körperaufbau aus Raumelementen

Für den Körperaufbau aus Raumelementen sind drei Eigenschaften von GIPSY von besonderer Bedeutung.

(1) Raumelemente werden durch die BODY-Funktion zu einem zusammenhängenden Körper vereinigt ("verschweißt"), wenn sie mindestens eine gemeinsame Fläche haben ("virtuelle"Fläche). Diese Fläche wird bei der Raumelementbeschreibung (SPACE-Funktion) jeweils mit entgegengesetzter Normalenrichtung verwendet.

(2) Beim Körper- und Raumelementaufbau wird zwischen einer Zuweisung (SET) und einer Zuordnung (BUILD) unterschieden (siehe 6.2.4), der SET-Befehl bewirkt ein Kopieren von Werten, während der BUILD-Befehl eine Referenz auf ein Objekt erzeugt.

(3) Die GIPSY-Flächen, die der Raumelementdefinition dienen, sind mit Ausnahme der Kugel unendlich ausgedehnt. Diese Tatsache führt bei Verwendung von geschlossenen Flächen (Kugel, Zylinder, Kegel) unter Umständen zu unerwarteten (aber mathematisch richtigen) Effekten, die durch zusätzliche Maßnahmen beseitigt werden müssen.

Anhand der folgenden Beispiele werden diese GIPSY-Eigenschaften näher erläutert.

```
VIRT: PROC OPTIONS(MAIN) REGENT(NODA,MPOOL=300000,PLOT=STATOS);
 DCL PLOTF2 FILE;
 ENTER GIPSY FILE(PLOTF2);
  /*-------------------------------------------------------------*/
  /* DIE FOLGENDEN BEISPIELE SOLLEN DIE VERSCHIEDENEN MOEGLICHKEI-  */
  /* TEN DES KOERPERAUFBAUS ZEIGEN UND DEREN KONSEQUENZEN VERDEUT-  */
  /* LICHEN. Abb.8.6                                               */
  /*-------------------------------------------------------------*/
  DCL P0 POINT;
  DCL X1(2) BIN FLOAT(21) INIT(0.015,0.035);
  DCL X2(2) BIN FLOAT(21) INIT(0.015,0.05);
  DCL X3(2) BIN FLOAT(21) INIT(0.01,0.05);
  DCL (E1(3),E2(3),E3(3))PLANE;
  DCL (R1,R2) SPACE(6);
  DCL K BODY;

  SAVE;

  SET P0=POINT(0.,0.,0.);

  /*------ DEFINITION DER EBENEN, DIE DEN KOERPER BEGRENZEN --------*/

  SET E1(1)=PLANE(P0,POINT(1.,0.,0.));   /*Y-Z-EBENE*/
  SET E2(1)=PLANE(P0,POINT(0.,1.,0.));   /*X-Z-EBENE*/
  SET E3(1)=PLANE(P0,POINT(0.,0.,1.));   /*X-Y-EBENE*/
  DO I=2 TO 3;
   SET E1(I)=SHIFT(E1(1),X1(I-1),0.,0.);
   SET E2(I)=SHIFT(E2(1),0.,X2(I-1),0.);
   SET E3(I)=SHIFT(E3(1),0.,0.,X3(I-1));
  END;

  /*-------------------------------------------------------------*/
  /*   AUFBAU DER RAUMELEMENTE, DIE DEN KOERPER BILDEN.          */
  /*   E2(2) IST DIE VIRTUELLE FLAECHE, DIE BEIDEN TEILRAEUMEN   */
  /*   GEMEINSAM IST.                                            */
  /*   DURCH BUILD WIRD ERREICHT, DASS BEIDE TEILRAEUME          */
  /*   DIESE FLAECHE (ALS DATENELEMENT) REFERIEREN.              */
  /*-------------------------------------------------------------*/

  BUILD R1=SPACE(E1(1)-E1(3)+E2(1)-E2(2)+E3(1)-E3(3));
  BUILD R2=SPACE(E1(1)-E1(2)+E2(2)-E2(3)+E3(2)-E3(3));
```

```
 /*-------- AUFBAU DES KOERPERS ----------------------------------*/

 BUILD K=BODY(R1+R2);

 /* ANGABE DER DARSTELLUNGSPARAMETER */

 OPEN PLOT DIN A(4) BROAD;
 CHANGE PROJ PI_N (1.,0.,0.),
        PROJ PARA(1.,1.2,0.9),
        PROJ ORIGIN(+0.10,+0.10);
 CHANGE INCREMENT LINEAR(0.001);

 PLOT(K);/* BILD 1.1 */

/* VERSCHIEBEN DER VIRTUELLEN FLAECHE */

SET E2(2)=SHIFT(E2(2),0.,0.01,0.);

OPEN PLOT DIN A(4) BROAD;
PLOT(K);/* BILD 1.2 */

/* AUFBAU DES KOERPERS MIT DEM SET-BEFEHL */

SET K=BODY(R1+R2);
OPEN PLOT DIN A(4) BROAD;
PLOT(K);/* BILD 1.3 : ENTSPRICHT BILD 1.1 */

/* VERSCHIEBEN DER VIRTUELLEN FLAECHE  */

SET E2(2)=SHIFT(E2(2),0.,0.01,0.);
OPEN PLOT DIN A(4) BROAD;
PLOT(K);/* BILD 1.4 */

/* AUFBAU DER TEILRAEUME UND DES KOERPERS MIT SET */

SET R1=SPACE(E1(1)-E1(3)+E2(1)-E2(2)+E3(1)-E3(3));
SET R2=SPACE(E1(1)-E1(2)+E2(2)-E2(3)+E3(2)-E3(3));
SET K=BODY(R1+R2);
OPEN PLOT DIN A(4) BROAD;
PLOT(K);/* BILD 1.5 */

/* AUFBAU DER TEILRAEUME MIT SET UND DES KOERPERS MIT BUILD */

SET R1=SPACE(E1(1)-E1(3)+E2(1)-E2(2)+E3(1)-E3(3));
SET R2=SPACE(E1(1)-E1(2)+E2(2)-E2(3)+E3(2)-E3(3));
BUILD K=BODY(R1+R2);
OPEN PLOT DIN A(4) BROAD;
PLOT(K);/* BILD 1.6 */



 STATUS;
END GIPSY
MESSAGE DEBUG ACTIVE;
FINISH;
END VIRT;
```

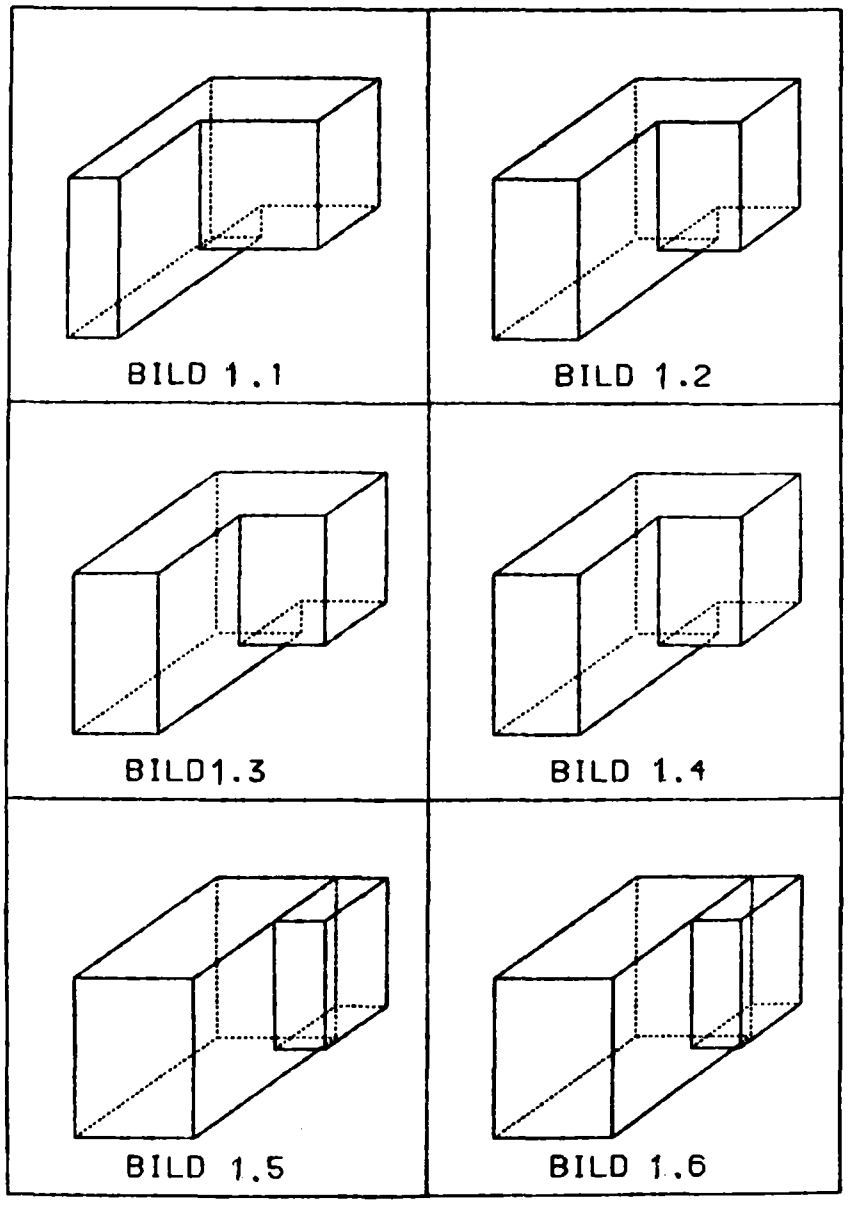


Abb. 8.6: Körperaufbau aus Raumelementen

```
BSP2: PROC OPTIONS(MAIN) REGENT(NODA,MPOOL=300000,PLOT=STATOS);
 DCL PLOTF FILE;
 ENTER GIPSY FILE(PLOTF);
  /*-------------------------------------------------------------*/
  /* DIESES BEISPIEL ZEIGT DEN AUFBAU EINES KOERPERS K AUS ZWEI  */
  /* TEILRAEUMEN, DIE BEIDE DURCH EINE GESCHLOSSENE FLAECHE, DEN */
  /* ZYLINDER CYL1, BEGRENZT WERDEN.Abb.8.7                      */
  /*-------------------------------------------------------------*/
  DCL(P0,P1)POINT;
  DCL X3(2) BIN FLOAT(21) INIT(0.05,0.07);
  DCL RADIUS BIN FLOAT(21) INIT(0.025);
  DCL BREITE BIN FLOAT(21) INIT(0.015);
  DCL (E1(3),E2(3),E3(3))PLANE;
  DCL E_HELP PLANE;
  DCL CYL1 CYLINDER;
  DCL (R1,R2) SPACE(10);
  DCL K BODY;

  SAVE;

  SET P0=POINT(0.,0.,0.);
  SET P1=POINT(0.06,0.06,0.);

  /* DEFINITION DER EBENEN, DIE DIE TEILRAEUME R1 UND R2 BEGRENZEN */

  SET E1(1)=PLANE(P0,POINT(1.,0.,0.));
  SET E2(1)=PLANE(P0,POINT(0.,1.,0.));
  SET E3(1)=PLANE(P0,POINT(0.,0.,1.));
  SET E1(2)=SHIFT(MOVETO(E1(1),P1),-BREITE/2.,0.,0.);
  SET E1(3)=SHIFT(E1(2),BREITE,0.,0.);
  SET E2(2)=MOVETO(E2(1),P1);
  DO I=2 TO 3;
   SET E3(I)=SHIFT(E3(1),0.,0.,X3(I-1));
  END;

  /* DEFINITION DES ZYLINDERS, DER DEN TEILRAEUM R2 BEGRENZT */

  SET CYL1=CYL(P1,SHIFT(P1,0.,0.,5.),RADIUS);

  /* AUFBAU DER BEIDEN TEILRAEUME */

  BUILD R1=SPACE(E1(2)-E1(3)+E2(1)-CYL1 +E3(1)-E3(2));
  BUILD R2=SPACE(CYL1+E3(1)-E3(3));

  /* AUFBAU DES KOERPERS K AUS DEN TEILRAEUMEN R1 UND R2 */

  BUILD K=BODY(R1+R2);
```

```
  /* ANGABE DER DARSTELLUNGSPARAMETER */

  OPEN PLOT DIN A(4) BROAD;
  CHANGE PROJ PI_N (1.,0.,0.),
         PROJ PARA(1.,1.2,0.8),
         PROJ ORIGIN(+0.10,+0.10);
  CHANGE INCREMENT LINEAR(0.001);

  /* BILD 2.1 ZEIGT, DASS DER TEILRAUM R1 DURCH DIE GESCHLOSSENE  */
  /* FLAECHE CYL1 NICHT AUF EINE ENDLICHE AUSDEHNUNG BEGRENZT     */
  /* WIRD. DER TEILRAUM R1 ERSTRECKT SICH BEIDERSEITS DES         */
  /* ZYLINDERS, WAS IM BILD AUF DER RECHTEN SEITE DES ZYLINDERS   */
  /* DURCH DIE SCHNITTLINIEN SICHTBAR WIRD.                       */

  PLOT(K);/*BILD 2.1  */

  /* IM FOLGENDEN WIRD EINE HILFSFLAECHE E_HELP EINGEFUEHRT,      */
  /* DURCH DIE DER TEILRAUM R1 BEGRENZT WIRD UND SOMIT DIE        */
  /* UNERWUENSCHTEN SCHNITTLINIEN GEMAESS BILD 2.1 ENTFALLEN      */

  SET E_HELP=MOVETO(E2(1),P1);
  BUILD R1=SPACE(E1(2)-E1(3)+E2(1)-CYL1 +E3(1)-E3(2)-E_HELP);
  BUILD R2=SPACE(CYL1+E3(1)-E3(3));
  BUILD K=BODY(R1+R2);
  OPEN PLOT DIN A(4) BROAD;
  PLOT(K);/*BILD 2.2  */
  STATUS;
 END GIPSY;
 MESSAGE DEBUG ACTIVE;
 FINISH;
END BSP2;
```

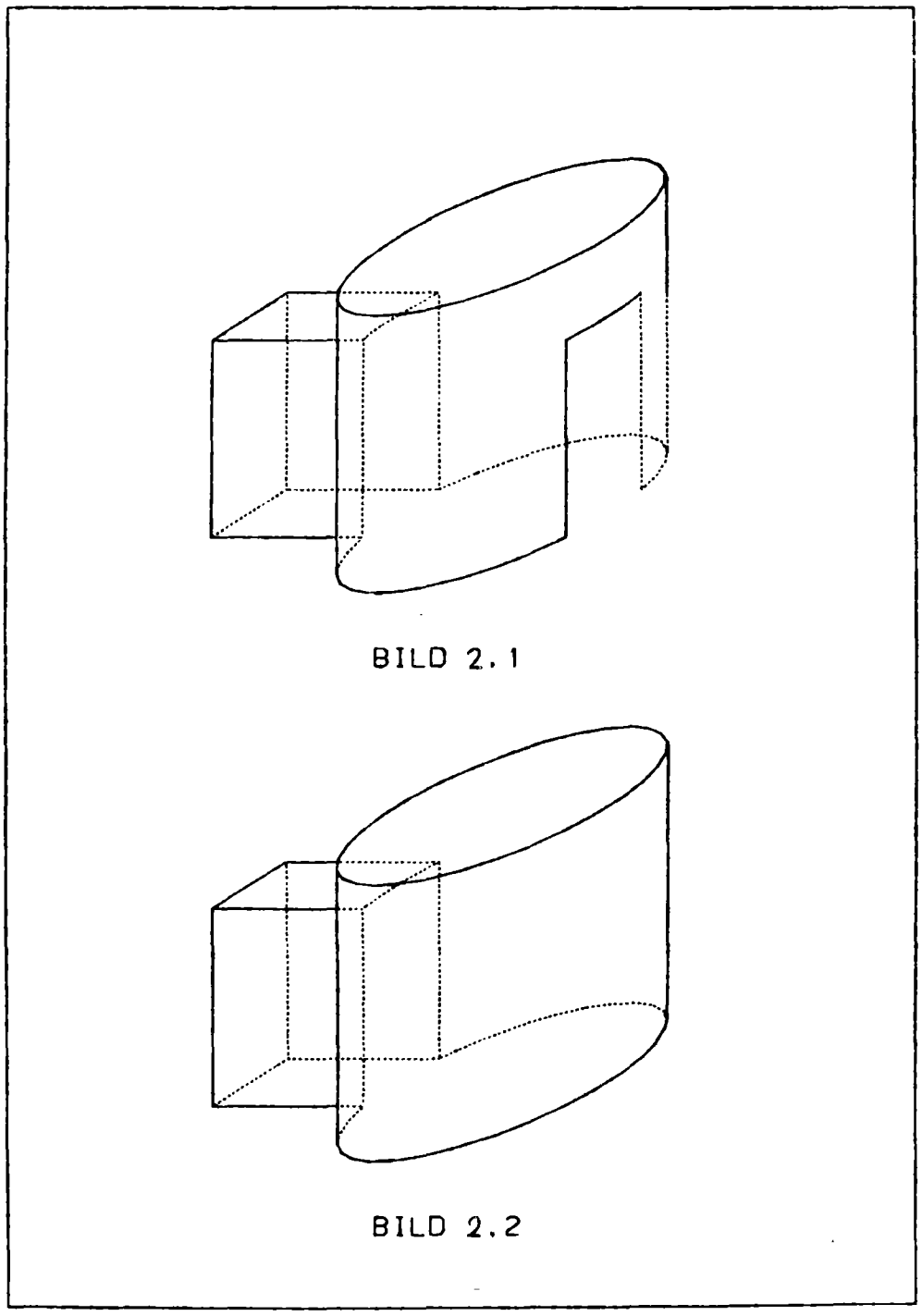


Abb. 8.7: Zylinder und Quader

9. Alphabetischer Handbuchteil

Das Kapitel 9 enthält alle GIPSY-Funktionen, Anweisungen und Objekte in alphabetischer Reihenfolge. Die Syntax und die Bedeutung wird jeweils angegeben. Beispiele verdeutlichen die Anwendung der GIPSY-Funktionen und GIPSY-Anweisungen.

Beispiele

1) BASED GIPSY-Objekte, nicht Member einer Struktur.

```
DCL A POINT BASED(PA);                    /* Deklarationen */
DCL B POLY2(12) BASED(PB);
DCL C (12,0:2) TEXT(100) BASED(PC);

ALLOC POINT A;                            /* Allokieren */
ALLOC POLY2 B;
ALLOC TEXT C;

SET A=POINT(x,y,z);                       /* Verwenden */
SET B=POLY2(....);
SET C(1,0)=TEXT(A,P2,P3,'xyz');
PLOT(B);

FREE POINT A;                             /* Freigeben */
FREE POLY2 B;
FREE TEXT C;
```

2) BASED Struktur mit GIPSY-Objekten

```
DCL 1  S BASED(PS),                              S BASED
     2  S1    POINT,
     2  S2    POLY(1000);
ALLOC  S;
```

Falsch wäre:

```
       ALLOC S1;                          /* S1 ist Substruktur */
       ALLOC POINT S1;
DCL A(N) POINT BASED;                     /* BASED nur bei festen
                                             Längen und
                                             Dimensionen */
```

ALLOCATE
ALLOC
Statement

Syntax

{ALLOC / ALLOCATE} type ident;

Erläuterung

type: GIPSY-Objekttyp, wie in DCL, also POINT,POINT2, TEXT, TEXT2, COLL, CYL, etc.

ident: Name eines GIPSY-Objektes mit der Storage Class BASED

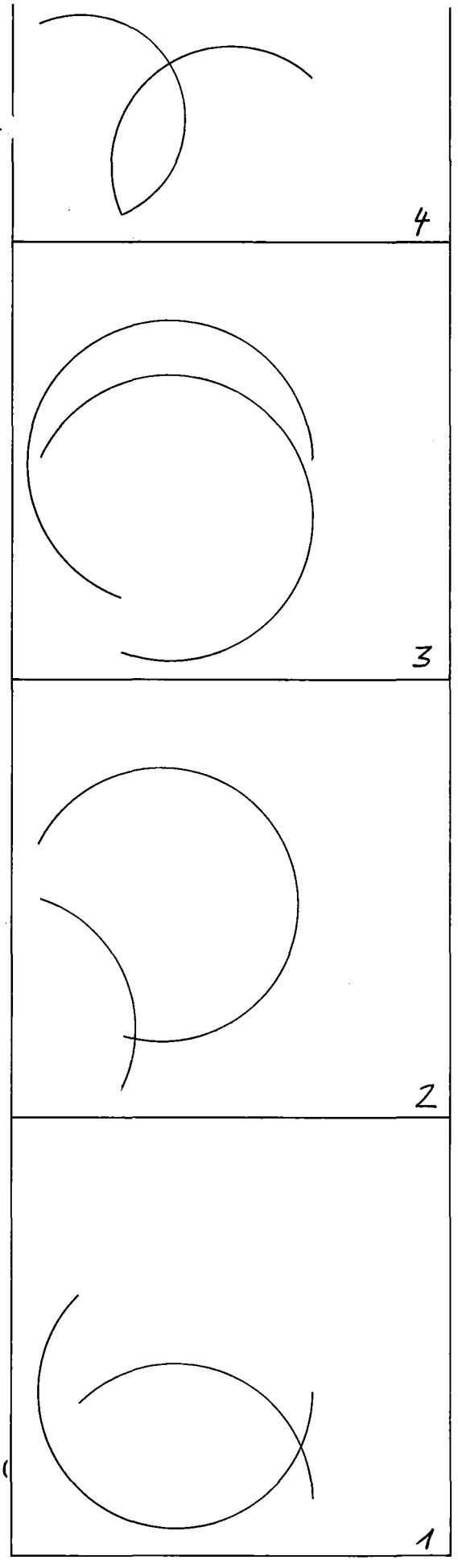
1) Siehe Kap. 6.9, DCL und FREE

2) Diese Form des ALLOC-Statements ist erforderlich für GIPSY-Objekte, die nicht Teil einer PL/1-Struktur sind und die Storage Class BASED haben

3) Strukturen, die PL/1-GIPSY-Objekte enthalten, werden mit normalen PL/1-ALLOC-Statements allokiert

4) Die Objekte werden mit einer besonderen FREE-Anwendung wieder freigegeben

```
// EXEC REGENT,PVOL=PNS001,PLIB='TSO253.TIPSY'
//P.SYSIN DD *
 BOGEN:PROC OPTIONS(MAIN) REGENT(INIT,NODA,PLOT=
  ENTER GIPSY;
   DO;          CHANGE UNITS LENGTH MM;
 /* BOGEN BEISPIELE. */
 /* 1. PUNKT,2WINKEL,RADIUS */

  DCL PU1   POINT2;
  DCL BO1    ARC2;
  DCL (WI1,WI2,RAD DEC FLOAT);
   OPEN PLOT SIZE(8 CM,8 CM);
   SET PU1 = POINT2(30.,10.);
   WI1 = 0.;
   WI2 = 135.;
   RAD =  25.;
   SET BO1 = ARC2(PU1,WI1,WI2,RAD);
   PRINT(BO1); PLOT(BO1);
   SET BO1 = ARC2(PU1,WI2,WI1,RAD);
   PLOT(SHIFT2(BO1,0.,20.));

   SET PU1 = POINT2(20.,5.);
   SET PU2 = POINT2(5.,40.);
   SET PU3 = POINT2(55.,30.);
 /* 2. PUNKT,PUNKT,RAD,GROSSER-,KLEINER BOGEN */

  DCL PU2     POINT2;
   OPEN PLOT SIZE(8 CM,8 CM) ;
   RAD = 25;
   SET BO1 = ARC2(PU1,PU2,RAD,'SMALL');
   PRINT(BO1); PLOT(BO1);
   SET BO1 = ARC2(PU1,PU2,RAD,'LARGE');
   PLOT(SHIFT2(BO1,0.,10.));

 /* 3. PUNKT,PUNKT,PUNKT */

  DCL PU3   POINT2;
   OPEN PLOT SIZE(8 CM,8 CM );
   SET BO1 = ARC2(PU1,PU3,PU2);
  /* KLEINER BOGEN PU1-PU2 WIRD
                   GEZEICHNET */
   PRINT(BO1); PLOT(BO1);
   SET BO1 = ARC2(PU1,PU2,PU3);
   PLOT(SHIFT2(BO1,0.,10.));

 /* 4. PUNKT,PUNKT,BOGENLAENGE. */

  DCL BOG DEC FLOAT;
   OPEN PLOT SIZE(8 CM,8 CM) ;
   BOG = 60.;
   SET BO1 = ARC2(PU1,PU2,BOG);
   PRINT(BO1); PLOT(BO1);
   SET BO1 = ARC2(PU3,PU1,BOG);
   PLOT(BO1);
  END;
    END GIPSY;
   MESSAGE ACTIVE DEBUG;
    FINISH;
 END BOGEN;
/*
//G.PLOTTAPE DD UNIT=T0800,VOL=SER=S253AR,LABEL=(
//
```



ARC2
Funktion
Deklaration

Syntax

```
{ ARC2(point2, avalue, avalue, radex)
{ ARC2(point2, point2, radex,{'LARGE'})
{                            {'SMALL'}
{ ARC2(point2, point2, point2)
{ ARC2(point2, point2, lvalue)
```

Erläuterung

Um einem Kreisbogen die bestimmenden Attribute zuzuweisen sind hier 4 Möglichkeiten gegeben:

1) point2 ist der Mittelpunkt des Kreisbogens, sein Radius ist durch radex gegeben, seine Länge durch die Winkelangaben. Der Kreisbogen wird von der ersten zur zweiten Winkelangabe im mathematischen positiven Sinn durchlaufen.

2) Der Kreisbogen ist definiert durch die Angabe von Anfangs- und Endpunkt, den Bogenradius sowie durch die Angabe, ob das kleinere Bogenstück ('SMALL') oder das größere ('LARGE') gemeint ist. Die Krümmung muß wieder, wenn man von Punkt 1 zu Punkt 2 geht,mathematisch positiv sein.
   Winkel werden in Grad (DEG) oder Radian (RAD) angegeben.

3) Es wird ein Kreisbogen erzeugt, ausgehend von Punkt 1 über Punkt 2 nach Punkt 3.

4) Ein Kreisbogen wird durch Anfangs- und Endpunkt und seine Länge bestimmt. Die Krümmung ist wieder positiv.

Deklaration

```
DCL [level] ident [dim] ARC2
    [storage class] ;
```

Siehe DCL.

Beispiele ARC

```
DCL P3 POINT;
DCL A1 ARC;
SET P3 = POINT(10,5,2);
SET A1 = ARC(POINT(0,0,0),POINT(1,3,0),P3);
PLOT(A1);

DCL AFELD(-1:10,5) ARC;
DCL AP(*,*) ARC PARAMETER;
DCL AB BASED(PA) ARC;
ALLOC ARC AB;
FREE ARC AB;
```

Beispiele AXIS2

```
DCL XAX AXIS2(4),
    YAX AXIS2(12),
    LAX AXIS2(5),
    (P0,P1) POINT2 ;
    CHANGE UNITS LENGTH(MM);
SET P0 = POINT2(50.,30.)
    P1 = POINT2(25.,30.)
SET XAX = AXIS2(P0,0,120,20CM,0DEG,2CM,'MSEC',5MM,);
SET YAX = AXIS2(P0,0,120,24CM,90DEG,2CM,'PRESSURE BAR',
                                              5MM,'LEFT');
SET LAX = AXIS2(P1, 1.E4,1.E8,24CM,90DEG,6CM,'HERTZ',
                                          3MM,'LOGEXLEFT');
SET LAX = AXIS2(P1,1.E4,1.E6,18CM,0DEG,,,'LOG');
SET XAX=AXIS2(P0,0.,50.,20CM,0DEG);
DCL  1  S PARAMETER,
        2 XAX1 AXIS2(10),
        2 YAX1(*)AXIS2(20);
```

ARC
Funktion
Deklaration

Syntax

ARC(P1, P2, P3)

Erläuterung

P1, P2, P3: g_ex mit Objekttyp POINT

Es wird ein Bogen von Punkt P1 über Punkt P2 nach Punkt P3 erzeugt.

Deklaration

DCL [level] ident [dim] ARC [storage class] ;

Siehe DCL.

AXIS2
Funktion
Deklaration

Syntax

AXIS2(point, x_{min}, x_{max}, länge, [winkel] , [abstand] , [text] , [texthöhe] , [options])

Deklaration

DCL [level] ident [dim] AXIS2(maxchar) [storage class] ;

maxchar: PL1_ex, maximale Anzahl von Zeichen in der Achsenbeschriftung.

Siehe DCL

Erläuterung siehe nächste Seite

Erläuterung AXIS2

point2	:	g_ex	Punkt, an dem die Achse beginnt
x_{min}	:	PL1_ex	minimaler Koordinatenwert
x_{max}	:	PL1_ex	maximaler Koordinatenwert
laenge	:	lvalue	Achsenlänge
winkel	:	avalue	Winkel der Achse
abstand	:	lvalue	Abstand der Skalenstriche
text	:	stringexpr	Textstring zur Achsenbeschriftung. Seine Länge wird in der Deklaration von AXIS 2 festgelegt.
texthöhe	:	lvalue	Texthöhe
options	:	['LIN' | 'LOG'] || ['EX'] || ['RIGHT' | 'LEFT']	Options besteht aus einem Textstring für Angaben zu den Achsen.
		'LIN'	Achsenteilung linear
		'LOG'	Achsenteilung logarithmisch
		'EX'	Der minimale Koordinatenwert liegt exakt auf dem Anfangspunkt der Achse, der maximale Wert auf dem Endpunkt. Dies kann zu "krummen" Werten an den Skalenstrichen führen. Wird 'EX' weggelassen, werden an den Skalenstrichen glatte Werte erzeugt.
		'RIGHT'	Achsenbeschriftung rechts der Achse
		'LEFT'	Achsenbeschriftung links der Achse

Das Format liegt zwischen F(3,1) und F(7,4). Für Werte ≥ 0 und < 100 wird auch das Format F(4,1) und für Werte ≥ 0 und < 1000 auch F(5,1) verwendet. Alle Nullen hinter dem Komma außer der ersten und führende Nullen werden weggelassen. Ist, gegeben durch Skalenabstand und Texthöhe, nicht genug Platz für eine Beschriftung an jedem Skalenstrich so wird nur an jeden n-ten Skalenstrich eine Beschriftung geschrieben.

Bemerkungen

1) Standardwerte für AXIS2 gibt es für den Abstand der Skalenstriche: 2cm, den Text: blank, die Texthöhe: 5 mm, den Winkel: 0 und die Options: LINRIGHT. Fehlende Optionsangaben werden durch Komma gekennzeichnet. Leerkommata am Ende können wegbleiben.

2) Der Abstand des Textes von der Achse liegt fest. Das Format der Skalenbeschriftung ergibt sich aus dem Abstand der Skalenstriche und der Texthöhe. Siehe Abb.9.1.

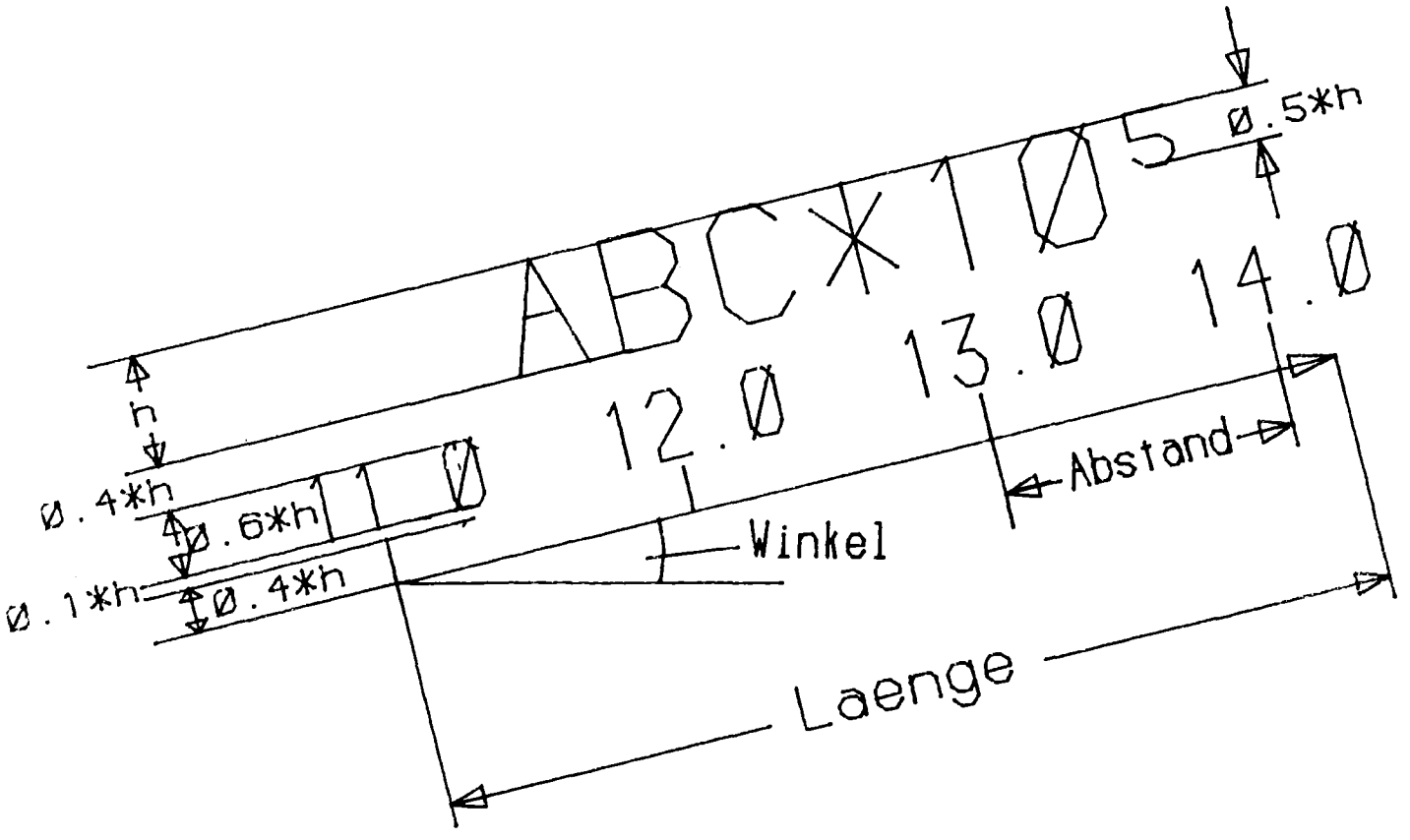


Abb.9.1: Achse mit Längenangaben zu der Skalenbeschriftung

Beispiele avalue

gültig:

```
45.
90.00 DEGREE
WINKEL*(J-1)  RAD
```

sofern das Ergebnis von WINKEL*(J-1) eine Zahl ist

ungültig:

```
DCL X  PTR;
X
X RAD
0.5 * X RAD
0.5 * X
```

Alle diese Formen führen zu Fehlermeldungen des PL/1-Compilers von der Art "Operand X illegal in element expression" oder "conflicting attributes of argument...to entry..."

Beispiel BALL

```
DCL KUGEL BALL;
DCL P1  POINT;
SET P1 = POINT(1.,1.,1.);
SET KUGEL=BALL(P1,5CM);
```

avalue

Syntax

PL1_ex a-unit

a-unit: DEGREE | GRAD | RADIAN

Erläuterung

1) Dies ist eine Winkelangabe.
2) PL1_ex muß ein in PL/1 gültiger Ausdruck sein, der in eine DECIMAL FLOAT(6) Darstellung ungewandelt werden kann.
3) Fehlt a-unit, so wird die Standardeinheit für Winkel benutzt. Siehe CHANGE UNIT ANGLE.

BALL
Funktion
Deklaration

Syntax

BALL(mittelpunkt, radius)

Erläuterung

mittelpunkt: g_ex vom Typ POINT,Kugelmittelpunkt
radius: PL1_ex, Kugelradius

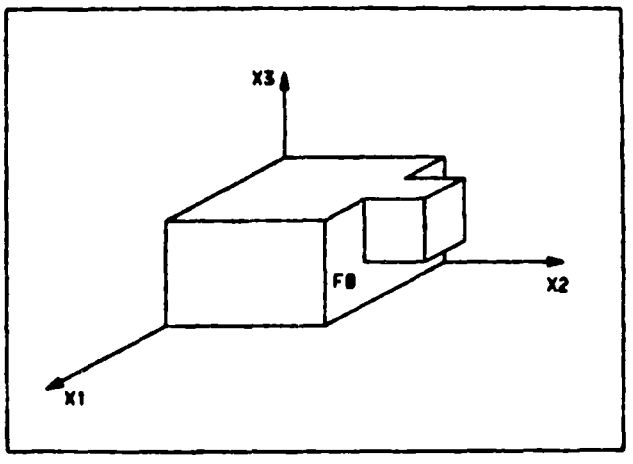
Eine Kugel wird durch ihren Muttelpunkt und ihren Radius beschrieben.

Deklaration

DCL [level] ident [dim] BALL
[storage class] ;

Siehe DCL.

Beispiel




```
DCL(SPOO,SPO1)SPACE(...);
DCL SP1 SPACE(...);
DCL SP2 SPACE(...);
DCL Fo PLANE;
BUILD SP1 = SPACE(SPOO + Fo);
BUILD SP2 = SPACE(SPO1 - Fo);

DCL BODY1 BODY;

BUILD BODY1= BODY(SP1 + SP2);
PLOT(BODY1);
```

BODY
Funktion
Deklaration

Syntax

BODY([+] arg - g_ex
[+ arg - g_ex]$_{o}^{59}$)

arg - g_ex = raumelement - g_ex | body - g_ex

Erläuterung

Die BODY-Funktion dient dazu, aus Raumelementen (SPACE) komplexere Körper zu bilden, indem je zwei Raumelemente die gleiche Fläche (allerdings mit entgegengesetztem Vorzeichen, virtuelle Fläche) referieren. Diese Funktion gestattet es, insbesondere auch konkave Körper zu erzeugen.

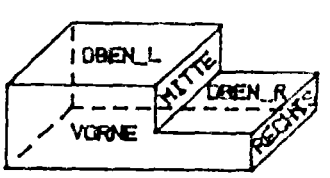
Deklaration

DCL [level] ident [dim] BODY
[storage class] ;

Siehe DCL.

Beispiel

```
DCL (VORNE,HINTEN,LINKS,RECHTS,MITTE,UNTEN)PLANE;
DCL (OBEN_RECHTS, OBEN_LINKS) PLANE;

/*Wir ersparen es uns, die Zuweisungen zu diesen Ebenen
ausführlich hinzuschreiben. Diese acht Ebenen sind in der
Skizze unten zu erkennen (mit Ausnahme der verdeckten Ebenen
HINTEN und LINKS)*/

DCL QUADER(2) SPACE(6);

/* Quader werden vereinbart. Jeder wird durch maximal 6 Flächen
   begrenzt */

DCL OBJEKT BODY;
DCL A COLLECTION;
SET A = COLL(VORNE,HINTEN,UNTEN);
BUILD QUADER(1)=SPACE(A+LINKS+MITTE);
BUILD QUADER(2)=SPACE(A+RECHTS-MITTE);
BUILD OBJEKT=BODY(QUADER(1)+QUADER(2));

/* Nun kann OBJEKT gezeichnet werden */
PLOT(OBJEKT);
/* Da MITTE in beiden QUADERn vorkommt, wird erkannt, daß
beide Quader einen Körper bilden und nicht zwei QUADER dicht
nebeneinander stehen */
/* Da der Körperaufbau über BUILD vorgenommen wurde, wirken
sich nachträgliche Änderungen der Fläche aus. Eine Verschie-
bung der Stufe um 1 cm, 2 cm etc. erfordert also nur:*/

DO J= 1 TO N;
SET MITTE=SHIFT(MITTE,1 CM,0,0);PLOT(OBJEKT);
END;

/* Eine Wiederholung der obigen Anweisungen zum Körperauf-
bau erübrigt sich */
```



BUILD
Statement

Syntax

```
{ BUILD space-g_obj=SPACE(.....);  }
{ BUILD body-g_obj=BODY(.....);    }
```

Erläuterung

1) Es wird eine Datenstruktur aufgebaut, welche Referenzen auf die in der Argumentliste der SPACE= bzw. BODY-Funktion angegebenen Objekte enthält.

2) Hinsichtlich der Lebensdauer der Objekte, die als Argumente auftreten, ist Kap.4.7 zu beachten.

CARDCOL
Funktion

Syntax

```
N = CARDCOL(c) ;
```

Erläuterung

CARDCOL ist eine GIPSY-Funktion, die,auf eine COLLECTION angewandt, die Anzahl deren Elemente liefert.(BIN FIXED(15)-Wert).

N - BIN FIXED(15)
c - g_obj von Typ COLLECTION

CARDCOL kann in normalen PL/1-Anweisungen auftreten. Als Parameter sind nur namentlich deklarierte Kollektionen erlaubt (g_obj), nicht jedoch Builtin-Funktionen (g_ex, z.B. COLL oder INTER).

Beispiele siehe nächste Seite

Beispiele

CARDCOL

```
DCL A POINT, B CIRCLE, D PLANE, E BALL;
DCL C COLL;
DCL N BIN FIXED(15);
SET C=COLL(A,B,D,E);

N=CARDCOL(C);
IF N=0 THEN PUT SKIP
  LIST('COLLECTION C IS EMPTY');
ELSE PUT SKIP LIST
  ('COLLECTION C HAS', N ,
    'ELEMENTS');
```

Falsch wäre:

```
I=CARDCOL(COLL(A,B));
```

da keine Builtin-Funktionen erlaubt sind.

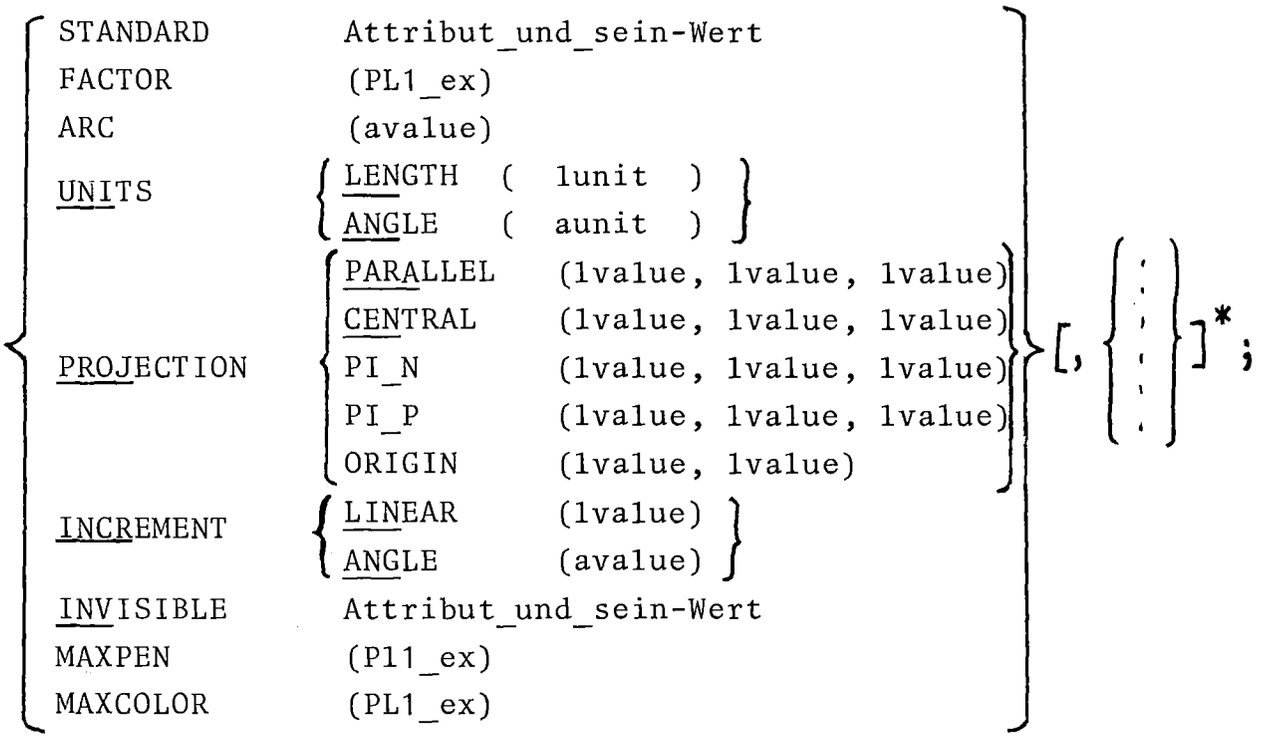
Beispiele

CHANGE

```
CHANGE UNITS LENGTH(MM),
       PROJECTION PARALLEL(0,0,1),
       PROJECTION PI_N(0,0,1),
       INVISIBLE LINETYPE(DASHED),
       STANDARD PEN(5),
       INVISIBLE PEN(4),
       MAXPEN(8),
       ARC(5GRAD);

CHANGE UNITS ANGLE(RADIAN);

CHANGE INCREMENT LINEAR(2CM),
       INCREMENT ANGLE(10DEG);
```

CHANGE
Statement

Syntax



Erläuterung

CHANGE ändert die Systemvariablen für die

- Standardeinheiten (Längen-, Winkelmaß)
- Projektionsart (parallel, zentral,...)
- Inkremente bei Kurvenberechnung
- Standardattribute (Linientyp, Farbe, Texthöhe,...)
- Skalierung (FACTOR)
- Umwandlung von Kreisen und Bögen in Polygone.

STANDARD: siehe CHANGE STANDARD
UNITS: " " UNITS
PROJECTION: " " PROJECTION
INVISIBLE: " " INVISIBLE
ARC,FACTOR,INCREMENT,MAXPEN,MAXCOLOR: siehe CHANGE ARC

Erläuterung

1) ARC setzt den Winkel für das Erzeugen eines gleichseitigen Polygons aus einem Kreis. Diese Operation wird beim Zeichnen und bei der Operation POLY/POLY2 (Kreis und Bogen) durchgeführt.

2) FACTOR setzt den Wert, den 1 m Länge in Plotter-Koordinaten hat. (Für Inch-Plotter 100/2.54 für cm-Plotter 100.0).

3) INCREMENT setzt die Schrittweite für die Körperanalyse für gerade Kanten (LINEAR) oder für Winkel (ANGLE, z.B.für Kugel, Zylinder) auf die angegebenen Werte.
Durch die Wahl der Inkremente wird die Effektivität bei der Körperanalyse stark beeinflußt. Standardwerte sind 8 mm und 5.7 Grad.

4) Die Inkremente sollen sich an der kleinsten (kürzesten) aufzulösenden Körperkante orientieren. Nur Konturen und verdeckte Kanten, die größer als das durch die Inkremente gesetzte Raster sind, werden bei der Körperanalyse erfaßt. Sehr kleine Inkremente dagegen vergrößern den Rechenzeitbedarf erheblich. Für das Längeninkrement kann als Faustregel ein Wert von 1% bis 2% der längsten Körperabmessung gewählt werden. Für Testläufe empfiehlt es sich, größere Inkremente zu wählen.

4) Durch CHANGE MAXPEN (PL1_ex_p) wird die höchstzulässige Stiftnummer festgelegt. Durch CHANGE MAXCOLOR(PL1_ex_c)die maximal zulässige Farbnummer. Höhere Werte werden gemäß

$$n' = \mathrm{mod}(n-1,nmax)+1$$

umgerechnet. Dadurch wird verhindert, daß eine höhere Stiftnummer bzw. Farbe als für das betreffende Gerät zulässig verwendet wird. Standardwerte: MAXPEN(4) und MAXCOLOR(64).

5) Alle Werte können durch RESET wieder auf die System-Standardwerte zurückgesetzt werden.

CHANGE
ARC,FACTOR,
INCREMENT,
MAXPEN,
MAXCOLOR
Statement

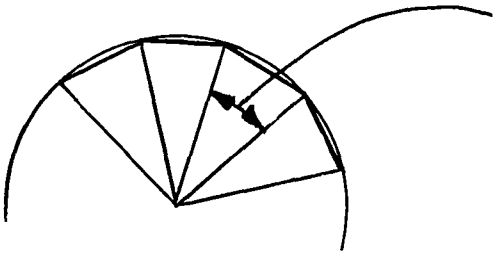
Syntax

CHANGE

```
{ ARC(avalue)
  FACTOR(p11_ex)
  INCREMENT  { LINEAR(1value) }
             { ANGLE(avalue)  }
  MAXPEN(p11_ex_p)
  MAXCOLOR(p11_ex_c)
}
```

Beispiele

```
CHANGE ARC (30GRAD);
```



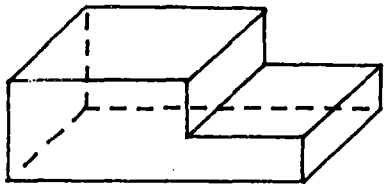
30^{o}, pro Vollkreis werden 13 Punkte erzeugt.

```
CHANGE ARC(1 GRAD); 361 Punkte pro Vollkreis
CHANGE FACTOR(100/2.54*F); Vergrößerung um Faktor F auf Inch-
                                                    Plotter
CHANGE INCREMENT LINEAR(2MM),INCREMENT ANGLE(5 DEGREES);
CHANGE MAXPEN(3),
       MAXCOLOR(128),
       ARC(10GRAD);
```

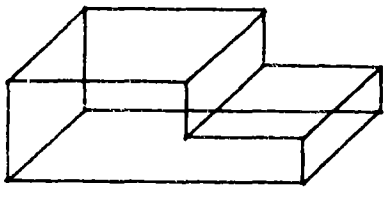
CHANGE
INVISIBLE

Beispiel

```
DCL OBJEKT BODY;
/* Wir nehmen an,folgender Körper sei durch GIPSY-Anweisungen
der Variablen OBJEKT zugewiesen */
```



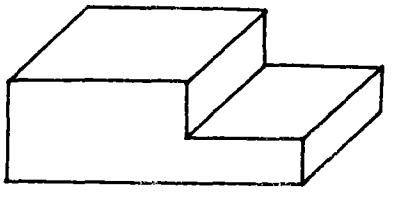
```
/* Obige Abbildung entspricht den Standardattributen:
VISIBLE-Linien sind SOLID; INVISIBLE-Linien DASHED*/
/* Diese Darstellung läßt sich ändern, wenn vor der Projektion
(in PLOT oder REDUCE) die Standard-Darstellungsattribute neu
definiert werden */
CHANGE INVISIBLE LINETYPE(SOLID);
```



```
CHANGE INVISIBLE LINETYPE(OMITTED);
```

andere Beispiele:

```
CHANGE INVISIBLE LINEWIDTH(0.5 MM),
       INVISIBLE PEN(3),
       INVISIBLE LENGTH(10 MM),
       INVISIBLE COLOR(2);
```



CHANGE
INVISIBLE
Statement

Syntax

CHANGE INVISIBLE standardattribut-i
[, INVISIBLE standardattribut-i]* ;
standardattribut-i::= LIKE STANDARD | standardattribut

Erläuterung

1) Mit dem CHANGE INVISIBLE statement wird die Darstellung derjenigen Linien festgelegt, die bei der Projektion von Körpern (BODY oder SPACE) in der REDUCE= oder PLOT=Operation als verdeckt erkannt werden.

2) Bezüglich standardattribut siehe CHANGE STANDARD -Statement

3) Standardwert der Darstellung unsichtbarer Kanten entspricht CHANGE STANDARD mit folgender Ausnahme
LINETYPE(DASHED)

4) Ein Fehler ist bei INVISIBLE die Angabe der Attribute
SLANT
WIDTH
DISTANCE
OPEN
CLOSED
Wenn für STANDARD und für INVISIBLE genau dieselben Attribute gelten, so sind verdeckte und sichtbare Kanten in der Darstellung nicht unterscheidbar. Die Ausführung der Projektion wird dadurch beschleunigt, da GIPSY dann die Sichtbarkeitsprüfung unterläßt. Dies wird am zweckmäßigsten erreicht durch CHANGE INVISIBLE LIKE STANDARD;

5) Siehe Kapitel 4.3.2 S.39 und 6.6.1 S. 93 bis 102.

Beispiel

Siehe Beispiel in Kap. 8.4.

```
CHANGE PROJECTION ORIGIN (20 CM, 5 CM);
```

Die Projektion des 3D-Nullpunktes in die Projektionsebene fällt mit dem Punkt (20 cm, 5 cm) des GIPSY-2D-Koordinatensystems zusammen.

```
CHANGE PROJECTION CENTRAL (
       COORD(P,1),
       COORD(P,2),
       COORD(P,3));
```

Der Punkt P wird zum Ursprung der Zentralprojektion.

```
CHANGE PROJEKTION PARALLEL(0,0,1);
```

Parallelprojektion senkrecht von "oben".

```
CHANGE PROJECTION PI_N(1,1,1),PROJECTION PI_P(A,B,C);
```

Die Projektionsebene steht senkrecht auf der Raumdiagonalen des ersten Raumoktanten. Die Projektionsebene geht durch den Punkt mit den Koordinaten (A,B,C)

CHANGE
PROJECTION
Statement

Syntax

CHANGE PROJECTION proj-option
[,PROJECTION proj-option]*;

$$\text{proj-option} = \left\{ \begin{array}{l} \text{centeroption} \\ \underline{\text{PI\_NORMAL}}\,(x_n, y_n, z_n) \\ \underline{\text{PI\_POINT}}\,(x_e, y_e, z_e) \\ \underline{\text{ORIGIN}}\ (x_o, y_o) \end{array} \right\}$$

$$\text{centeroption::=} \left\{ \begin{array}{l} \underline{\text{CEN}}\text{TRAL}\ (x_c,\ y_c,\ z_c) \\ \underline{\text{PARA}}\text{LLEL}\,(x_p,\ y_p,\ z_p) \end{array} \right\}$$

x_n bis z_p = lvalue

Erläuterung

1) Wird eine proj-option wiederholt, so gilt die jeweils letzte.

2) PI_NORMAL legt die Normale auf der Projektionsebene fest. PI_POINT ist ein beliebiger Punkt in der Projektionsebene.

3) Die Projektion des 3D-Koordinatenursprungs erhält im 2D-GIPSY-Raum die Koordinaten x_o, y_o.

4) CENTRAL ist das Projektionszentrum bei Zentralprojektion. PARALLEL ist ein Punkt, der bei Parallelprojektion zum 3D-Koordinatenursprung hin projiziert wird.

5) Hinsichtlich der Orientierung des Bildes gilt: GIPSY versucht "oben" (z positiv in 3D) "oben" auf der Zeichnung zu halten. Siehe Kap.5.2.3.

Erläuterung

1) Die Standardattribute erhalten neue Werte

2) Diese Standardattribute sind wirksam bei der Erzeugung von Punkten, Texten und Linienzügen aufgrund von Builtin-Funktion sowie bei der Erzeugung von sichtbaren Umrißlinien und Schnittlinien von Körpern bei der REDUCE-Operation (explizit oder implizit in der PLOT-Anweisung).

3) Wird ein Attribut wiederholt, so gilt die letzte Angabe.

4) Standardwerte sind

SYMBOL(1)	Punktsymbol
SLANT(0 GRAD)	Neigung von Text-Buchstaben
HEIGHT(5 MM)	Höhe von Punktsymbolen und Buchstaben
WIDTH(5 MM)	Breite von Textbuchstaben
EVERY(1)	Jeder Punkt eines Polygons wird markiert
DISTANCE(20 MM)	Skalenstrichabstand bei Achsen
OPEN	Polygonzüge offen
LINETYPE(SOLID)	Linienzüge durchgezogen
LENGTH(5 MM)	Strichabstand unterbrochener Linien
PEN(0)	Stiftauswahl nicht benutzt
COLOR(0)	Farbe irrelevant
LINEWIDTH(0.0)	Strichstärke nicht benutzt

5) Umriß- und Schnittlinien werden stets als OPEN erzeugt unabhängig von der Standardattributangabe.

6) Bei Ausgabe graphischer Objekte auf graphischen Geräten mit der PLOT-Anweisung kann es vorkommen,daß bestimmte Darstellungsattribute nicht realisiert werden können. Der Gerätetreiber wird dann in einer gerätespezifischen Weise darauf reagieren.

7) Siehe Kapitel 4.3.2 S.39 und 6.6.1 S.93 bis 102.

Syntax

```
CHANGE STANDARD standardattribut
     [ STANDARD standardattribut ]*  ;
```

Konflikte können auftreten, wenn z.B. gleichzeitig einerseits PEN, andererseits COLOR, LINEWIDTH angegeben ist. Hier gilt stets:

PEN =0, COLOR =0, LINEWIDTH =0:
Geräteabhängige Standardwerte gelten

PEN > 0, COLOR <=0, LINEWIDTH <=0:
Der PEN mit der angegebenen Nummer wird ausgewählt, falls das Gerät dies zuläßt.

PEN > 0, COLOR > 0 oder/und LINEWIDTH > 0:
Der angegebenen PEN-Nummer werden die angegebene Farbe und Strichstärke zugeordnet, der PEN wird ausgewählt.

PEN < =0, COLOR > 0 oder/und LINEWIDTH > 0:
Die angegebene Farbe und Strichstärke wird benutzt, falls das Gerät dies zuläßt.

Beispiele

```
CHANGE STANDARD HEIGHT(5 MM),STANDARD WIDTH (4MM),
       STANDARD SLANT(15 GRAD), STANDARD DISTANCE(1 CM);
CHANGE STANDARD LENGTH( X MM);
CHANGE STANDARD PEN(J);
```

Beispiele

Folgende Anweisungen haben dieselbe Wirkung:

```
CHANGE UNIT LENGTH M, UNIT ANGLE GRAD;
```

oder

```
CHANGE UNIT ANGLE(DEGREE), UNIT LENGTH M;
```

oder

```
CHANGE UNIT ANGLE DEGREE;
CHANGE UNIT LENGTH(M);
```

CHANGE
UNIT
statement

Syntax

```
CHANGE UNIT unitoption [,UNIT unitoption] ;
unitoption::= LENGTH ( lunit )
            | ANGLE ( aunit )
lunit::= MM | CM | M | INCH | FOOT | YARD | KM
aunit::= DEGREE | GRAD | RADIAN
```

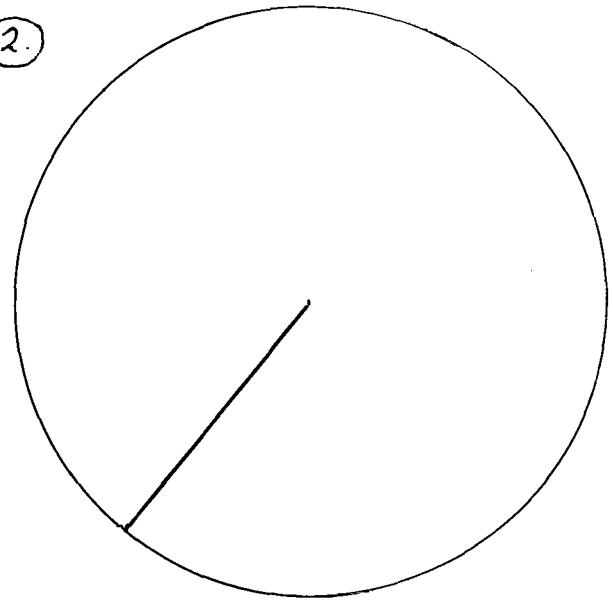
Erläuterung

1) Die Angabe der Einheit wird wirksam für alle folgenden Längenangaben (lvalue) und Winkelangaben (avalue), die nicht mit eigener Einheitenbezeichnung versehen sind.

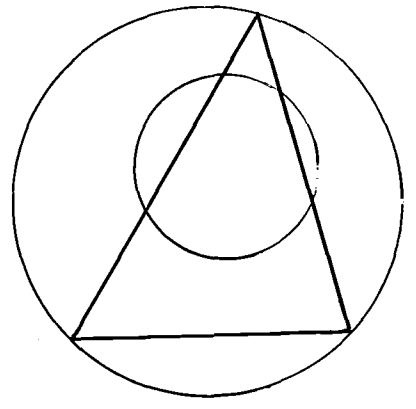
2) Defaultwerte sind

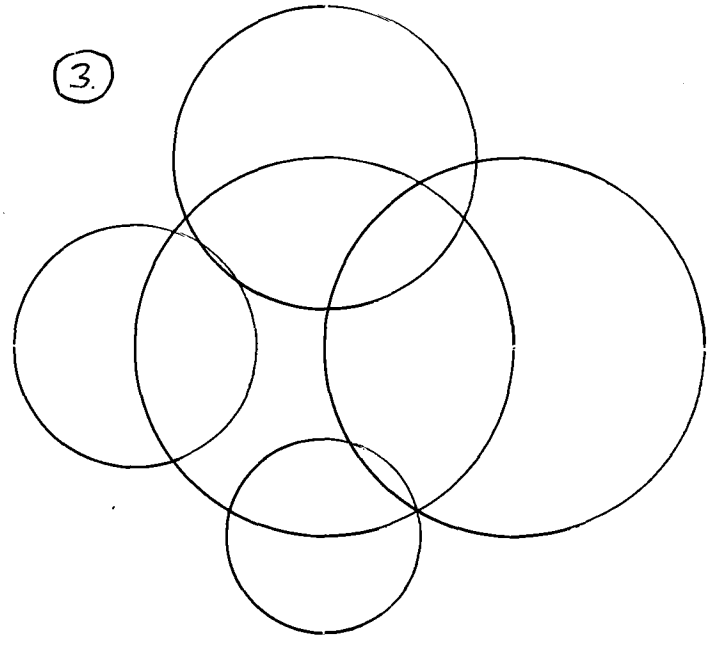
lunit: M
aunit: DEGREE

Beispiele




```
 CHANGE UNITS LENGTH MM;
 DCL PERPO(3) POINT2;
 DCL UMKREIS CIRCLE2;
 DCL INKREIS CIRCLE2;
 DCL LINIE POLY2(2);
 OPEN PLOT SIZE(17 CM,17 CM);
 DO;
  SET PERPO(1) = POINT2(25.,25.);
  SET PERPO(2) = POINT2(62.5,25.);
  SET PERPO(3) = POINT2(50.,67.5);
  SET LINIE = POLYGON( PERPO(1),PERPO(2) );
  PLOT(LINIE);
  SET LINIE = POLYGON( PERPO(3),PERPO(2) );
  PLOT(LINIE);
  SET LINIE = POLYGON( PERPO(3),PERPO(1) );
  PLOT(LINIE);
/** (1) 3 PUNKTE,IN_OUT                    **/
     SET UMKREIS =CIRCLE2(PERPO(1),PERPO(2),PERPO(3),'OUT');
   SET INKREIS =CIRCLE2(PERPO(1),PERPO(2),PERPO(3),'IN');

  PLOT(UMKREIS);
  PLOT(SHIFT2(INKREIS,0.,10.));
/** (2.) 2 PUNKTE: MITTE,UMFANG            **/
 DCL MIPU POINT2;
 DCL UMFPU POINT2;
 DCL P2KREIS CIRCLE2;
  SET MIPU = POINT2(50.,50.);
  SET UMFPU = POINT2(25.,20.);
  SET LINIE = POLYGON(           MIPU,UMFPU) ;
  PLOT(SHIFT2(LINIE,0.,70.));
  SET P2KREIS = CIRCLE2(MIPU,UMFPU);
  PLOT(SHIFT2(P2KREIS,0.,70.));
/** (3.) PUNKT,RADIUS                      **/
 DCL RAD BIN FLOAT;
 DCL KREIS CIRCLE2;
 OPEN PLOT SIZE(17 CM,17 CM);
  SET MIPU = POINT2(0,0);
  RAD = 25.;
  SET KREIS = CIRCLE2(MIPU,RAD);
  SET KREIS = SHIFT2(KREIS,50,50);
  PLOT(KREIS);
  SET MIPU = POINT2(25,0);
  SET KREIS = CIRCLE2(MIPU,RAD);
  SET KREIS = SHIFT2(KREIS,50,50);
  PLOT(KREIS);
  SET KREIS = CIRCLE2(MIPU,RAD);
  DO I = 1 TO 3;
  RAD = RAD*0.8;
  SET MIPU = REF(KREIS);
  SET KREIS = SCALE2(KREIS,0.8,0.8,MIPU);
  SET KREIS = ROTATE2(KREIS,90);
  PLOT(SHIFT2(KREIS,50,50));
  END;
END;
```





CIRCLE2
Funktion
Deklaration

Syntax

```
{ CIRCLE2 (point2,point2,point2, { 'IN'  } )
                                 { 'OUT' }

{ CIRCLE2 (point2, pperiph)

{ CIRCLE2 (point2, lvalue)
```

Erläuterung

Einem Kreisobjekt können auf verschiedene Weisen die bestimmenden Attribute zugefügt werden:

1) Ein Kreis wird bestimmt durch drei Punkte auf seinem Umfang; oder als ein die Verbindungsstrecken dieser Punkte tangierender Kreis (Um- und Inkreis eines Dreieckes). Die Point2 sind GIPSY-2D-Objekte, bei Angabe von 'OUT' liegen sie auf dem Kreisumfang, durch 'IN' wird ein Inkreis erzeugt.

2) Es wird ein Kreis durch zwei Punkte gebildet. Das erste Argument liefert den Mittelpunkt, das zweite einen Punkt auf dem Kreisumfang.

3) Der Kreis wird durch Angabe von Mittelpunkt und Radius festgelegt. Hier ist point2 wieder ein zweidimensionales Punktobjekt, durch lvalue wird die Radiuslänge geben.

Deklaration

```
DCL [level] ident [dim] CIRCLE2
    [storage class] ;
```

Siehe DCL.

Beispiel

```
DCL (P1,P2) POINT;
DCL RICHTUNG(3) DEC FLOAT(6);
DCL X(3,2) DEC FLOAT(6)
     INIT( , , , , , );
DCL POLY1 POLYGON(2);
DCL KREIS CIRCLE;
DCL R DEC FLOAT(6) INIT(5);
SET P1=POINT(X(1,1),X(2,1),X(3,1));
SET P2=POINT(X(1,2),X(2,2),X(3,2);
```

(1) Mittelpunkt - Achsenpunkt - Radius

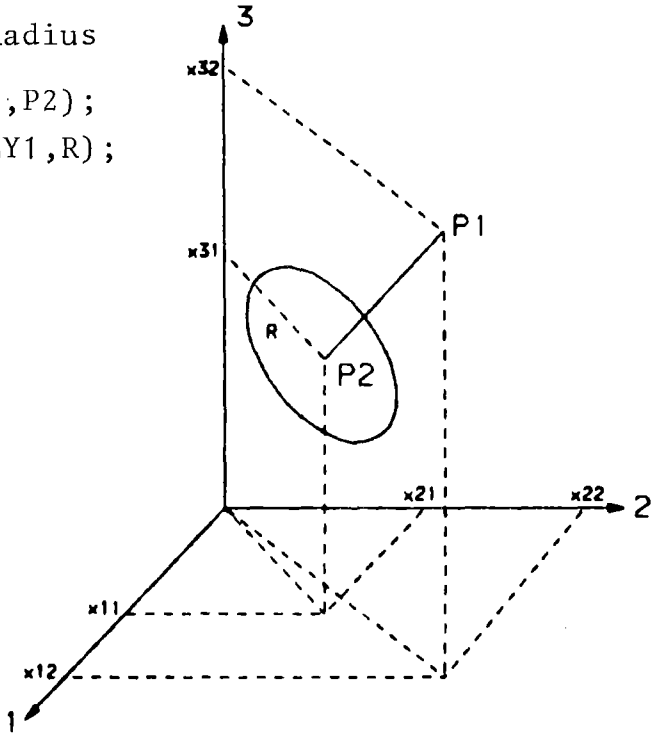
```
    SET KREIS=CIRCLE(P1,P2,5.);
```

(2) Mittelpunkt - Achsenrichtung - Radius

```
    RICHTUNG(1)= x(1,2)-x(1,1);
    RICHTUNG(2)= x(2,2)-x(2,1);
    RICHTUNG(3)= x(3,2)-x(3,1);
    SET KREIS=CIRCLE(POINT(X(1,1),X(2,1),X(3,1),RICHTUNG,R);
```

(3) Achsenpolygonzug - Radius

```
    SET POLY1=POLYGON(P1,P2);
    SET KREIS=CIRCLE(POLY1,R);
```



P1 = Achsenpunkt
P2 = Mittelpunkt

CIRCLE
Funktion
Deklaration

Syntax

```
{ CIRCLE (mittelpunkt-g_ex,
  achsenpunkt-g_ex, radius-1value)
  CIRCLE (mittelpunkt-g_ex,
  achsrichtungsfeld, radius-1value)
  CIRCLE(achsenpolygonzug-g_ex,radius-1value) }
```

Erläuterung

Ein Kreis kann durch

(1) Mittelpunkt-Achsenpunkt-Radius
(2) Mittelpunkt-Achsrichtung-Radius
(3) Achsenpolygonzug-Radius

beschrieben werden.

Deklaration

DCL [level] ident [dim] CIRCLE [storage class] ;

Siehe DCL

Beispiele

```
DCL C(0:N,-2:3) COLLECTION;
/* P(*) seien Punkte */
DO J = 1 TO N;
SET C(0,-2)=COLLECTION(C(0,-2),PLANE(P(0),P(J));
END;
PLOT(BODY(SPACE(C(0,-2))));
SET C(1,1)=INTERSECTION(CIRCLE2(P(3),R),CIRCLE2(P(4),R));
     N=CARDOL(0(1,1));
     IF N=0 THEN; /* kein Schnittpunkt */
     ELSE SET P(5)=NCOL(C(1,1),2);
     /* P(5) wird der zweite Schnittpunkt zugewiesen */
DO J=0 TO N;
DO I=-2 TO 3;
   EMPTY(C(J,I));
END;
END;
```

COLLECTION
Funktion
Deklaration

Syntax

$$\left\{\begin{matrix}\text{COLL}\\ \text{COLLECTION}\end{matrix}\right\} \text{(g\_ex } \left[\text{ ,g\_ex }\right]_{o}^{59} \text{)}$$

Erläuterung

Eine Kollektion ist eine Menge von GIPSY-Objekten, die unter einem Namen zusammengefaßt werden. Sie können dann unter diesem Namen als Ganzes angesprochen werden.

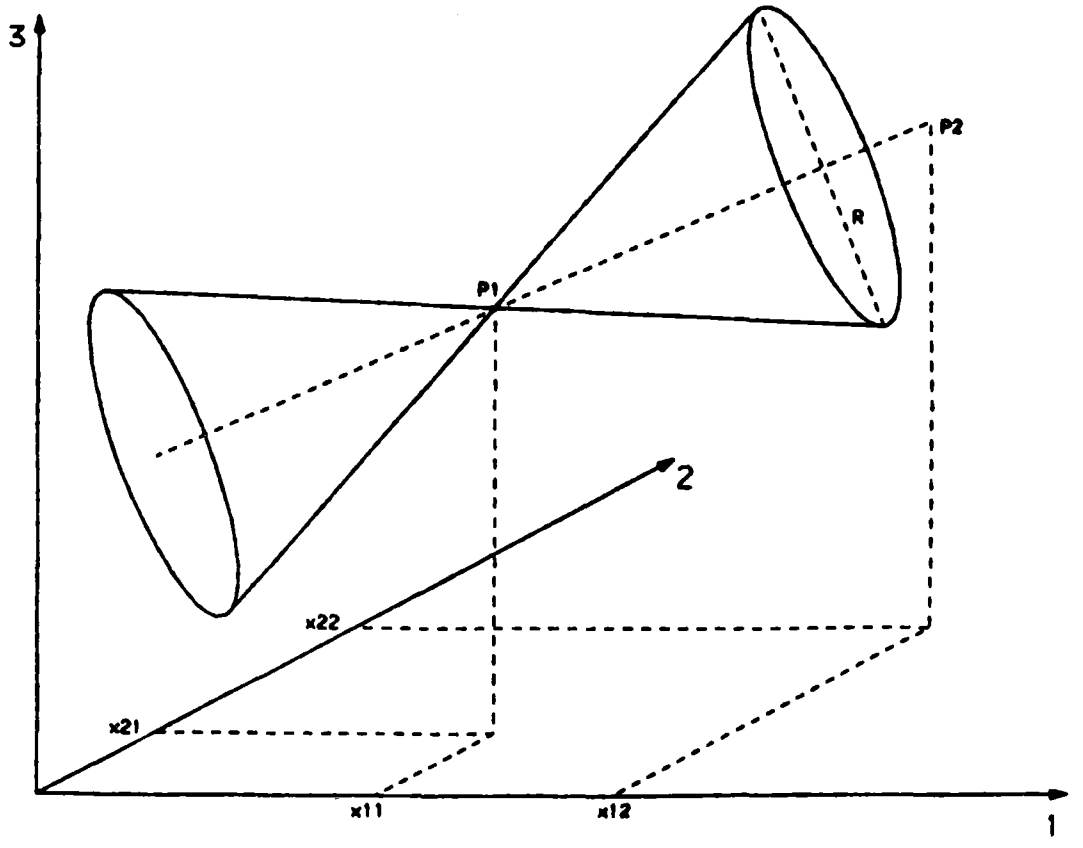
Deklaration

DCL [level] ident [dim] $\left\{\begin{matrix}\text{COLL}\\ \text{COLLECTION}\end{matrix}\right\}$

[storage class] ;

Siehe DCL.

Beispiel




```
DCL (P1,P2) POINT;
DCL POLY1 POLY(3);
DCL RICHTUNG(3) DEC FLOAT(6) INIT (DX1,DX2,DX3);
DCL KEGEL CONE;
SET POLY1=POLY(P1,P2);
```

(1) Spitze - Achsenpunkt - Radius

```
SET KEGEL=CONE(P1,P2,R);
```

(2) Spitze - Achsenrichtung - Radius

```
SET KEGEL=CONE(P1, RICHTUNG, R);
```

(3) Achse - Radius

```
SET KEGEL = CONE (POLY1, R);
```

CONE
Funktion
Deklaration

Syntax

```
{ CONE(spitze-g_ex, achsenpunkt-g_ex, radius-lvalue)      }
{ CONE(spitze-g_ex, achsenrichtung-g_ex, δ/2 -avalue)     }
{ CONE(achspolygonzug--g_ex, δ/2 -avalue)                 }
```

Erläuterung

Ein GIPSY-Kegel (ein unendlich ausgedehnter, doppelter, gerader Kreiskegel) kann in drei Formen beschrieben werden:

(1) durch seine Spitze, einen Punkt auf der Kegelachse und den Radius an diesem Achsenpunkt,

(2) durch die Spitze, die Achsrichtung und den halben Öffnungswinkel,

(3) durch die ersten beiden Punkte eines Polygonzuges, die die Lage der Spitze und die Achse bestimmen, und den halben Öffnungswinkel.

Deklaration

```
DCL [level] ident [dim] CONE
    [storage class] ;
```

Siehe DCL.

Beispiele

```
DCL (x,y,z) DEC FLOAT (6);
DCL CI CIRCLE,P POINT;
DCL A ARC2; DCL PP POLY2(10);
DCL (a1,a2,e1,e2) DEC FLOAT (6);
 x = COORD(P,1);     /* x-Wert von P */
 y = COORD(P,2);     /* y-Wert von P */
 z = COORD(P,3);     /* z-Wert von P */

 x = COORD(CI,1);    /* x-Wert, y-Wert */
 y = COORD(CI,2);    /* und z-Wert */
 z = COORD(CI,3);    /* des Kreises */

 a1= COORD(A,1,1); /* a1 und a2 sind die xy-Koordinaten */
 a2= COORD(A,2,1); /* des Anfangspunktes von A(Bogen) */
 e1= COORD(A,1,2); /* e1, e2 die Koordinaten des Endpunktes */
 e2= COORD(A,2,2);

 x5= COORD(PP,1,5) /* x-Wert des 5.Punktes */
 y5= COORD(PP,2,5) /* y-Wert des 5.Punktes */
```

COORD
Funktion

Syntax

COORD(g_obj, index [,select]);

Erläuterung

COORD liefert den Koordinatenwert eines graphischen Objektes g_obj. Index kann die Werte 1,2 oder 3 annehmen und entspricht dem x,y bzw. z-Wert. Bei Polygonen und Bögen dient select zur Angabe des Punktes, dessen Koordinaten geliefert werden sollen. Zulässig sind alle graphischen Objekte mit Ausnahme von BODY, SPACE und COLL.

Bei allen anderen graphischen Objekten wie CIRCLE2, CONE, PLANE, BALL etc. werden die Koordinaten des entsprechenden Referenzpunktes geliefert.

Beispiel

```
SET PVONT = CURVE2(T,P,TAX,PAX);
SET XY    = CURVE2(X(*,1),X(*,2),X_AX,Y_AX);
SET MESH  = CURVE2(X(*,I),Y(*,I),XLENGTH,YLENGTH;
```

Standard-Anwendung

```
DCL T(M) DEC FLOAT(6),
    P(M) DEC FLOAT(6);
DCL PVONT POLY2(M),
    TAX   AXIS2(4),
    PAX   AXIS2(8);
```

T, P } zuweisen

```
SET TAX = XAXIS(T,20 MM,'TIME'  );
SET PAX = YAXIS(P,,'PRESSURE',,'LINLEFT');
SET PVONT = CURVE2(T,P,TAX, PAX);
PLOT(PVONT,TAX,PAX);
```

CURVE2
Funktion

Syntax

CURVE2(x-vector, y-vector, x-axis, y-axis)

Erläuterung

x-vector : x-Koordinaten, Zahlenvektor

y-vector : y-Koordinaten, Zahlenvektor

x-axis : x-Achse

y-axis : y-Achse

1) Die Funktion CURVE2 dient der Erzeugung von Kurvenzügen, z.B. für die Darstellung an einem Diagramm. Sie liefert einen 2-dimensionalen Polygonzug (POLY2) zurück.

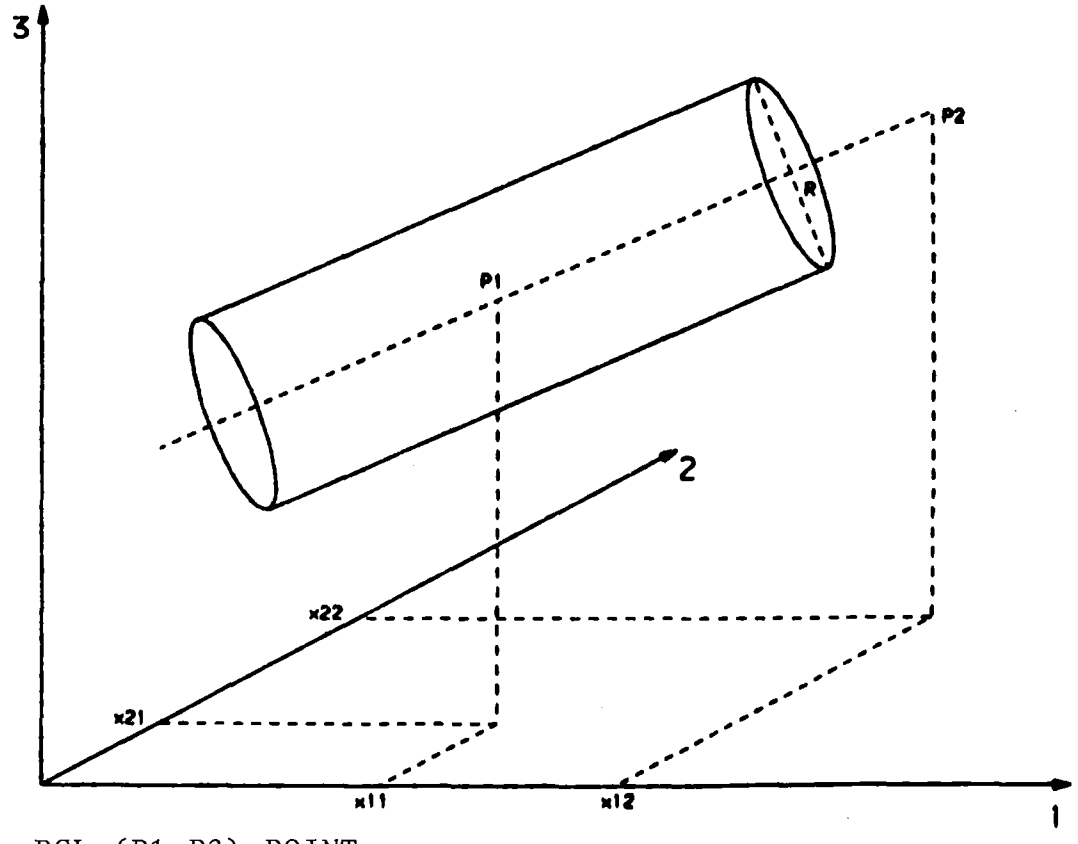
2) Die Koordinaten der Felder (x-vector, y-vector) werden entsprechend den in den Achsen (x-axis, y-axis) vorgegebenen Wertebereichen skaliert und als Koordinaten eines Polygonzuges gespeichert.

3) Der Polygonzug, der mit CURVE2 gefüllt wird, wird so deklariert:

```
DCL g_obj POLY2(M);
```

Hier ist M die Feldgrenze des Polygonzuges und der Vektoren der x- und y-Koordinaten.

Beispiel




```
DCL (P1,P2) POINT;
DCL Z CYLINDER;
DCL POLY1 POLY(2);
DCL RICHTUNG(3) DEC FLOAT(6) INIT(DX1,DX2,DX3);
SET POLY1=POLY(P1,P2);
```

(1) 2 Achsenpunkte, Radius

```
SET Z=CYLINDER(P1,P2,5);
```

(2) Achsenpunkt, Richtung, Radius

```
SET Z=CYLINDER(P1,RICHTUNG,5);
```

(3) Achsenpolygonzug, Radius

```
SET Z=CYLINDER(POLY1, 5);
```

CYLINDER
Funktion
Deklaration

Syntax

```
{ CYLINDER(mittellinienpunkt1-g_ex,
  mittellinienpunkt2-g_ex, radius-1value)

  CYLINDER(mittellinienpunkt-g_ex,
  achsrichtung-PL1_ex,
  radius-1value)

  CYLINDER(achspolygonzug-g_ex,
  radius-1value)                          }
```

Erläuterung

Ein Zylinder wird intern durch einen Punkt auf der Achse, die Richtung der Achse und der Radius beschrieben. Diese Werte können folgendermaßen spezifiziert werden:

1) durch zwei Punkte auf der Achse und den Radius
2) durch einen Punkt auf der Achse, die Achsrichtung und den Radius
3) durch einen Polygonzug, dessen beide ersten Punkte als Achsenpunkte (gemäß 1) interpretiert werden und den Radius.

Ein GIPSY-Zylinder ist in Achsrichtung unendlich ausgedehnt. Statt CYLINDER ist auch CYL zulässig.

Deklaration

```
DCL [level] ident [dim] { CYLINDER }
                        { CYL      }
    [storage class] ;
```

Siehe DCL.

Beispiele

```
DCL HLINE(13) POLY2(2),
    VLINE(5)  POLY2(2);
```

Damit könnte man ein Netz von 12x4 Maschen erzeugen.

```
DCL KREIS POLY2(12);
N = 20;  M = 50;
BEGIN;
     DCL A(0:N,10) POLY2(M);
     CALL SUB(A);
     SUB: PROC(AA);
        DCL AA(*,*) POLY2(*)PARAMETER;
        SET AA(I,J) = POLY2(....);
         :
     END SUB;
  END;
  DCL 1 S,
       2  S1 TEXT2(20),
       2  S2 COLL,
       2  S3(5*10,100)POINT2;
  DCL PP(100,100) POLY2(100) BASED(PQ);
  ALLOC POLY2 PP;
    :
  FREE POLY2 PP;
```

Falsch wäre:

```
  DCL 1 A, 2 P POINT;  /* P ist ohne Qualitfikation nicht
                          eindeutig */
  DCL 1 B, 2 P POINT;
```

DCL
DECLARE
2D-Objekte

Syntax

DCL [level] ident [dim] type [storage class] ;

Erläuterung

level : level innerhalb von PL/1-Strukturen

ident : Name des Objektes nach PL/1-Regeln, mit maximal 20 Zeichen
Namen von GIPSY-Objekttypen und GIPSY-Builtin-Funktionen sind verboten

dim : Dimension nach PL/1-Regeln
Maximal 6 Dimensionen sind erlaubt

type : POINT2 | CIRCLE2 | ARC2 |
POLY2(max-point-number) |
TEXT2(max-char-number) |
AXIS2(max-char-number) | COLL | COLLECTION

storage class: AUTOMATIC | BASED | PARAMETER
CONTROLLED ist verboten, ebenso wie die Attribute STATIC und EXTERNAL

max-point-number: pl1_ex, maximale Punktanzahl des Polygonzuges

max-char-number: pl1_ex, maximale Länge der Zeichenkette des Textes oder der Achsenbeschriftung

1) Für BASED-Objekte gibt es besondere ALLOC- und FREE-Statements
2) Objekte, die Parameter von Unterprogrammen sind, müssen das Attribut PARAMETER tragen. Bei Parameter-Strukturen genügt die Angabe von PARAMETER auf Level 1.
3) Alle Namen von GIPSY-Objekten müssen auch ohne Strukturqualifikation eindeutig sein.
4) Siehe Kapitel 4

Beispiele

```
DCL (A,B,C) PLANE,
    D SPACE(100);
DCL 1 S(20),
     2 S1(4,-10:+10),
      3 S2(10) POLY(100),
      3 S3(10) TEXT(12),
     2 S4 PTR;
CALL SUB(A,B,C,D,S,N);
SUB:PROC(PA,PB,PC,PD,SP,N);
   DCL (PA,PB,PC) PLANE PARAMETER,
        PD SPACE(*) PARAMETER;
   DCL 1 SP(*)PARAMETER,
         2 S1P(4,*),
            3 PS2(*) POLY(100),
            3 PS3(10) TEXT(*),
         2 S4 PTR;
   DCL N BIN FIXED(15);
    :
END SUB;
DCL Z1 BODY BASED(P1),
    Z2 SPACE(20) BASED(P2);
DCL 1 Z3(100) BASED(P3),
         2 Z3A POINT,
         2 Z3B PTR;
ALLOC BODY Z1;              FREE BODY Z1;
ALLOC SPACE Z2;             FREE SPACE Z2;
ALLOC Z3;                   FREE Z3;
```

DCL
DECLARE
3D-Objekte

Syntax

{ DCL / DECLARE } [level] ident [dim] type [storage class] ;

ident : Name nach PL1-Regeln mit maximal 20 Zeichen

dim : Dimension nach PL1-Regeln, maximal 6 Dimensionen

type : POINT | CIRCLE | PLANE | BALL | CYL | CYLINDER | CONE |
POLY(length) | POLYGON(length)
| SPACE(surface number) |
BODY | COLLECTION | COLL

length : Anzahl der Zahlentupel

surface number: maximale Anzahl der Oberflächen, die ein Raumelement aufnehmen kann

storage class: AUTOMATIC | BASED | PARAMETER

level : level innerhalb von Datenaggregaten

Erläuterung

Siehe Kap. 4.

Die Speicherklasse PARAMETER muß für alle graphischen Variablen angegeben werden, die in der Parameterliste stehen.

Die Allokierung von BASED-Variablen erfolgt durch die speziellen Anweisungen ALLOC und FREE (siehe dort).

Beispiel

```
DCL C(10) COLLECTION;
DO J = 1  TO 5;
EDIT HEIGHT(3 MM),WIDTH(4 MM), SYMBOL(J) FOR(C(J));
END;

/* Die Kollektionen C(1) bis C(5) werden editiert.
Alle Texte erhalten die Höhe 3 mm und die Breite 4 mm.
Der Symboltyp variiert von Kollektion zu Kollektion */.
```

Syntax

EDIT attribut [,attribut]* {OF / FOR} (g_obj [,g_obj]*);

attribut = SYMBOL(integer) |
HEIGHT(lvalue) |
EVERY(integer) |
OPEN | CLOSED |
LINETYPE [(] linetype [)] |
LENGTH(lvalue) |
PEN(integer) |
COLOR(integer) |
LINEWIDTH(lvalue) |
WIDTH(lvalue) |
SLANT(avalue)
DISTANCE(lvalue)

Erläuterung

1) Die Darstellungsattribute der Punkt, Texte, Linien in den graphischen Objekten der FOR-Option werden neu festgelegt.

2) Wenn für eines der Objekte eines der Attribute irrelevant ist, so ist es wirkungslos.

3) Siehe CHANGE STANDARD und CHANGE INVISIBLE

4) Siehe Kap. 4.3.2 S.39 und 6.6.2 S.98

Beispiel

```
DCL CO COLLECTION;
SUB :  PROC(C);
DCL C COLLECTION PARAMETER;
DCL S SPACE(3), B BODY, T TEXT2;
etc.
BUILD S = SPACE(....);
BUILD B = BODY(....);
SET C = COLLECTION(REDUCE(B),T);
EMPTY(B);
END SUB;
CALL SUB(CO);
PLOT(CO);
EMPTY(CO);
```

EMPTY

Statement

Syntax

EMPTY(g_obj [, g_obj]*);

Erläuterung

1) Graphische Objekte vom Typ

   COLLECTION |
   SPACE |
   BODY

   werden "geleert"(in den Grundzustand versetzt).

2) Bei allen anderen Objekttypen keine Wirkung.

3) Diese Anweisung ist bei Objekten vom Typ COLLECTION oder BODY erforderlich. Sie muß vor Verlassen des Blockes, in dem die Objekte deklariert sind,ausgeführt werden. Wird dies versäumt, so bleiben im Arbeitsspeicher unzugängliche Daten liegen, die unnötig Platz verbrauchen.

Beispiel

```
END GIPSY;
```

```
DCL GI_PTR POINTER;
END GIPSY LEAVE(GI_PTR);
```

GI_PTR ist ein PL/1-Pointer der bei ENTER GIPSY REENTER(GI_PTR) den Systemzustand beim END GIPSY wiederherstellt.

END GIPSY

Statement

Syntax

END GIPSY;

Erläuterung

Dieser Befehl beendet einen GIPSY-Prozeß. Dieser Begriff ist im Kapitel 6.1 erläutert.

END GIPSY LEAVE

Statement

Syntax

END GIPSY LEAVE(ptrref);

Erläuterung

Dieser Befehl unterbricht einen GIPSY-Prozeß. Dieser Begriff ist im Kapitel 6.1 erläutert.
ptrref ist ein PL/1-Pointer.Bei einer Wiederaufnahme eines GIPSY-Prozesses durch die ENTER GIPSY REENTER-Anweisung muß dieser Pointer wieder angegeben werden.

Siehe Kapitel 6.1

Beispiel

```
ENTER GIPSY;

DCL PLOTFILE FILE;
ENTER GIPSY FILE(PLOTFILE);
```

PLOTFILE ist der GIPSY-Ausgabefile.

```
DCL GI_PTR POINTER;
ENTER GIPSY REENTER(GI_PTR);
```

GI_PTR ist ein PL/1-Pointer, in den durch END GIPSY LEAVE (GI_PTR) vorher der GIPSY-Systemzustand gerettet wurde.

ENTER GIPSY

Statement

Syntax

ENTER GIPSY FILE (file) ;

Erläuterung

Dieser Befehl leitet einen GIPSY-Prozeß ein. Dieser Begriff ist in Kapitel 6.1 beschrieben.

Durch die Angabe FILE(file), wird die Datei mit dem DD-Namen file zum GIPSY-SAVE-File.

ENTER GIPSY REENTER

Statement

Syntax

ENTER GIPSY REENTER(ptrref);

Erläuterung

Dieser Befehl leitet die Wiederaufnahme eines unterbrochenen GIPSY-Prozesses ein. Dieser Begriff ist im Kapitel 6.1. erläutert. ptrref ist der gleiche Pointer, der bei einer vorhergehenden Unterbrechung des GIPSY-Prozesses durch den END GIPSY LEAVE (ptrref) referiert wurde.

Siehe Kap. 6.1

Beispiele

```
DCL P2 POLY2(10),
    P3 POLYGON(8);

DCL W2(20) DEC FLOAT(6)
    INIT(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20),
    W3(24) DEC FLOAT(6)
    INIT(0,0,0,1,1,1,2,2,4,5,6,7,3,3,3,4,4,4,-1,0,-2,0,1,0);

FILL P2 WITH(W2(1),W2(2),W2(3),W2(4),W2(5),W2(6),W2(7),W2(8),
        W2(9),W2(10),W2(11),W2(12),W2(13),W2(14),W2(15),W2(16),
        W2(17),W2(18),W2(19),W2(20));

FILL P3 WITH(0,0,0,SIN(0.3**x ),1,1,1,2,2,-1,0,1,W3(13),W3(14),
        W3(15),W3(16),W3(17),W3(18),0.7,3.2,7.1,-1,-2,-3);
```

FILL

Statement

Syntax

FILL P2 WITH $(x_1,y_1,x_2,y_2,\ldots x_n,y_n)$;

FILL P3 WITH $(x_1,y_1,z_1,x_2,y_2,z_2,\ldots x_K,y_K,z_K)$;

Erläuterung

Mit dieser Anweisung werden 2- bzw. 3-dimensionale Polygonzüge gefüllt. FILL ist daher nur auf Polygone anwendbar und weist dem i-ten Punkt die xy-Werte x_i,y_i bzw. die xyz-Werte x_i,y_i,z_i zu.
Als Werte kommen in Betracht Konstanten, Realzahlen oder PL/1-Expressions, die in einen DEC FLOAT(6)-Wert gewandelt werden können.

P2 - POLY2

P3 - POLY

x_i, y_i, z_i - DEC FLOAT(6)

Beispiele:

1) GIPSY-Objekte mit storage class BASED, nicht Teil einer Struktur:

```
DCL A1 TEXT2(10) BASED(P1);
DCL A2 POLY(100) BASED(P2);
DCL A3(10,10,20) PLANE BASED(P3);
ALLOC TEXT2 A1;
ALLOC POLY A2;
ALLOC PLANE A3;

SET A1 = ...
SET A2 = ...
SET A3(I,J,K) = ...

PLOT(A1,A2,A3(1,1,1));
 :
FREE TEXT2 A1;
FREE POLY  A2;
FREE PLANE A3;
```

2) GIPSY-Objekte, Teil einer BASED Struktur

```
DCL 1 PL1(20) BASED(PP),
      2 G1(2)AXIS2(12),
      2 G2(20)POLY(80);
ALLOC PL1;
SET PL1.G1(1)=....
PLOT(PL1.G2(I),....
  :
FREE PL1;
```

FREE

Statement

Syntax

FREE type name;

Erläuterung

type: GIPSY-Objekttyp, wie im DCL,
also POINT, TEXT, AXIS2, etc.

name: Name eines GIPSY-Objektes mit storage class BASED.

1) GIPSY-Objekte mit storage class BASED, die nicht Teil einer Struktur sind, müssen mit dieser Form des FREE-Statements freigegeben werden.

2) Siehe Kap. 6.9, DCL und ALLOC.

3) Strukturen, die GIPSY-Objekte enthalten, werden mit dem normalen PL/1-FREE-Satement freigegeben.

Beispiel

graphical object

```
     DCL  1  S  BASED(Q)
          2     P PTR,
          2     N,
          2     TEIL(M REFER(S.N),Y),
                3 A BIN FIXED(15),
                3 TEXT CHAR(10);
                3 PUNKT POINT;
     Q = R;
     DO WHILE(Q¬= NULL( ))
     DO J = 1 TO N;
     PLOT(PUNKT(J,2));
     END;
     IF P¬= NULL( ) THEN
        PLOT P->S.TEIL(P->S.N,4). PUNKT
     END;
```

graphical expression

```
SHIFT2(REDUCE(SCALE(Q->PUNKT(1),A,5,0.3)),DX,DY);
```

g_obj

g_ex

Erläuterung

1) graphical objekt g_obj.

Ein ausreichend qualifizierter und indizierter Name eines graphischen Objektes, für das in dem betreffenden Block eine Deklaration bekannt ist.

2) graphical expression g_ex

Entweder ein graphical object(g_obj) oder der Aufruf einer der in diesem Handbuch beschriebenen Funktionen, die ein graphisches Objekt liefern (g_func).

Beispiel

```
DCL A(2) ARC2,
    C(2) CIRCLE2,
    P(2) POLY2;
/x Anweisungen zur Definition von A, C, P, x/
DCL ALL COLLECTION;
 DO J1=1 TO 2; SET ALL=COLL(ALL+A(J1));
 DO J2=1 TO 2; SET ALL=COLL(ALL+A(J2));
 DO J3=1 TO 2; SET ALL=COLL(ALL+A(J3);
SET ALL=COLLECTION(ALL,
        INTERSECTION(A(J1),C(J2)),
        INTERSECTION(A(J1),P(J3)),
        INTERSECTION(C(J2),P(J3)));
END; END; END;
EDIT SYMBOL(3)FOR(ALL);
PLOT(ALL);
```

INTERSECTION

Funktion

2D

Syntax

$$\left\{\begin{matrix}\text{INT}\\ \text{INTERSECTION}\end{matrix}\right\} (\text{g\_ex}_1, \text{g\_ex}_2)$$

g_ex sind beide 2D-Objekte

Erläuterung

1) Das Ergebnis der Funktion ist stets eine Kollektion.

2) Die Ergebniskollektion ist leer, wenn kein Schnittpunkt vorliegt, oder sie enthält die Schnittpunkte.

3) Wenn beide g_ex POLY2 sind, so sind die Schnittpunkte nach den Abschnitten von g_ex_1 geordnet.

4) Wenn nur ein g_ex ein POLY2 ist, so sind die Schnittpunkte nach den Abschnitten dieses Polygonzuges geordnet.

5) Schnittpunkte von CIRCLE2 und ARC2 sind nach steigenden x-Koordinaten (falls diese gleich sind: nach y-Koordinaten) geordnet.

6) Ist eine der g_ex ein 3D-Objekt, so ist die Kollektion leer (Fehlermeldung).

Beispiel

```
DCL P POLYGON(3), ALL COLLECTION;
DCL PL PLANE,
    B  BALL,
    CY CYLINDER,
    CO CONE,
 S(2) SPACE(4),
    BY BODY;
/* Anweisungen zur Definition von P, PL, B, CY, CO */
SET S(1)=SPACE(PL+B+CY+CO);
SET S(2)=SHIFT(S(1),10 CM, 0, 0);
SET BY  =BODY(S(1)+S(2));
SET ALL=COLLECTION (
        INTERSECTION(P,PL),
        INTERSECTION(P,B),
        INTERSECTION(P,CY),
        INTERSECTION(P,CO));
CHANGE STANDARD SYMBOL(5);
SET ALL=COLLECTION(ALL,
        INTERSECTION(P,BY);
PLOT(ALL);
```

INTERSECTION

Funktion

3D

Syntax

$$\left\{\begin{array}{l}\text{INT}\\ \text{INTERSECTION}\end{array}\right\}\ (g\_ex_1,\ g\_ex_2)$$

g_ex sind beide 3D-Objekte

Erläuterung

1) Das Ergebnis ist stets eine Kollektion.

2) Die Ergebniskollektion ist leer, wenn kein Schnittpunkt vorliegt, oder sie enthält die Schnittpunkte.

3) Derzeit sind nur folgende Parameterkombinationen erlaubt:

   POLY mit SPACE, BODY, PLANE, BALL, CYLINDER oder CONE. Ermittelt werden jeweils die Schnittpunkte des ersten Polygonzugabschnittes mit den Flächen bzw. den Oberflächen der Körper.

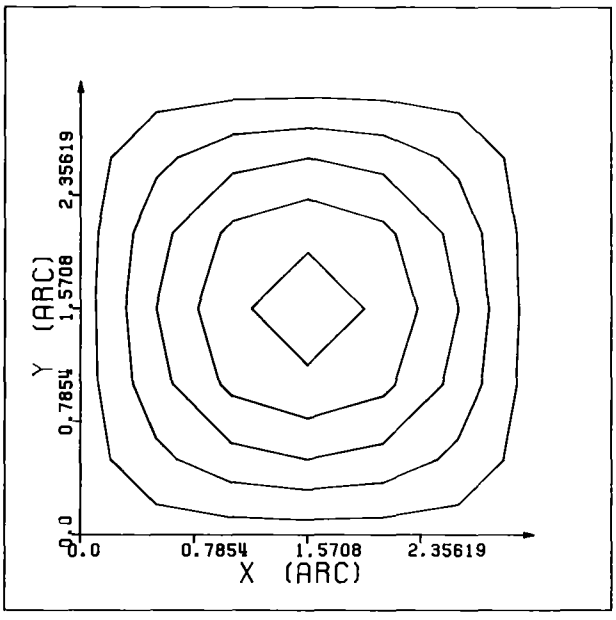
Beispiel:

```
SET TI=ISO2(X,Y,T,TL,XA,YA);

SET PL=ISO2(X1,X2,P,100.E5,XA,YA);
```

Standard-Anwendung:

```
DCL X(M,N),
    Y(M,N),
    T(M,N);

DCL TIS COLLECTION;

DCL XA AXIS2 (10),
    YA AXIS2 (10);

OPEN PLOT DIN A(4) BROAD;

SET XA=XAXIS(X(*,1),0.04    );

SET YA=YAXIS(Y(1,*),0.04    );

SET TIS=ISO2(X,Y,T,TL,XA,YA);

PLOT(TIS);

EMPTY(TIS);
```

```
ISO2:PROC OPTIONS(MAIN)
     REGENT(INIT,NODA,MPOOL=100000,PLOT=STATOS);
DCL M   BIN FIXED(15) INIT(7),
    N   BIN FIXED(15) INIT(7),
    NHL BIN FIXED(15) INIT(5),
    (I  ,J  ) BIN FIXED(15);
ENTER GIPSY;
  DCL BERG_X(M,N) DEC FLOAT(6),
      BERG_Y(M,N) DEC FLOAT(6),
      BERG_Z(M,N) DEC FLOAT(6);
  DCL DA    DEC FLOAT(6),
      ZMAX DEC FLOAT(6) INIT(-1.E10),
      ZMIN DEC FLOAT(6) INIT( 1.E10),
      HIGH DEC FLOAT(6);
  DCL XA AXIS2(10),
      YA AXIS2(10);
  DCL HLINE COLL;
  DA = 2.*3.14159/12.;
  DO J=1 TO N;
     DO I=1 TO M;
        BERG_X(I,J) = (I-1)*DA;
        BERG_Y(I,J) = (J-1)*DA;
        BERG_Z(I,J) = SIN(BERG_X(I,J))*SIN(BERG_Y(I,J));
        IF BERG_Z(I,J)>ZMAX THEN ZMAX=BERG_Z(I,J);
        IF BERG_Z(I,J)<ZMIN THEN ZMIN=BERG_Z(I,J);
     END;
  END;
  OPEN PLOT SIZE (0.160 , 0.160 ) ;
  SET XA = AXIS2(POINT2(0.02,0.02),0.,BERG_X(M,1),
               0.12,0.,3 CM,'X (ARC)',0.005,'LINEXRIGHT');
  SET YA = AXIS2(POINT2(0.02,0.02),0.,BERG_Y(1,N),
               0.12,90 DEG,3 CM,'Y (ARC)',0.005,'LINEXLEFT');
  DO I=1 TO NHL;
     HIGH = ZMIN+(I-0.5)*(ZMAX-ZMIN)/NHL;
     SET HLINE=ISO2(BERG_X,BERG_Y,BERG_Z,HIGH,XA,YA);
     PLOT (HLINE);
     EMPTY(HLINE);
  END;
  PLOT (XA,YA);
END GIPSY;
MESSAGE ACTIVE DEBUG;
FINISH;
END ISO2;
```



ISO2

Funktion

Syntax

ISO2(x-arrayex,y-arrayex,z-arrayex,z-value,x-axis,y-axis)

Erläuterung

x-arrayey : x-Koordinaten, Zahlenfeld

y-arrayex : y-Koordinaten, Zahlenfeld

z-arrayex : z-Koordinaten, Zahlenfeld

z-value : z-Koordinate, Wert der gesuchten Höhenlinie

x-axis : x-Achse für die Skalierung

y-axis : y-Achse für die Skalierung

1) Die Funktion ISO2 ermittelt Höhenlinien. Der Aufruf von ISO2 steht in einer graphischen Zuweisung oder im PLOT-Statement.

2) Die Argumentliste der ISO2-Funktion enthält der Reihe nach die Felder für die x-,y- und z-Koordinaten(zweifach-indizierte Felder gleicher Dimension), den z-Koordinatenwert, für den die Höhenlinie gesucht wird, und die x- und y-Achsen, die für die Skalierung benötigt werden.

3) Die Funktion ISO2 liefert eine Kollektion von Polygonzügen (2-D) zurück.

4) Nach dem Plotten der Höhenlinie sollte der Speicherplatz wieder freigegeben werden:

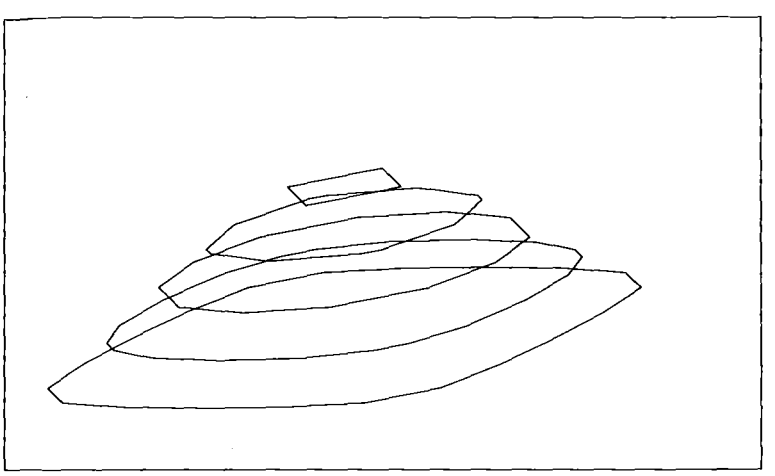
```
EMPTY(g_obj);
```

Beispiel

Standard-Anwendung:

```
DECl X(M,N),
     Y(M,N),
     P(M,N);
DCL  PRESS COLL;
DCL  XAX AXIS2 (10),
     YAX AXIS2 (10);
OPEN PLOT SIZE (0.2,0.2);
CHANGE PROJ PI_N(1.,0.,0.)
       PROJ PARA(+1.,+1.,+1.)
       PROJ ORIGIN(0.05,0.1);
SET XAX=XAXIS(X(*,1) );
SET YAX=YAXIS(Y(1,*) );
SET PRESS=ISO(X,Y,P,100.E5,XAX,YAX,50.E5,1.E-8);
PLOT(PRESS);
EMPTY(PRESS);
```

```
ISO:PROC OPTIONS(MAIN)
      REGENT(INIT,NODA,HPOOL=100000,PLOT=STATOS);
DCL M   BIN FIXED(15) INIT(7),
    N   BIN FIXED(15) INIT(7),
    NHL BIN FIXED(15) INIT(5),
    (I  ,J  ) BIN FIXED(15);
ENTER GIPSY;
  DCL BERG_X(M,N) DEC FLOAT(6),
      BERG_Y(M,N) DEC FLOAT(6),
      BERG_Z(M,N) DEC FLOAT(6);
  DCL DA   DEC FLOAT(6),
      ZMAX DEC FLOAT(6) INIT(-1.E10),
      ZMIN DEC FLOAT(6) INIT( 1.E10),
      ZFAC  DEC FLOAT(6),
      HIGH DEC FLOAT(6);
  DA = 2.*3.14159/12.;
  DO J=1 TO N;
     DO I=1 TO M;
        BERG_X(I,J) = (I-1)*DA;
        BERG_Y(I,J) = (J-1)*DA;
        BERG_Z(I,J) = SIN(BERG_X(I,J))*SIN(BERG_Y(I,J));
        IF BERG_Z(I,J)>ZMAX THEN ZMAX=BERG_Z(I,J);
        IF BERG_Z(I,J)<ZMIN THEN ZMIN=BERG_Z(I,J);
     END;
  END;
  DCL XA AXIS2(10),
      YA AXIS2(10);
  DCL HLINE COLL;
  CHANGE PROJ PI_N (1., 0., 0.)
         PROJ PARA (-1., -1., -0.5)
         PROJ ORIGIN (0.080, 0.090);
  OPEN PLOT SIZE (0.200 , 0.120 ) ;
  SET XA = AXIS2(POINT2(0.08,0.07),0.,BERG_X(M,1),
              0.08,0. ,2 CM,'X (ARC)',0.005,'LINEXLEFT');
  SET YA = AXIS2(POINT2(0.08,0.07),0.,BERG_Y(1,N),
              0.12,90. DEG,2 CM,'Y (ARC)',0.005,'LINEXRIGHT');
  ZFAC  = 0.05/(ZMAX-ZMIN);
  DO I=1 TO NHL;
     HIGH = ZMIN+(I-0.5)*(ZMAX-ZMIN)/NHL;
     SET HLINE=ISO(BERG_X,BERG_Y,BERG_Z,HIGH,
                   XA,YA,ZMIN,ZFAC);
     PLOT (HLINE);
     EMPTY(HLINE);
  END;
END GIPSY;
MESSAGE ACTIVE DEBUG;
FINISH;
END ISO;
```



ISO

Funktion

Syntax

ISO(x-arrayex, y-arrayex, z-arrayex, z-value, x-axis, y-axis, zo-value, zfac-value)

Erläuterung

x-arrayex	:	x-Koordinaten, Zahlenfeld
y-arrayex	:	y-Koordinaten, Zahlenfeld
z-arrayex	:	z-Koordinaten, Zahlenfeld
z-lvalue	:	z-Koordinate, Wert der gesuchten Höhenlinie
x-axis	:	x-Achse für die Skalierung
y-axis	:	y-Achse für die Skalierung
zo-value	:	Bezugswert für z-Koordinate
zfac-value	:	Skalierungsfaktor für z-Koordinate

1) Die Funktion ISO ermittelt Höhenlinien. Der Aufruf von ISO steht in einer graphischen Zuweisung oder im PLOT-Statement.

2) Die Argumentliste der ISO-Funktion enthält der Reihe nach die Felder für die x-, y- und z-Koordinaten (zweifach-indizierte Felder gleicher Dimension), den z-Koordinatenwert, für den die Höhenlinie gesucht wird, die x- und y-Achsen, die für die Skalierung benötigt werden, sowie Bezugswert und Faktor für die Skalierung in der z-Richtung.

3) Die Skalierung in der z-Richtung erfolgt nach folgender Vorschrift:

$$Z_{ISO} = (Z - Z_o) \cdot Z_{fac}$$

Die beiden für die Skalierung benötigten Achsen (x-Achse und y-Achse) können nicht problemgerecht abgebildet werden, da sie vom Typ 2-D sind.

4) Die Funktion ISO liefert eine Kollektion von Polygonzügen (3-D) zurück.

5) Nach dem Plotten der Höhenlinie sollte der Speicherplatz wieder frei gegeben werden:

```
EMPTY(g_obj);
```

6) Die NIVEAU-Funktion erzeugt Höhenlinien ohne Skalierung.

Beispiel

2-dimensional:

```
/* Gitternetz */
DCL XGITTER(N) POLY2(2);
DCL YGITTER(M) POLY2(2);
DCL (CX,DY) DEC FLOAT(6);
DCL (P1,P2) POINT2;
X=0;
DO I=1 TO N;
   SET P1=POINT2(X,0);
   SET P2=POINT2(X,(M-1)*DY);
   SET XGITTER(I)=LINE(P1,P2);
   X=X+DX;
END;
Y=0;
DO I=1 TO M;
   SET P1=POINT2(0,Y);
   SET P2=POINT2((N-1)*DX,Y);
   SET YGITTER(I)=LINE(P1,P2);
   Y=Y+DY;
END;
```

3-dimensional:

```
DCL P POLY(20);
SET P=LINE(POINT(X1,Y1,Z1),POINT(X2,Y2,Z2));
```

LINE

Funktion

Syntax

```
{ LINE(point2,point2) }
{ LINE(point, point)  }
```

Erläuterung

Man kann einem Polygonzug auf mehrere Arten Koordinaten zuweisen. Hier können zwei Punkte mit der Funktion LINE verbunden werden. Die Parameter müssen entweder 2 zweidimensionale Punkte point2 oder 2 dreidimensionale Punkte point sein. Entsprechend ist der Typ des erzeugten Polygonzuges.

Beispiel

gültig:

```
10.3  M
10.3
A+X*J MM
```

sofern A+X*J eine Zahl ergibt.

ungültig:

```
DCL X ENTRY;
X
X CM
4*X MM
0.5*X
```

Alle diese Formen führen zu Fehlermeldungen des PL/1-Compilers von der Art:
"Operand X illegal in element expression"
oder
"conflicting attributes of argument...
to entry..."

lvalue

lunit

Syntax

lvalue = PL1_ex lunit

lunit = MM|CM|M|KM|INCH|FOOT|YARD

Erläuterung

1) Dies ist eine Längenangabe.

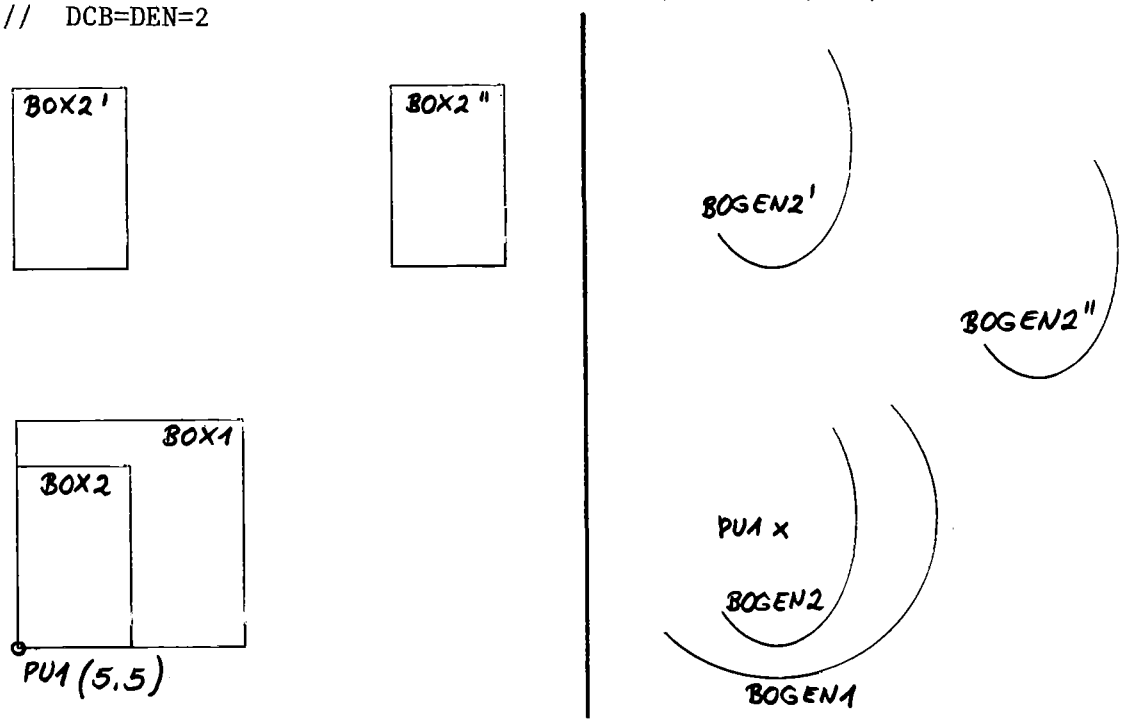
2) PL1_ex muß ein in PL/1 gültiger Ausdruck sein, der in eine DECIMAL FLOAT(6) Darstellung umgewandelt werden kann.

3) Fehlt lunit, so wird die Standardeinheit für Längen benutzt.

   Siehe CHANGE UNIT LENGTH

Beispiele

```
//IRE253MV JOB  (0253,330,POH1A),BECHLER,REGION=1024K,                    00010004
//  NOTIFY=IRE253,MSGLEVEL=(0,0)                                          00020001
// EXEC REGENT,PVOL=PNS001,PLIB='TSO253.TIPSY'                            00030000
//P.SYSIN DD *                                                           00040000
 MOVE:PROC OPTIONS(MAIN) REGENT(INIT,NODA,PLOT=STATOS,MPOOL=600000);     00050008
  ENTER GIPSY;                                                          00060000
          CHANGE UNITS LENGTH MM;                                       00070003
  /**** MOVE MOVETO ****/                                              00080004
  DCL (BOX1,BOX2) POLY2(5);DCL (BOGEN1,BOGEN2) ARC2;                    00090007
  DCL PU1 POINT2;                                                       00100000
   OPEN PLOT SIZE(18 CM,18 CM);                                         00110002
    SET BOX1 = POLY2(POINT2(50.,50.),POINT2(80.,50.),                   00120005
               POINT2(80.,80.),POINT2(50.,80.),POINT2(50.,50.));        00130000
     SET PU1 = REFPOINT(BOX1);                                          00140000
     SET BOX2 = SCALE2(BOX1,0.5,0.8,PU1);                               00150000
    PLOT(BOX1,BOX2);                                                    00160000
    SET BOX2 = MOVE(BOX2,POINT2(0.,50.0));          BOX2 '              00170000
    PLOT(BOX2);                                                         00180000
      SET BOX2 = MOVETO(BOX2,POINT2(100.,100.));    BOX2 "              00190000
   PRINT(BOX2);                                                         00200000
    PLOT(BOX2);                                                         00210000
   OPEN PLOT SIZE(18 CM,18 CM);                                         00220006
    SET BOGEN1 = ARC2(POINT2(50.,50.),POINT2(80.,50.),                  00230007
                 POINT2(80.,80.));                                      00240006
     SET PU1 = REFPOINT(BOGEN1);                                        00250007
     SET BOGEN2 = SCALE2(BOGEN1,0.5,0.8,PU1);                           00260007
    PLOT(BOGEN1,BOGEN2);                                                00270007
    SET BOGEN2 = MOVE(BOGEN2,POINT2(0.,50.0));      BOGEN2'             00280007
    PLOT(BOGEN2);                                                       00290007
      SET BOGEN2 = MOVETO(BOGEN2,POINT2(100.,100.)); BOGEN2"            00300007
   PRINT(BOGEN2);                                                       00310007
    PLOT(BOGEN2);                                                       00320007
                                                                        00330000
    END GIPSY;                                                          00340000
    MESSAGE DEB ACTIVE;                                                 00350001
    FINISH;                                                             00360001
 END MOVE;                                                              00370008
//G.PLOTTAPE DD UNIT=T0800,VOL=SER=S253MV,LABEL=(1,NL),                 00380004
//  DCB=DEN=2                                                           00390000
```



MOVE

Funktion

2D-Bereich

Syntax

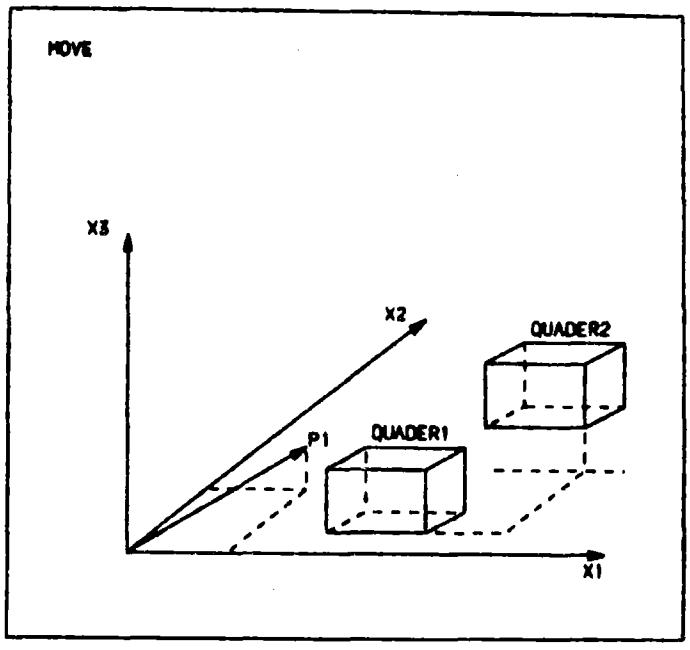
MOVE(g_ex, point2)

Erläuterung

Diese Funktion dient zur Durchführung einer Verschiebung. point2 stellt den Vektor dar, um den das Ergebnisobjekt gegenüber den Argumentobjekt verschoben wird. Das bedeutet, daß g_ex um die Koordinatenwerte des Punktes point2 verschoben wird. MOVE wird auch im 3-D Raum angewendet.

Die Verschiebung direkt zu einem Punkt geschieht mit MOVETO.

Beispiel




```
DCL P1 POINT;
DCL (QUADER1, QUADER2) SPACE(6);
DCL (X1,X2,X3) BIN FLOAT(21);
SET P1=POINT(X1,X2,X3);
SET QUADER1=SPACE( ... ); /*siehe SPACE */
SET QUADER2=MOVE(QUADER1,P1);
```

MOVE
Funktion
3D-Bereich

Syntax

MOVE(3D-move-obj-g_ex,translate-vektor-g_ex);

translate-vector-g_ex muß vom Typ POINT sein.

Erläuterung

Das Objekt 3D-move-obj-g_ex wird um den Vektor translate-vector-g_ex verschoben. Der Translationsvektor wird als Punktobjekt spezifiziert, dessen Koordinaten die Vektorkomponenten sind.

MOVETO
Funktion
2D-Bereich

Syntax

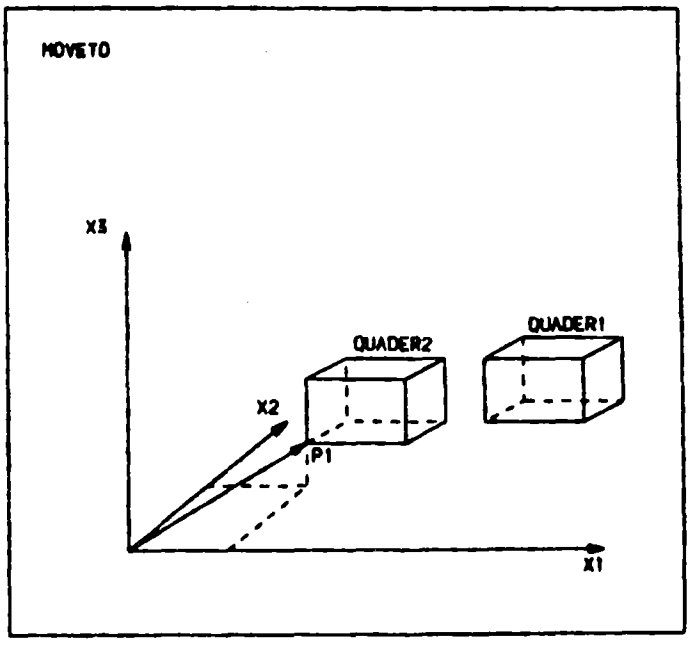
$$\left\{\begin{matrix}\text{MOVETO}\\ \text{MOTO}\end{matrix}\right\}(\text{g\_ex, point2})$$

Erläuterung

Mit MOVETO wird eine Verschiebeoperation zu einem Punkt point2 durchgeführt. Diese Operation verschiebt das Objekt g_ex derart, daß sein Referenzpunkt auf den Punkt point2 zu liegen kommt. Die Verschiebung um die Koordinaten eines Punktes geschieht mit MOVE.
MOVETO wird auch im 3D-Raum angewendet.

Beispiel siehe MOVE 2D-Bereich.

Beispiel MOVETO




```
DCL P1 POINT;
DCL (QUADER1,QUADER2) SPACE(6);
DCL (X1,X2,X3) BIN FLOAT(21);
SET P1=POINT(X1,X2,X3);
SET QUADER1=SPACE...../* siehe SPACE */
SET QUADER2 = MOVETO(QUADER1, P1);
```

Beispiel NCOLL

```
DCL C2 COLL, A2 POINT, B2 BALL;
DCL C  COLL, A POINT, B BALL;
      SET C=COLL(A,B);
      DO J=1 TO CARDOL(C);
       IF OTYP(NCOLL(C,J))='POINT'
         THEN SET A2=NCOLL(C,J);
       ELSE IF OTYP(NCOLL(C,J))='BALL'
         THEN SET B2=NCOLL(C,J);
       ELSE SET C2=COLL(NCOLL(C,J));
      END;
```

MOVETO
Funktion
3D-Bereich

Syntax

{ MOTO / MOVETO } (3D-move-obj-g_ex, new-refpoint-pos-g_ex)

newrefpoint-pos-g_ex muß ein Objekt vom Typ POINT sein.

Erläuterung

Der Referenzpunkt des Objekts 3D-move-obj-g_ex wird an den Punkt new-refpoint-pos-g_ex verschoben.

NCOLL
Funktion

Syntax

NCOLL(coll_g_ex,n)

Erläuterung

coll_g_ex ist eine COLLECTION
n BIN FIXED(15) mit $1 \leq n \leq$ CARDCOL(C)

NCOLL liefert das n-te Element einer COLLECTION.
n muß daher ≥ 1 sein und $\leq$ der Anzahl der Elemente (CARDCOL) dieser COLLECTION.

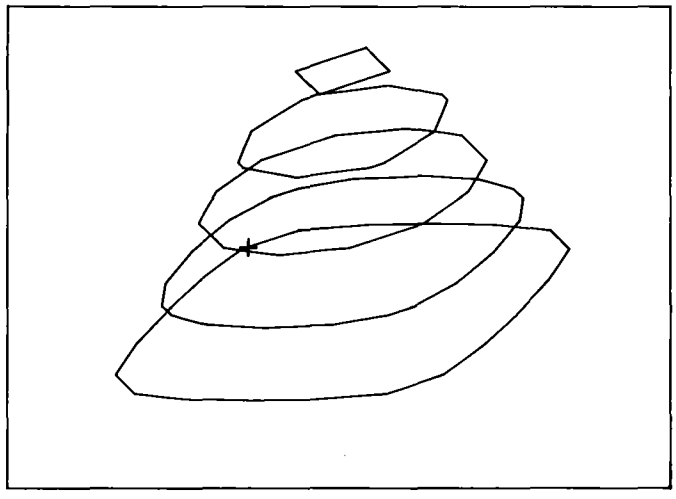
Beispiel

```
SET HLINE=NIVEAU(X1,X2,X3,X3_NIV);
SET PRSL =        SHIFT(NIVEAU(X,Y,P,ISOBAR),0.1,0.1);
SET TI(I)=NIVEAU(X,Y,TEMP,TISO);
```

Standard-Anwendung:

```
DCL X(M,N),
    Y(M,N),
    T(M,N);
DCL TI COLLECTION;
CHANGE PROJ ORIGIN(0.05,0.05)
       PROJ PI_N  (1.,0.,0.)
       PROJ PARA  (+1.,+1.,+1.);
OPEN PLOT SIZE(0.2, 0.2);
SET TI=NIVEAU(X,Y,T,TISO);
PLOT(TI);
EMPTY(TI);
```

```
NIVEAU:PROC OPTIONS(MAIN)
      REGENT(INIT,NODA,MPOOL=100000,PLOT=STATOS);
DCL M   BIN FIXED(15) INIT(7),
    N   BIN FIXED(15) INIT(7),
    NHL BIN FIXED(15) INIT(5),
    (I  ,J  ) BIN FIXED(15);
ENTER GIPSY;
  DCL BERG_X(M,N) DEC FLOAT(6),
      BERG_Y(M,N) DEC FLOAT(6),
      BERG_Z(M,N) DEC FLOAT(6);
  DCL DA   DEC FLOAT(6),
      ZMAX DEC FLOAT(6) INIT(-1.E10),
      ZMIN DEC FLOAT(6) INIT( 1.E10),
      HIGH DEC FLOAT(6);
  DA = 2.*3.14159/12.;
  DO J=1 TO N;
     DO I=1 TO M;
        BERG_X(I,J) = (I-1)*DA;
        BERG_Y(I,J) = (J-1)*DA;
        BERG_Z(I,J) = SIN(BERG_X(I,J))*SIN(BERG_Y(I,J));
        BERG_X(I,J) = BERG_X(I,J)*1.E-2;
        BERG_Y(I,J) = BERG_Y(I,J)*2.E-2;
        BERG_Z(I,J) = BERG_Z(I,J)*5.E-2;
        IF BERG_Z(I,J)>ZMAX THEN ZMAX=BERG_Z(I,J);
        IF BERG_Z(I,J)<ZMIN THEN ZMIN=BERG_Z(I,J);
     END;
  END;
  DCL NP POINT;
  DCL HLINE COLL;
  CHANGE PROJ PI_N (1., 0., 0.)
         PROJ PARA (-1., -1., -1.)
         PROJ ORIGIN (0.040, 0.040);
  OPEN PLOT SIZE (0.110 , 0.080 ) ;
  SET NP = POINT(0.,0.,0.);
  EDIT SYMBOL (3) OF (NP);
  PLOT (NP);
  DO I=1 TO NHL;
     HIGH = ZMIN+(I-0.5)*(ZMAX-ZMIN)/NHL;
     SET HLINE=NIVEAU(BERG_X,BERG_Y,BERG_Z,HIGH);
     PLOT (HLINE);
     EMPTY(HLINE);
  END;
END GIPSY;
MESSAGE ACTIVE DEBUG;
FINISH;
END NIVEAU;
```



NIVEAU

Funktion

Syntax

NIVEAU(x-arrayex, y-arrayex, z-arrayex, z-lvalue)

Erläuterung

x-arrayex : x-Koordinaten, Zahlenfeld
y-arrayex : y-Koordinaten, Zahlenfeld
z-arrayex : z-Koordinaten, Zahlenfeld
z-lvalue : z-Koordinate,Wert der gesuchten Höhenlinie

1) Die Funktion NIVEAU berechnet Höhenlinien. Sie wird in einer graphischen Zuweisung aufgerufen.

2) Die Argumentliste der NIVEAU-Funktion enthält die zweifach indizierten Arrays x,y,z (Koordinatenwerte des darzustellenden Zahlenfeldes) und einem Wert z, für den die Höhenlinien durch das Zahlenfeld gezogen werden soll. Alle Argumente sind vom Typ DECIMAL FLOAT(6).

3) Die Funktion liefert eine Kollektion (COLLECTION) von Polygonzügen (3D) zurück.

4) Die Zahlenfelder x, y und z (gleiche Dimensionierung) enthalten die Koordinatenwerte geordnet, d.h. so, daß benachbarte Werte in den Arrays geometrisch benachbarte Punkte repräsentieren.

5) Es empfiehlt sich, nach den Zeichnen der Höhenlinien den Speicherplatz wieder freizugeben:

```
EMPTY(g_obj);
```

6) Da die Darstellung der Höhenlinien in den meisten Fällen skaliert werden muß, empfiehlt sich die Anwendung von ISO oder ISO2 statt NIVEAU.

Beispiel

3-dimensional:

```
DCL POLY1 POLY(3);
DCL P1 POINT;
SET POLY1=POLY(....);
SET P1=NPOINT(POLY1,2);
```

2-dimensional:

```
DCL  P2 POLY2(N);
DCL  PP2 POINT2;
SET P2=POLY2(.....);
SET PP2=NPOINT2(P2,I+1);
PLOT(PP2);
```


Beispiel

```
DCL (C,CC) COLL;
 DCL A POINT,A2 POINT2,P2 POLY2(2), P POLYGON(3); SET A=POINT(1,1,1);
 SET A2=RED(A);SET P=POLYGON(POINT(0,0,0),POINT(1,1,1),
                             POINT(5,5,5));
 SET P2=POLY2(POINT2(0,0),POINT2(1,1));
 SET A=NPOINT(P,2);   PRINT(A);
 SET A2=NPOINT2(P2,2); PRINT(A2);
 PRINT(NPOLY(P,2,3));
 PRINT(NPOLY(P,1,2)); PRINT(NPOLY2(P2,1,2));
```

NPOINT2

NPOINT

Funktion

Syntax

NPOINT(polygon-g_ex, n-PL1_ex)

NPOINT2(2D-polygon-g_ex, n-PL1_ex)

Erläuterung

NPOINT extrahiert aus dem angegebenen Polygonzug (polygon-g_ex) die Koordinaten des n-ten Punktes und bildet mit ihrer Hilfe ein Punktobjekt. NPOINT liefert einen 3-dimensionalen, NPOINT2 einen 2-dimensionalen Punkt.

NPOLY2

NPOLY

Funktion

Syntax

NPOLY2(poly2_g_ex, begin_PL1_ex, end-PL1_ex)

{NPOLYGON / NPOLY} (poly_g_ex, begin-PL1_ex, end-PL1_ex)

Erläuterung

NPOLY2 erzeugt aus einem Teil eines 2D-Polygonzuges (vom begin_ten bis zum end-ten Punkt einschließlich) einen neuen 2D-Polygonzug.

Mittels NPOLYGON wird aus einem 3D-Polygonzug poly_g_ex ein Teilbereich herausgelöst, dessen Beginn und Ende durch begin-PL1_ex und end-PL1_ex festgelegt ist. NPOLYGON kann nur auf 3D-Polygonzüge angewendet werden.

Beispiel

```
OPEN PLOT DIN A(4) BROAD;  (1)
OPEN PLOT SIZE (35 CM, 40 CM);  (2)
OPEN PLOT SIZE (10 CM, 50 CM) NOFRAME;  (3)
CHANGE UNIT LENGTH CM;
OPEN PLOT DIN A(5) POSITION(20,20);  (4)
x = +13.5;
y = 20;
OPEN PLOT SIZE(30,50) NOFRAME POSITION(x,y);  (5)
OPEN PLOT DIN A(3);  (6)
```

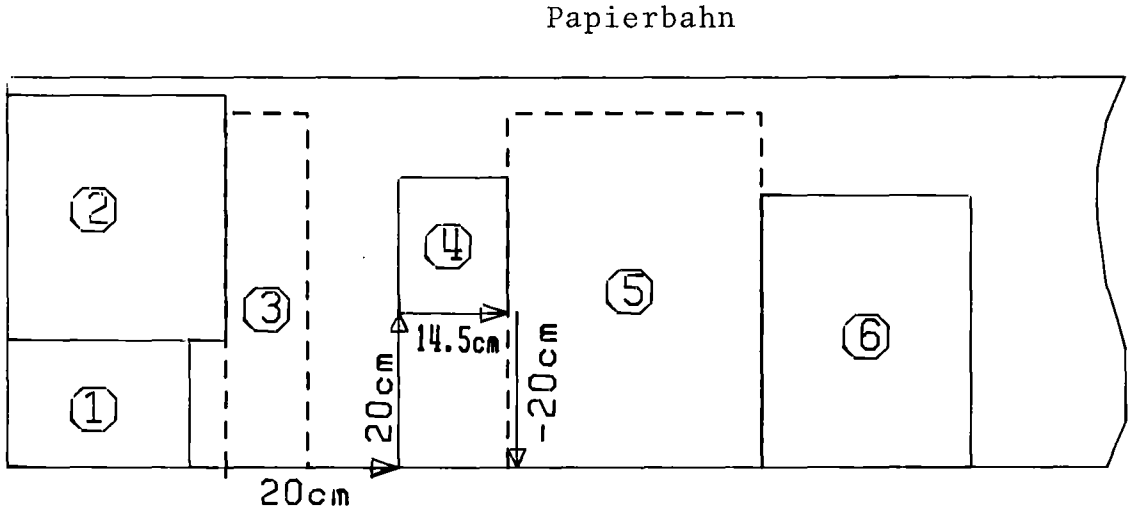


Abb. 9.2: Anordnung der OPEN-PLOT-Rahmen

OPEN PLOT

Statement

Syntax

OPEN PLOT { DIN A(p11_ex) [BROAD] / SIZE(x-1value, y-1value) }

{ [NOFRAME] [POSITION(x_o-1value, y_o-1value)] / [POSITION (x_o-1value,y_o-1value)] [NOFRAME] } ;

Erläuterung

p11_ex: DIN-Papierformat

x-1value
y-1value — Abmessungen des neuen Bildfensters, Papierkoordinaten

x_o-1value
y_o-1value — Lage der linken unteren Ecke des Bildfensters, Papierkoordinaten, bezogen auf die linke untere Ecke des aktuellen Bildfensters

1) Es wird ein neues Bildfenster (Papierteil, Bildschirm) angefordert

2) DIN oder SIZE geben die Abmessungen des neuen Bildfensters

3) Bei Angabe von NOFRAME wird kein Rahmen um das Bildfenster gespeichert

4) POSITION definiert die Lage des neuen Bildfenster bezüglich des alten Bildfensters.Wird POSITION nicht angegeben, so erfolgt eine möglichst günstige Papierausnutzung.

5) Siehe Kap.3, 6.3, 8.1 sowie VIEW.

Beispiel

```
    DCL P1 POINT, P7 BALL;
    DCL WAS CHAR(32) VAR INIT('');
    SET P1=POINT(0,1,1);
    DCL P PTR;
1.  WAS=OTYP(P1);/*WAS='POINT';*/
2.  WAS=OTYP(P7);/*WAS=''BALL_UNINIT';*/
3.  WAS=OTYP(P); /*WAS='ILLEGAL';*/
```

OTYP

Funktion

Syntax

OTYP(g_obj)

Erläuterung

Mit der GIPSY-Builtin Funktion OTYP kann der Typ eines GIPSY-Objektes abgefragt werden. OTYP liefert einen Charakterstring (CHAR(32)VARYING) und zwar

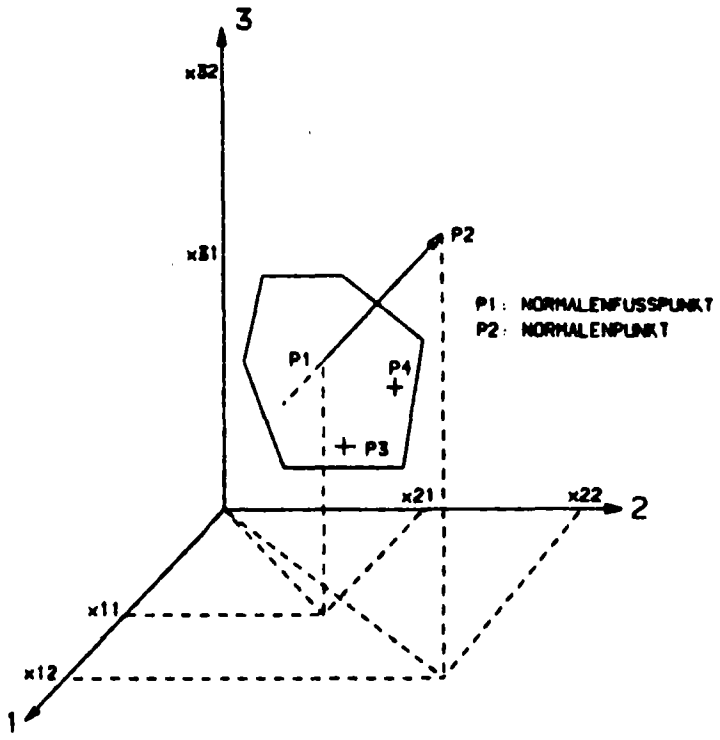
1. 'ILLEGAL' - falls kein GIPSY-Objekt vorliegt-

2. 'POINT', 'TEXT', 'POLY', 'CIRCLE', 'PLANE', 'BALL', 'CYLINDER', 'CONE', 'SPACE', 'COLL', 'POINT2', 'TEXT2', 'POLY2', 'CIRCLE2', 'ARC2', 'AXIS2', 'BODY', 'ARC'
   je nachdem, welches dieser GIPSY-Objekte OTYP übergeben wurde bzw.

3. 'UNDEFINED'- in allen übrigen Fällen.

Im 2.Fall wird zusätzlich '_UNINIT' zurückgeliefert, falls das GIPSY-Objekt nicht initialisiert war, also z.B.'POINT2_UNINIT'.

Beispiel




```
DCL PL PLANE;
DCL(P1,P2,P3,P4)POINT;
DCL POLY1 POLYGON(2);
DCL RICHTUNG(3) DEC FLOAT(6) INIT(DX1,DX2,DX3);
SET POLY1=POLY(COLL(P1,P2));
```

(1) Normalenfußpunkt, Normalenpunkt

```
    SET PL=PLANE(P1,P2);
```

(2) Drei Punkte in der Ebene

```
    SET PL=PLANE(P1,P4,P3);
```

(3) Ebenenpunkt-Normalenrichtung

```
    SET PL=PLANE(P1,RICHTUNG);
```

(4) Normalenpolygon

```
    SET PL=PLANE(POLY1);
```

PLANE

Funktion

Deklaration

Syntax

```
{ PLANE(normalenfußpunkt-g_ex,
        normalenpunkt-g_ex)
  PLANE(ebenenpunkt1-g_ex,
        ebenenpunkt2-g_ex,
        ebenenpunkt3-g_ex)
  PLANE(ebenenpunkt-g_ex,
        normalenrichtung-PL1_ex)
  PLANE(normalenpolygon-g_ex) }
```

(1) Eine GIPSY-Ebene wird festgelegt durch einen Punkt in der Ebene (Normalenfußpunkt)und einen Punkt auf der Normalen oder

(2) drei Punkte in der Ebene (Umlaufsinn legt die Richtung der Normalen fest, Korkenzieherregel) oder

(3) einen Punkt in der Ebene und die Normalenrichtung als Feld von Vektorkomponenten oder

(4) einen Polygonzug, dessen erster Punkt als Normalenfußpunkt und dessen zweiter Punkt als Normalenpunkt dient.

Im Beispiel links werden diese vier Möglichkeiten beispielhaft dargestellt.

Deklaration

DCL [level] ident [dim] PLANE [storage class] ;
Siehe DCL.

Beispiele

```
DCL A ARC2,(P1,P2) POINT2;
DCL PP POLY2(7), C CIRCLE2;
   SET A = ..., SET PP = ....;
   /* Zuweisungen für A, PP, P1, P2, C */
PLOT(A);
PLOT(LINE (P1,P2));
PLOT(SHIFT2(PP,1.EO,O.2));
PLOT(A,C);
```

PLOT

Statement

Syntax

```
{ PLOT(W1  ,W2    ,W3  ...  ,WK  );  }
{ PLOT(W);                           }
```

Erläuterung

W, W1, W2, ... WK sind graphische Ausdrücke oder namentlich angesprochene graphische Objekte.

Beispiel

Aufruf

```
PLOT RELIEF OF(DRUCK);
PLOT RELIEF OF(DRUCK)INDEXVARIATION(2);
PLOT REL OF(TC)SCALED BY(SCALE);

PLOT REL OF(X)LINES('ABOVE');
```

Standard-Anwendung:

```
DCL DRUCK(M,N,3) DEC FLOAT(6);
DO J=1 TO N;
 DO I=1 TO M;
   DRUCK(I,J,1)=X(I,J)*ZX;
   DRUCK(I,J,2)=Y(I,J)*ZY;
   DRUCK(I,J,3)=P(I,J)*ZP;
 END;
END;
OPEN PLOT  SIZE(200,200);
CHANGE PROJ ORIGIN(100.,100.)
       PROJ PARA (1.,1.,1.)
       PROJ PI_N(1.,0.,0.);
PLOT RELIEF OF(DRUCK);
```

PLOT RELIEF
Statement

Syntax

```
PLOT RELIEF OF (tripel)
  [ INDEXVARIATION (index)      ]
  [ SCALED [BY] (scale-arrayexpr)  ]
  [ LINES ('ABOVE') ];
```

tripel : Name des Zahlenfeldes

index : Nr. des variierten Indexes(1|2)

scale-arrayexpr.: Feld d. Skalierungsfaktoren

Erläuterung

1. Die Option PLOT RELIEF erlaubt die reliefartige Darstellung von Problemwertetripeln mit Hilfe von Schnittlinien. Dabei wird immer z über x und y dargestellt.

2. Der Schnittlinienverlauf kann durch INDEXVARIATION gesteuert werden. INDEXVARIATION(1)bedeutet: der erste Feldindex wird variiert, die Schnittlinien verlaufen quer zur x-Richtung. INDEXVARIATION(2)bedeutet: der zweite Feldindex wird variiert, die Schnittlinien verlaufen quer zur y-Richtung.

   Default: beide Indizes werden variiert.

3. Mit der Option SCALED kann die Skalierung der Zahlenfeldwerte vorgenommen werden. Dazu sind ein Feld zu deklarieren:

   ```
   DCL scale (3) DEC FLOAT(6);
   ```

   und die Skalierungsfaktoren zuzuweisen.

4. Die Option LINES mit dem Kennwort 'ABOVE' wird die graphische Ausgabe so gesteuert, daß nur der sichtbare Teil der Reliefoberseite gezeichnet wird.

5. Da das Zahlenfeldtripel normalerweise nicht mit einer SET-Zuweisung gefüllt wird, sind die Koordinaten für die Abbildung in der Einheit Meter (m) zuzuweisen bzw. zu skalieren.

Beispiel

Aufruf

```
PLOT SHADE OF(PYRMD)IN SURFACE(HIGH)
     WITH DIRECTION (VERT)
     WITH DISTANCE (DIST);
PLOT SHADE OF(KEGEL)IN SURF(STUMPF);
```

Standard-Anwendung

```
CHANGE PROJ ORIGIN(100.,100)
       PROJ PI_N    (1.,0.,0.)
       PROJ PARA    (1.,1.,1.);
OPEN PLOT WIDTH(200.) HEIGHT(200.);
PLOT SHADE OF(PYRMD)IN SURF(HIGH);
PLOT(PYRMD);
```

PLOT SHADE

Statement

Syntax

PLOT SHADE [OF] (räuml - objekt)
[IN] SURFACE (ebene1 , [ebenei]*)
[[WITH] DIRECTION (direktion-objekt)]
[[WITH] DISTANCE (distance)] ;

räuml-object : räumliche Objekte

ebene1, ebenei : Schnittebene

direction-object: Abstand der Schraffur

Erläuterung

1. Mit dem Befehl PLOT SHADE wird die Schnittfläche eines räumlichen Objektes (SPACE,BODY) mit einer Fläche (Ebene oder Fläche) bestimmt und schraffiert.
2. Hinter dem Kennwort SURFACE werden die das räumliche Objekt schneidenden Flächen aufgeführt.
3. Die Schraffurlinien werden mit DIRECTION und DISTANCE genauer gekennzeichnet. Die Richtung der Schraffurlinien wird mit einer Ebene beschrieben, so daß die Schraffur senkrecht zur Ebenennormalen verläuft.
4. Als Standard-Werte für Richtung und Abstand der Schraffurlinien sind initialisiert:
   Richtung: ⊥ Ebenen-Normalenrichtung
   Abstand: 10 mm
5. Das graphische Objekt, das die Schraffurlinien und die Kontur der Schnittfläche enthält, ist nach dem Aufruf PLOT SHADE nicht mehr verfügbar.
6. In dem PLOT SHADE-Befehl wird nur die Schraffur gezeichnet. Soll das zugehörige räumliche Objekt auch abgebildet werden, so muß ein zusätzlicher PLOT-Befehl dafür geschrieben werden.

Beispiel POINT2

```
DCL P1 POINT2;
DCL P2(N,M,3) POINT2;
CALL SUB(P2);
SUB: PROC(PPP);
   DCL PPP(*, *, 3) POINT2 PARAMETER;
    :
   SET PPP(I,J,1)=POINT2(x,y);
    :
END SUB;
DCL P3 POINT2 BASED(PP3);
ALLOC POINT2 P3;
SET P3=POINT2(10 CM, 20 CM);
```

Beispiel POINT

```
DCL P1 POINT;
DCL P2 (20,30) POINT  PARAMETER;
CALL S(P2);
  S : PROC(PA);
  DCL PA(*,*)POINT  PARAMETER;
  SET PA(5,6)=POINT(X1,Y1);
  END S;
DCL P3 POINT BASED(PP3);
ALLOC POINT P3;
SET P3=POINT(5.7 MM, 3 CM);
PLOT(P3);
FREE POINT P3;
```

POINT2

Funktion

Deklaration

Syntax

POINT2(x-1value, y-1value)

Erläuterung

POINT2 liefert ein 2-dimensionales Punktobjekt mit den Koordinaten x-1value und y-1value.

Deklaration

DCL [level] ident [dim] POINT2 [storage class] ;

Siehe DCL

POINT

Funktion

Deklaration

Syntax

POINT(X1-1value, X2-1value,X3-1value)

Erläuterung

Ein Punkt-Objekt wird durch die POINT-Funktion aus den drei Koordinatenwerten X1, X2, X3 des GIPSY-3D-Koordinatensystems erzeugt.

Deklaration

DCL [level] ident [dim] POINT [storage class] ;

Siehe DCL.

Beispiel

```
DCL C CIRCLE2,
    A ARC2,
    P1 POLY2(10),
    P2 POLY2(20),
    PP POLY2(1000),
    K  COLL,
    (PK1, PK2, PK3, PK4) POINT2;
SET C=.....
```

Wertezuweisung an A, P1, P2, PK1, PK2, PK3, PK4

```
SET PP=POLY2(PK1, P2, C, A, PK3, P1);
EDIT CLOSED OF(PP);
PLOT(PP);
SET K=COLL(POINT2(10,12),A, PK1, A, PK2);
SET PP=POLY2(K);
PLOT(PP);
SET PP=POLY2(K,P2,PP,P1,A,C,K);
PLOT(PP);
SET PP=POLY2(COLL(P1,P2),POINT2(x,y),INT(P1,P2),ARC(PK1,PK2,PK3));
PLOT(PP);
```

POLY2

Funktion

Deklaration

Syntax

POLY2(g_ex [,g_ex]*)

Erläuterung

Erzeugt einen 2D-Polygonzug aus 2D-Punkten, Polygonzügen, Kreisen und Kreisbögen. Objekte, die nicht erlaubt sind (z.B.Text, Ebene, 3D-Objekte) werden mit Fehlermeldung übergangen.

Es werden auch Kreise und Kreisbögen akzeptiert und daraus Polygonzüge erzeugt, die den Parametern entsprechen, welche beim Zeichnen benutzt werden. (Kreise werden beim Zeichnen durch Polygonzüge dargestellt). Auch Kollektionen von Objekten der Typen POINT2, POLY2, ARC2 und CIRCLE2 sind erlaubt.

Deklaration

DCL [level] ident [dim] POLY2 (length) [storage class] ;

length: Maximale Anzahl von Polygon-Stützpunkten.

Siehe DCL.

Beispiel

```
DCL (POLY1, POLY2, POLY3) POLY(20);
DCL P(3)POINT;
DCL COL1 COLL;
SET POLY1=POLY(P1,ARC(P(1),P(2),P(3)));
SET POLY2=POLY(POLY1,COL1);
SET POLY3=POLY(CIRCLE(P(1),P(2),5.);
```

POLYGON
POLY
Funktion
Deklaration

Syntax

$$\left\{\begin{matrix}\text{POLYGON}\\ \text{POLY}\end{matrix}\right\}(\text{g\_ex}\ [,\text{g\_ex}]^{*})$$

g_ex: graphischer Ausdruck vom Typ POINT, POLY, ARC, CIRCLE oder COLL mit diesen Typen als Subobjekt.

Erläuterung

Ein Polygonzug wird aus einer Menge von graphischen Ausdrücken erzeugt, die in einer Folge von Punkten entwickelt wurden oder bereits aus einer Punktfolge bestehen. Aus Elementen vom Typ ARC und CIRCLE werden Punktfolgen erzeugt wie beim Zeichnen (siehe CHANGE ARC).

Deklaration

$$\text{DCL}\ [\text{level}]\ \text{ident}\ [\text{dim}]\ \left\{\begin{matrix}\text{POLY}\\ \text{POLYGON}\end{matrix}\right\}(\text{number-of-points})\ [\text{storage class}]\ ;$$

Siehe DCL

Beispiel

```
DCL N TEXT2(3);  PRINT(N);
```

```
TEXT2:N
THIS IS AN UNITIALIZED OBJECT
PEN:    1          LINEWD.:999999E-05     COLOR:     0
NCHAR:
X1(*)        0.000000E+00          0.000000E+00
X2(*)        4.999999E-03          0.000000E+00
X3(*)        0.000000E+00          4.999999E-03

STRING:
```

```
DCL Q CIRCLE2;  PRINT(Q);
```

```
CIRCLE2:Q
THIS IS AN INITIALIZED OBJECT
SYMBOL:    1              HEIGHT: 4.999999E-03
LTYPE:     1         LENGTH: 4.999999E-03   EVERY_P:    1
PEN:       1         LINEWD.: 9.999999E-05  COLOR:      0
CENTER   0.000000E+00          0.000000E+00
X1(*)    0.000000E+00          0.000000E+00
X2(*)    0.000000E+00          0.000000E+00
```

```
DCL A POINT; PRINT(A);
```

```
POINT:A
THIS IS AN UNITIALIZED OBJECT
SYMBOL:    1              HEIGHT: 4.999999E-03
PEN:       1         LINEWD.: 9.999999E-05    COLOR:    0
X(*)        0.000000E+00     0.000000E+00      0.000000E+00
```

PRINT

Statement

Syntax

PRINT(A ,B ,C ... ,Z) ;

Erläuterung

A, B, C....Z sind graphische Ausdrücke oder namentlich identifizierte graphische Objekte.

Beispiel

siehe Kapitel 5.2.3

PLOT(REDUCE(BODY));

hat dieselbe Wirkung wie

PLOT(BODY);

ist jedoch aufwendiger in der Ausführung

```
DCL A POINT, B POINT2,C COLLECTION;
SET B=REDUCE(A);
SET C=REDUCE(COLLECTION(A));
SET C=REDUCE(COLLECTION(A,B));
```

REDUCE

Funktion

Syntax

REDUCE(g_ex)

Erläuterung

1) Das Ergebnis der REDUCE-Funktion ist ein graphisches Objekt, dessen Typ vom Typ des graphischen Ausdruckes g_ex abhängt. Es gilt folgende Zuordnung:

Typ von g_ex		Typ des Ergebnisses
2D	POINT2, CIRCLE2,ARC2, TEXT2, POLY2, AXIS2	derselbe Typ wie g_ex
3D	POINT	POINT2
	CIRCLE	CIRCLE2
	ARC	ARC2
	TEXT	TEXT2
	POLY	POLY2
	BALL	CIRCLE
	SPACE	COLLECTION
	BODY	COLLECTION
2D/3D	COLLECTION	COLLECTION
	PLANE, CONE, CYLINDER	Fehler

2) Alle 2D-Objekte werden unverändert übernommen. Alle 3D-Objekte werden mit der aktuellen Projektion (siehe CHANGE PROJEKTION) in die GIPSY-2D-Welt transformiert. Bei Körperkanten wird die Sichtbarkeit geprüft (siehe CHANGE INVISIBLE)

Beispiel

REFPOINT/2

1. Rotieren eines Zylinders um seine Spitze

```
DCL CC CONE;
SET CC = CYL(...);
SET CC = ROTATE(CC,A1,A2,A3,REFPOINT(CC));
/* REFPOINT(CC) liefert die Spitze des Zylinders CC */
```

2. Verschieben des Polygonzuges P1 so, daß sein erster Punkt auf dem ersten Punkt von P2 liegt.

```
DCL (P1,P2) POLY2;
SET P1 = ...; SET P2 = ...;
SET P1 = MOVETO(P1, REFPOINT2(P2));
```

Beispiel

RESET

```
RESET STANDARD LINETYPE;
RESET PROJECTION PI_POINT;
RESET ALL;
```

REFPOINT2
REFPOINT
Funktion

Syntax

REFPOINT (3D-g_ex)

REFPOINT2(2D-g_ex)

Erläuterung

Diese Funktion erlaubt die Extraktion eines Referenzpunktes aus einem graphischen Objekt (siehe Kapitel 4.3.1)
Die Funktion REFPOINT kann nur auf 3D-Objekte angewandt werden, sie liefert einen 3D-Punkt. REFPOINT2 liefert einen 2D-Punkt und ist nur auf 2D-Objekte anwendbar.

RESET
Statement

Syntax

{RESET ALL;
RESET attribut;}

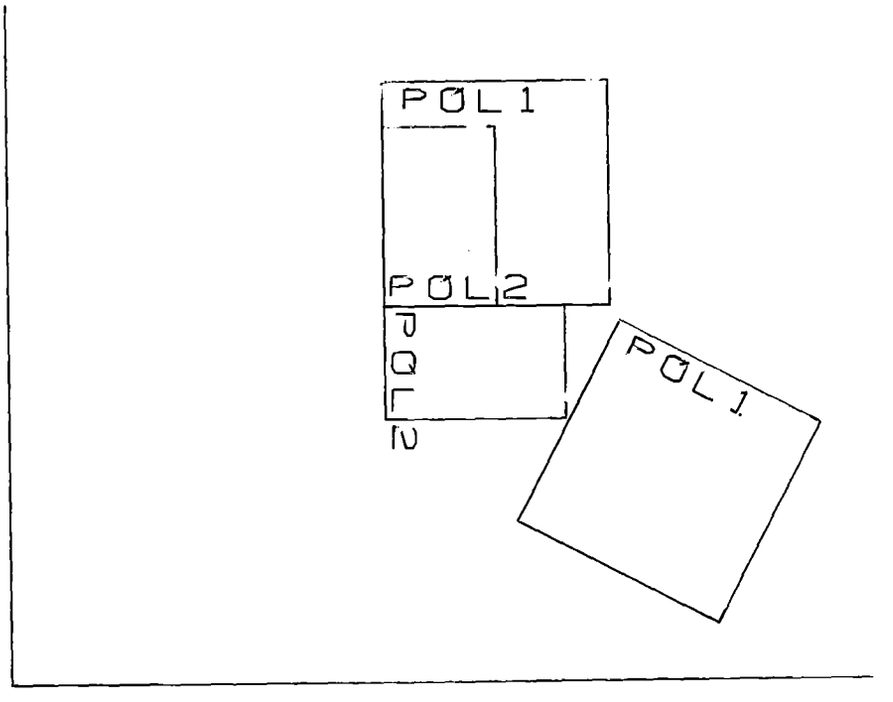
attribut: Genau wie in CHANGE, aber ohne Attribut-Wert.

Erläuterung

RESET setzt die durch CHANGE veränderten Systemattribute auf ihren Standardwert zurück. RESET ALL setzt alle Attribute zurück.

Beispiel

```
 DO;              CHANGE UNITS LENGTH MM;
/**** ROTATE2 ****/
DCL (POL1,POL2) POLY2(5);
 DCL (PU1,BOP1,BOP2) POINT2;
 DCL (POLT1,POLT2)  TEXT2(7);
 OPEN PLOT SIZE(15 CM,15 CM);
  SET POL1 = POLY2(POINT2(50.,50.),POINT2(80.,50.),
              POINT2(80.,80.),POINT2(50.,80.),POINT2(50.,50.));
   SET PU1 = REFPOINT2(POL1);
   SET BOP1 = SHIFT2(PU1,3.,26);
   SET POLT1 = TEXT2(BOP1,0.,'POL1');
   SET POL2 = SCALE2(POL1,0.5,0.8,PU1);
   SET BOP2 = SHIFT2(PU1,1.,1.);
   SET POLT2 = TEXT2(BOP2,0.,'POL2');
   EDIT HEIGHT(3 MM) FOR(POLT1,POLT2);
   PLOT(POLT1); PLOT(POL1);
   PLOT(POLT2); PLOT(POL2);

  SET PU1 = REFPOINT2(POL2);
  SET POL2 = ROTATE2(POL2,270 DEG,PU1);
  PLOT(POL2);
  SET POLT2 = ROTATE2(POLT2,270 DEG,PU1);
  PLOT(POLT2);
  SET POL1 = ROT2(POL1,5.28 RAD);
  SET POLT1 = ROT2(POLT1,5.28 RAD);
  PLOT(POLT1);
  PLOT(POL1);
  SET POLT1 = ROTATE2(POLT1,30 DEG);
  SET POL1 = ROTATE2(POL1,30 DEG);
  PLOT(POL1);
  PLOT(POLT1);
END;
```



ROTATE2

Funktion

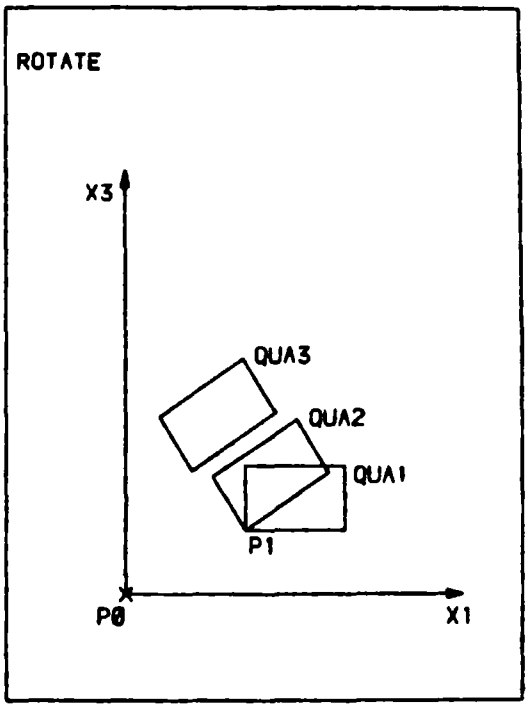
Syntax

$$\left\{\begin{matrix}\text{ROTATE2}\\ \text{ROT2}\end{matrix}\right\}\ (\text{g\_ex, avalue } [\text{,po2-g\_ex}])$$

Erläuterung

Diese Funktion bewirkt die Drehung eines Objektes g_ex um einen Winkel avalue. Gedreht wird im mathematisch positiven Sinn um po2-g_ex oder wenn dieses Argument fehlt, um den Koordinatenursprung. po2-g_ex muß ein Objekt vom Typ POINT2 sein, bei einem sonstigen 2D-Objekt wird der jeweilige Referenzpunkt benutzt.

Beispiel




```
DCL(PO,P1)POINT;
DCL(QUA1, QUA2, QUA3) SPACE(6);
DCL X2 BIN FLOAT(21);
SET QUA1=SPACE...../* siehe SPACE */
SET QUA2=ROTATE(QUA1,O.,X2,O.,P1);
SET QUA3=ROTATE(QUA1,O.,X2,O.);
```

ROTATE

Funktion

Syntax

ROTATE(3D-rotate-obj-g_ex,
X3-avalue, X2-avalue, X1-avalue,
[, rotation-point-g_ex])

Erläuterung

Das Objekt 3D-rotate-obj-g_ex wird um den Koordinatenursprung oder um den Punkt rotation-point-g_ex gedreht und zwar um den Winkel X3-avalue um die X3-Achse,
den Winkel X2-avalue um die X2-Achse
und den Winkel X1-avalue um die X1-Achse.

Beispiel

```
SAVE START FORMATTED;
OPEN PLOT....;          }   Ausgabe auf
PLOT(....);             }   Plotfile

SAVE PLOT;              }
OPEN PLOT....;          }   Ausgabe auf
PLOT(....);             }   Plotfile und
                        }   Plotter

SAVE END;               }
OPEN PLOT......;        }   Ausgabe auf
PLOT(....);             }   Plotter
```

```
IF SAVE_BIT THEN SAVE;
    OPEN PLOT....;      }   Ausgabe auf Plotfile
    PLOT(....);         }   oder Plotter je nach
                            SAVE_BIT.
IF SAVE_BIT THEN SAVE END;
```

SAVE

Statement

Syntax

SAVE [START { → FORMATTED / UNFORMATTED }] [PLOT] ;

Beginn der Plotfile-Ausgabe aller PLOT-Statements.

Erläuterung

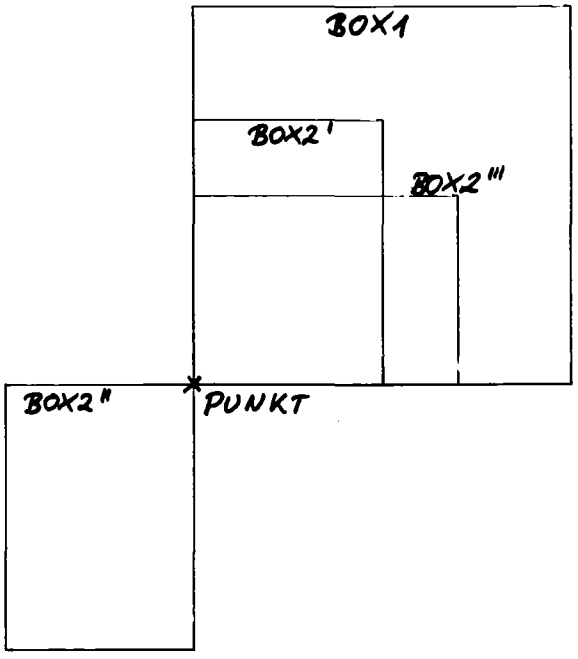
1) SAVE START ist nur beim ersten SAVE gültig, FORMATTED erzeugt einen formatierten Plotfile (für Bearbeitung bei einer anderen Installation), UNFORMATTED einen unformatierten (benötigt weniger Platz).

2) SAVE PLOT erzeugt eine Ausgabe auf Plotter und Plotfile gleichzeitig.

3) Siehe Kapitel 6.5 und 7.3

Beispiele

```
                    CHANGE UNITS LENGTH MM;
DCL PUNKT POINT2;
DCL (BOX1,BOX2) POLY2(5);
SET BOX1 = POLY2(
                POINT2(40.,40.),
                POINT2(90.,40.),
                POINT2(90.,90.),
                POINT2(40.,90.),
                POINT2(40.,40.) );
PLOT(BOX1);
SET PUNKT = REFPOINT(BOX1);
SET BOX2 = SCALE2(BOX1,0.5,0.7,PUNKT);          BOX2'
PLOT(BOX2);
SET BOX2 =   SCALE2(BOX1,-0.5,-0.7,PUNKT);      BOX2''
PLOT(BOX2);

SET PUNKT = REFPOINT(BOX1);
SET BOX2=SCALE2(BOX1,0.7,0.5,PUNKT);            BOX2'''
PLOT(BOX2);
```



SCALE2

Funktion

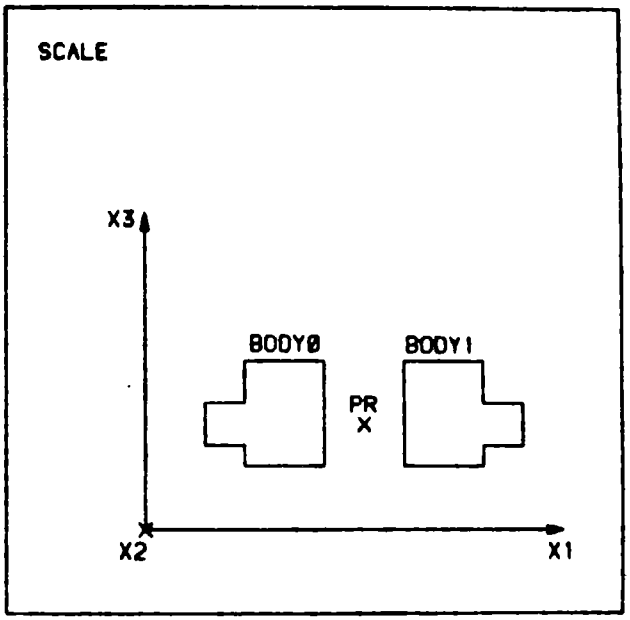
Syntax

$$\left\{\begin{matrix}\text{SCALE2}\\ \text{SCA2}\end{matrix}\right\}\ (\text{g\_ex, scalex, scaley } [\text{,po2-g\_ex}]\ );$$

Erläuterung

Durch die Angabe zweier Faktoren scalex, scaley wird die Skalierung eines graphischen Objektes bewirkt.
Bei Angabe von po2-g_ex bezieht sich die Skalierung auf den angegebenen Punkt, im anderen Fall auf den Ursprung.
Bei negativen Skalierungsfaktoren erfolgt eine Spiegelung an den Koordinaten des Ursprunges oder des Punktes po2-g_ex.

Beispiel SCALE




```
DCL(BODY1, BODY2, B3)BODY;
DCL PR POINT;
SET BODY1=.....;
SET BODY2=SCALE(BODY1,-1.,1.,1.,PR);
SET B3=SCALE(BODY1,5.,3.1,4);
```

Beispiel SET

```
DCL P(2) POINT2;
DCL C(2) COLLECTION;
DCL K(2) CIRCLE;
```

Richtig ist z.B.

```
SET P(1) = REFPOINT2(REDUCE(L(1));
SET P(2) = REDUCE(NCOL(INTERSECTION(K(1),K(2)),1));
SET C(2) = INTERSECTION(K(1),K(2));
SET C(1) = COLLECTION(K(1));
```

Falsch ist z.B.

```
SET K(1) = C(1);
SET C(1) = K(1);
```

SCALE

Funktion

Syntax

SCALE(3D-g_ex,X1-factor, X2-factor, X3-factor
[, ref-point])

Erläuterung

Die Koordinaten des Objektes 3D-g_ex werden mit den Faktoren X1-factor, X2-factor, X3-factor multipliziert, falls kein Referenzpunkt ref-point angegeben wird. Wird ein Referenzpunkt angegeben, so wird bezüglich dieses Punktes skaliert, d.h.

$$y_i = (x_i - x_{Ri}) \cdot X_i\text{-factor} + x_{Ri}$$

SET

Statement

Syntax

SET g_obj = g_ex;

Erläuterung

1) Der Wert des graphischen Ausdruckes g_ex wird dem graphischen Objekt g_obj zugewiesen.

2) Die Objekttypen von g_obj und g_ex müssen gleich sein.

Beispiel

```
DCL P POINT;

SET COORD(P,1)=0;        /* x-wert */
SET COORD(P,2)=0;        /* y-wert */
SET COORD(P,3)=1;        /* z-wert */


DCL B BALL;

SET COORD(B,1)=0;        /* Kugel B */
SET COORD(B,2)=1;        /* mit Mittelp. */
SET COORD(B,3)=2;        /* x=0, y=1, z=2 */


DCL PO(3)POLY(10);

  DCL W(3,10,2) DEC FLOAT(6);
  DO I=1 TO 3;
   DO J=1 TO 10;

    DO K=1 TO 2;

   SET COORD(PO(I),K,J)=W(I,J,K);

   END;
  END;
 END;
/* PO wird mit den Werten WK */
/* initialisiert              */
/* K=1   x-Wert, K=2   y-Wert */
/* J=1,2,...10 entspricht 1., 2.,...10. */
/* Punkt des Polygons       */
/* I=1,2,3 drei Elemente des Feldes PO */
```

SET COORD

Statement

Syntax

SET COORD(g_obj, index [,select])= expr;

Erläuterung

Mit SET COORD werden einem graphischen Objekt g_obj Koordinatenwerte expr zugewiesen. Mit index wird der xyz-Wert angesprochen (1,2,3), mit select derjenige Punkt bei Polygonen bzw. Bögen, dessen Koordinatenwerte zugewiesen werden sollen.

g_obj : kann jedes GIPSY-Objekt sein mit Ausnahme von BODY, SPACE, COLL.

index : ist der Index der Koordinate (1,2,3)

select : wird ignoriert bei allen graphischen Objekten außer POLY, POLY2, ARC und ARC2. Bei ARC, ARC2 gilt:

Anfangspunkt für select = 1
Endpunkt für select = 2
REFPOINT für select ≠ 1,2

Bei POINT2, POINT, CIRCLE2, CIRCLE, TEXT2, TEXT, PLANE, BALL, CYLINDER und CONE ist der Punkt, dessen Koordinatenwerte zugewiesen werden sollen, der REFPOINT dieser Objekte.

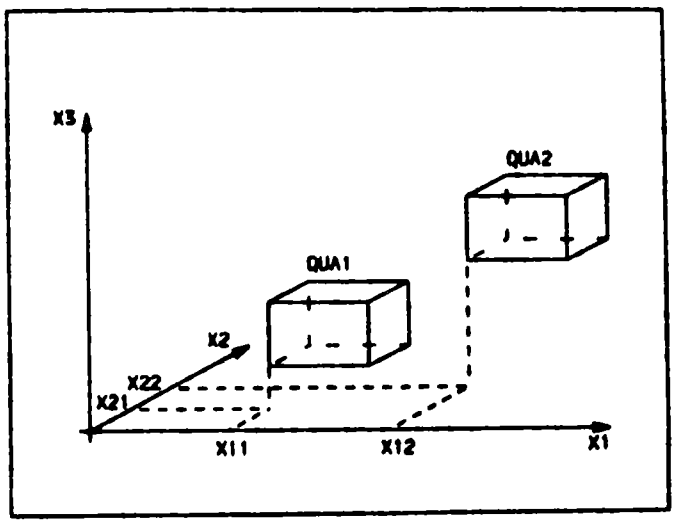
Bei POLY und POLY2 ist der Punkt, dessen Koordinatenwerte zugewiesen werden sollen, der select-te Punkt. Dieser Punkt ist REFPOINT (NPOINT(g_obj, select)) bzw. REFPOINT2(NPOINT2 (g_obj, select)).

expr : PL/1-expression vom Typ DEC FLOAT(6)

Beispiele SHIFT2

```
TEXTE:PROC OPTIONS(MAIN) REGENT(INIT,NODA,PLOT=STATOS,MPOOL=600000);
 ENTER GIPSY;
  DO;              CHANGE UNITS LENGTH MM;
 /**** SHIFT2 ****/
 DCL (POL1,POL2) POLY2(5);
  DCL (PU1,PKT1,PKT2) POINT2;
  DCL (T_SHIFT1,T_SHIFT2) TEXT2(7);
  OPEN PLOT SIZE(16 CM,16 CM);
   SET POL1 = POLY2(POINT2(50.,50.),POINT2(80.,50.),
              POINT2(80.,80.),POINT2(50.,80.),POINT2(50.,50.));
    SET PU1 = REFPOINT2(POL1);
    SET PKT1 = SHIFT2(PU1,3.,26);
    SET T_SHIFT1 = TEXT2(PKT1,0.,'EINS');
    SET POL2 = SCALE2(POL1,0.5,0.8,PU1);
    SET PKT2 = SHIFT2(PU1,10.,10.);
    SET T_SHIFT2 = TEXT2(PKT2,45.,'ZWEI');
    EDIT HEIGHT(3 MM) FOR(T_SHIFT1,T_SHIFT2);
    PLOT(T_SHIFT1);
    PLOT(T_SHIFT2);
  SET PKT2 = REF(POL2);
  SET PKT2 = SHIFT2(PKT2,30.,30.);
  SET T_SHIFT2 = TEXT2(PKT2,30.,'DREI');
  PLOT(T_SHIFT2);
 END;
   END GIPSY;
   MESSAGE DEB ACTIVE;
   FINISH;
END TEXTE;
```

Beispiel SHIFT




```
DCL (QUA1, QUA2) BODY;

SET QUA1 = BODY.....;

SET QUA2 = SHIFT(QUA1, X12-X11, X22-X21, X32-X31);
```

SHIFT2

Funktion

Syntax

{SHIFT2 / SHI2} (g_ex,dx, dy)

Erläuterung

dx, dy: 1value, Verschiebevektor
Das Objekt g_ex, das zweidimensional sein muß, wird in x- und y-Richtung um die Werte dx und dy verschoben. g_ex kann auch eine Kollektion zweidimensionaler Objekte sein. Nicht -2D-Objekte werden mit Fehlermeldung übergangen.

SHIFT

Funktion

Syntax

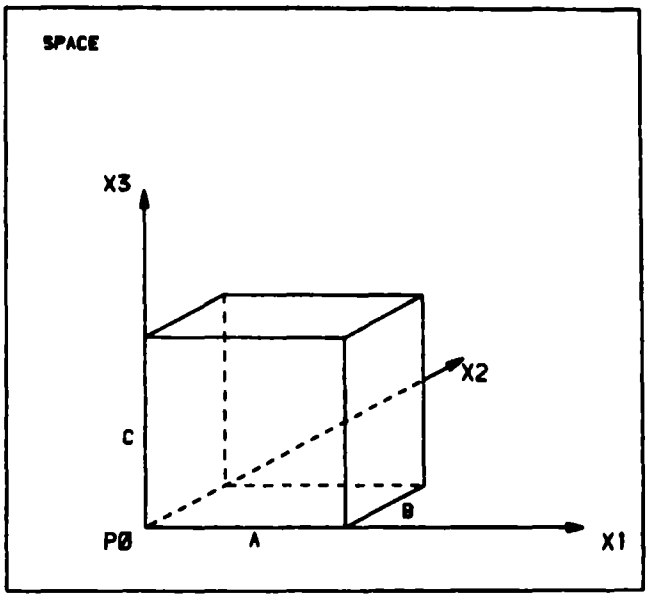
SHIFT(3D-g_ex, delta-x, delta-y, delta-z)

Erläuterung

Das Objekt 3D-g_ex wird um die angegebenen Werte delta-x, delta-y, delta-z verschoben. 3D-g_ex muß ein dreidimensionales Objekt oder eine Kollektion solcher Objekte sein. Andere Objekte werden mit Fehlermeldung übergangen.

delta-x, delta-y, delta-z } 1value, Verschiebevektor

Beispiel




```
DCL (A,B,C) BIN FLOAT(21);
DCL (PL1, PL2, PL3) PLANE;
DCL PO POINT;
DCL QUADER SPACE(6);
SET PO=POINT(0.,0.,0.);
SET PL1=PLANE(PO,POINT(1.,0.,0.));
SET PL2=PLANE(PO,POINT(0.,1.,0.));
SET PL3=PLANE(PO,POINT(0.,0.,1.));
SET QUADER=SPACE(PL1-SHIFT(PL1,A,0.,0.)
                 +PL2+SHIFT(PL2.0.,B,0.)
                 +PL3-SHIFT(PL3,0.,0.,C));
```

SPACE

Funktion

Deklaration

Syntax

SPACE($\pm$ arg-g_ex

$\left[\pm \text{ arg-g\_ex }\right]_{0}^{59}$)

arg-g_ex : raumelement-g_ex | flächen-g_ex

flächen-g_ex : plane-g_ex | ball-g_ex |
cylinder-g_ex | cone-g_ex

Erläuterung

Mit der SPACE-Funktion werden GIPSY-Raumelemente erzeugt, die durch GIPSY-Flächen begrenzt sind. Dabei zeigen die Normalen der Flächen ins Innere des Raumelements. Mit Hilfe des Operators - (minus) kann die Normalenrichtung umgedreht werden. Bei der Definition von Raumelementen muß dafür gesorgt werden, daß sie endlich sind.

Deklaration

DCL [level] ident [dim] SPACE(smax) [storage class] ;

smax : PL1_ex, maximale Anzahl von Flächen, die das Raumelement aufnehmen kann.

Siehe DCL

Beispiel

```
+-------------------------------------------------------------+
!                                                             !
! GRAPHIC STATE INFORMATION: ALL        OF GIPSY RELEASE 3.1  !
! ALL LENGTH VALUES IN METER                                  !
+-------------------------------------------------------------+
! STANDARD ATTRIBUTES                                         !
! DISTANCE      !   2.000000E-02   ! SYMBOL       !   1      !
! HEIGHT        !   4.999999E-03   ! EVERY        !   1      !
! LENGTH        !   4.999999E-03   ! COLOR        !   0      !
! LINEWIDTH     !   0.000000E+00   ! PEN          !   0      !
! SLANT         !   0.000000E+00   ! LINETYPE     !   SOLID  !
! CHARACTER WID!   4.999999E-03   ! OPEN                      !
+-------------------------------------------------------------+
! LENGTH UNIT: METER                                          !
! ANGLE UNIT: DEGREES                                         !
+-------------------------------------------------------------+
! PARALLEL PROJECTION WITH DIRECTION:                         !
!  1.000000E+00   0.000000E+00   0.000000E+00                 !
! PROJECTION ORIGIN:                                          !
!  0.000000E+00   0.000000E+00                                !
+-------------------------------------------------------------+
! PICTURE PLANE:                                              !
! PI_N          !   1.000000E+00   0.000000E+00   0.000000E+00 !
! PI_P          !   0.000000E+00   0.000000E+00   0.000000E+00 !
+-------------------------------------------------------------+
! PROSPECTIVE MATRIX:                                         !
+-------------------------------------------------------------+
!  1.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00  !
!  0.000000E+00   0.000000E+00   1.000000E+00   0.000000E+00  !
!  0.000000E+00   0.000000E+00   0.000000E+00   1.000000E+00  !
+-------------------------------------------------------------+
! INCREMENT FOR COMPUTATION OF INTERSECTIONS:                 !
! STRAIGHT      !   7.999998E-03                              !
! CURVED        !   1.200000E+01                              !
+-------------------------------------------------------------+
! VIEWING TRANSFORMATION:                                     !
! ORIGIN(M)     !   0.000000E+00   0.000000E+00               !
! SCALE         !   1.000000E+00   1.000000E+00               !
+-------------------------------------------------------------+
! CLIPP VALUES:                                               !
! LOWER LEFT    ! -1.000000E+50  -1.000000E+50                !
! UPPER RIGHT   !   1.000000E+50   1.000000E+50               !
+-------------------------------------------------------------+
! INVISIBLE LINE   ATTRIBUTES:                                !
! HEIGHT        !   4.999999E-03   ! EVERY        !   1      !
! LENGTH        !   4.999999E-03   ! COLOR        !   0      !
! LINEWIDTH     !   0.000000E+00   ! PEN          !   0      !
! SYMBOL        !   1              ! LINETYPE     !   DASHED !
+-------------------------------------------------------------+
! MINIMUM/MAXIMUM COORDINATES:                                !
! DOWNLEFT(M)   !   1.000000E+50   1.000000E+50               !
! UPRIGHT(M)    ! -1.000000E+50  -1.000000E+50                !
+-------------------------------------------------------------+
! FACTOR        !   3.937007E+01   ! ARC      !   3.500000E-02 !
+-------------------------------------------------------------+
! PAPER(M)      !   3.000000E+01   5.200000E-01               !
! FIGURE(M)     !   0.000000E+00   0.000000E+00               !
! MAX POSIT.(M)!   0.000000E+00   0.000000E+00               !
+-------------------------------------------------------------+
! PICTURE NR    !   1                                         !
! SAVE UNFORMATTED               ! SAVE CALL OFF              !
! PLOT CALL ON                   ! PLOT ALLOWED               !
+-------------------------------------------------------------+
```

```
STATUS;
```

STATUS

Statement

Syntax

STATUS { → ALL / UNITS / PROJECTION / PLOT } ;

Erläuterung

Mit STATUS kann der aktuelle Zustand der Systemvariablen abgefragt werden. Sind nur bestimmte Variablen von Interesse, so kann der STATUS-Befehl näher spezifiziert werden, z.B.
STATUS PLOT;

Beispiel

TEXT2

siehe folgende Seiten

```
DCL(P1,P2)POINT2;
SET P1=....;Set P2=....;
DCL(T1,T2,T3,T4)TEXT2(20);
SET T1 = TEXT2(P1,20GRAD,'ABC')
    Form1, Standardwerte für höhe, breite, neigung
SET T2 = TEXT2(P1,P2,'ABC',,6 MM)
    Form2,  breite  = 6 mm, Standardwerte für höhe und neigung
SET T3 = TEXT2(P1, 10 DEG,'ABC',,,10 GRAD)
    Form1, neigung = 10°
SET T4 = TEXT2(P1,P2,'ABC',5 MM, 3 MM, 12 DEG)
    Form2, höhe = 5 mm, breite = 3 mm, neigung = 12°
DCL 1 ST BASED(PST),
      2  ST1 TEXT2(10),
      2  ST2(-1:3)TEXT2(15);
ALLOC ST;
SET ST1 = TEXT2(POINT2(10,10),P2,'ST1');
PLOT (ST . ST1);
```

Erläuterung

Dem Textobjekt wird eine Zeichenkette text zugewiesen (PL1_ex). Durch point1 wird die Lage des linken unteren Eckpunktes des ersten Text-Zeichens festgelegt. (Siehe Abb.9.3)

point1 : g_ex	Anfangspunkt des Textes
winkel(Form1):	avalue, Winkel zwischen positiver x-Achse und Textrichtung
text: PL1_ex	Zeichenkette, CHAR(N)VARYING
point2(Form2):	g_ex, Punkt auf dem Strahl der Textrichtung
höhe :	1value:Höhe der Zeichen
breite:	1value:Breite der Zeichen
neigung:	avalue:Neigung der Zeichen

Die Attribute höhe, breite, neigung können weggelassen werden. Nur wenn die letzten Attribute fehlen, dürfen auch die letzten trennenden Kommas fehlen.

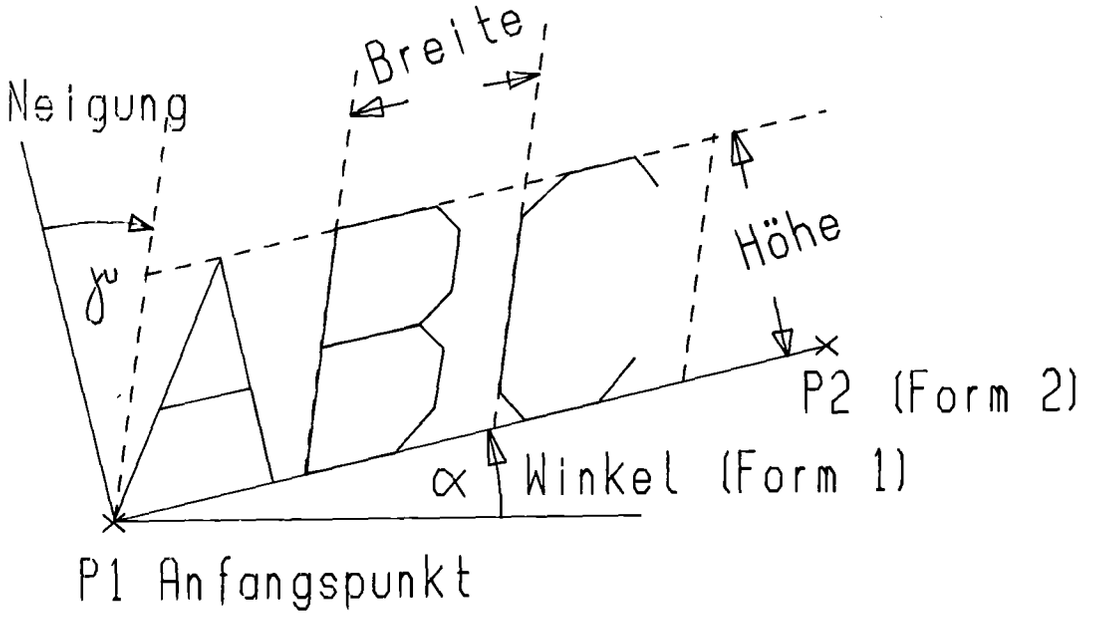


Abb. 9.3: Lage und Attribute von TEXT2

Die Text2-Attribute höhe, breite und neigung können durch das EDIT-Statement noch verändert werden. Standardwerte: Siehe Kap.4.3.2 und CHANGE STANDARD.

TEXT2

Funktion

Deklaration

Syntax

Form1:

```
TEXT2(point1, winkel, text,
     [höhe] , [breite] , [neigung] )
```

Form2:

```
TEXT2(point1, point2, text,
     [höhe] , [breite] , [neigung] )
```

Deklaration

DCL [level] ident [dim] TEXT2(maxchar) [storage class] ;

maxchar : PL1_ex, maximale Zeichenanzahl im Text.

Siehe DCL.

Beispiele siehe S.341

Beispiel

```
DCL (P1,P2,P3) POINT;
DCL T1 TEXT(20);
DCL T2(5,7) TEXT(N);
DCL (T3,T4) TEXT(12);
SET P1 = ..., P2 = ..., P3 = ...;
SET T1 = TEXT(P1,P2,P3,'ABC')
              Standardwerte für höhe, breite, neigung
SET T2(1,1) = TEXT(P1,P2,P3,'ABC',,6 MM)
              breite = 6 mm, Standardwerte für höhe und neigung
SET T3 = TEXT(P1,P2,P3,'ABC',,,10 GRAD)
              neigung = 10°
SET T4 = TEXT(P1,P2,P3,'ABC', 5 MM, 3 MM, 12 DEG)
              höhe = 5 mm, breite = 3 mm, neigung = 12°.
DCL T5 TEXT(10) BASED(PT);
ALLOC TEXT T5;
SET T5=TEXT(P1,P2,POINT(5,6,7),'XYZ123');
EDIT SLANT(10DEG),WIDTH(6 MM), HEIGHT(4 MM) OF(T5);
PLOT(T5);
FREE TEXT T5;
```

TEXT

Funktion

Deklaration

Syntax

TEXT (refpunkt, pos-x-achsen-punkt, textebenenpunkt,
textstring,
[höhe] , [breite] , [neigung])

Erläuterung

Mit dieser Operation wird ein Textobjekt erzeugt. Neben dem Text selbst (textstring) werden Angaben über die Länge des Textes (der Textebene, des Textkoordinatensystems) im Raum erwartet.

Refpunkt ist der 3D-Punkt an dem der Ursprung des Textkoordinatensystems liegt, pos-x-achsenpunkt ein Punkt auf der positiven x-Achse des Textkoordinatensystems (nicht der Ursprung), textebenenpunkt ein Punkt der Textebene mit positivem y-Wert (im Textkoordinatensystem).

höhe : lvalue, Texthöhe

breite : lvalue, Breite eines Textzeichens

neigung: avalue, Winkel zwischen der Senkrechten zur Text-x-Achse und den senkrechten Linien der Textzeichen, siehe Abb.9.6

Die Attribute höhe, breite, neigung können weggelassen werden. Nur wenn die letzten Attribute fehlen, dürfen auch die letzten trennenden Kommas fehlen.

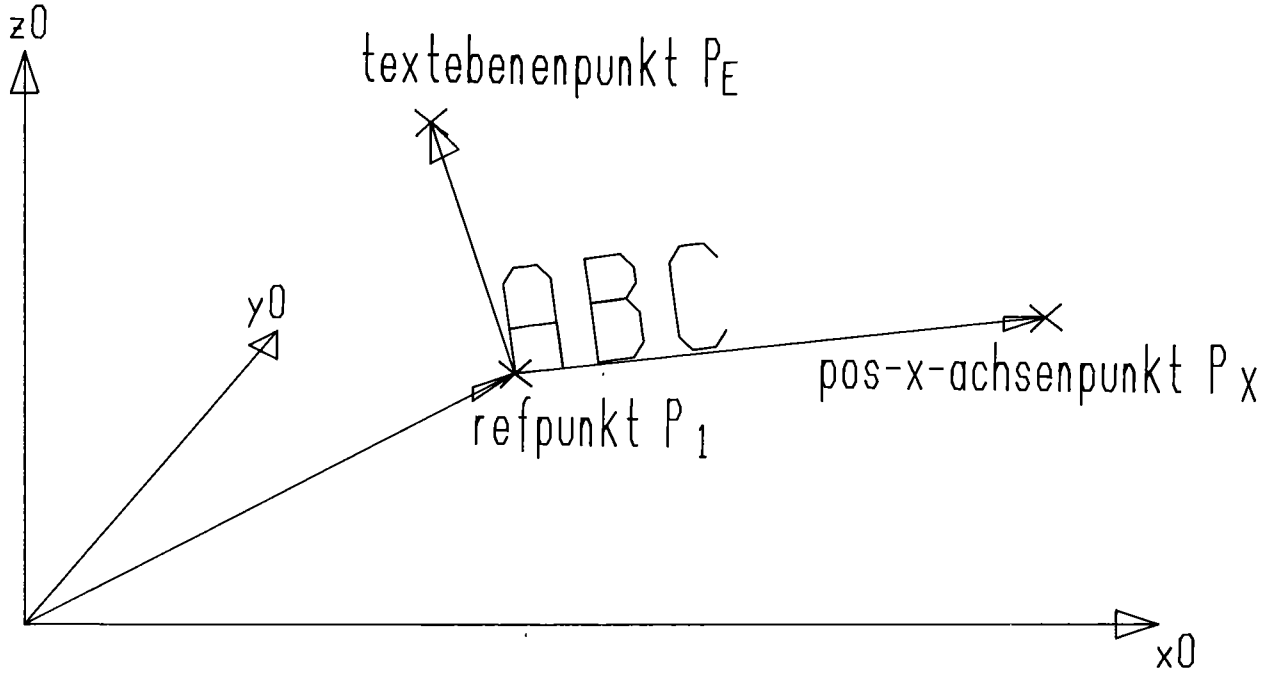


Abb. 9.5: Textebene im Raum

Bezüglich der Transformationen wird ein Text wie ein Polygon behandelt. Nach der Transformation der Punkte P_1, P_2, P_3 wird die Breite aus dem Abstand P_1P_2, die Höhe aus dem senkrechten Abstand P_1P_3 und die Neigung aus dem Winkel $P_2P_1P_3$ errechnet.

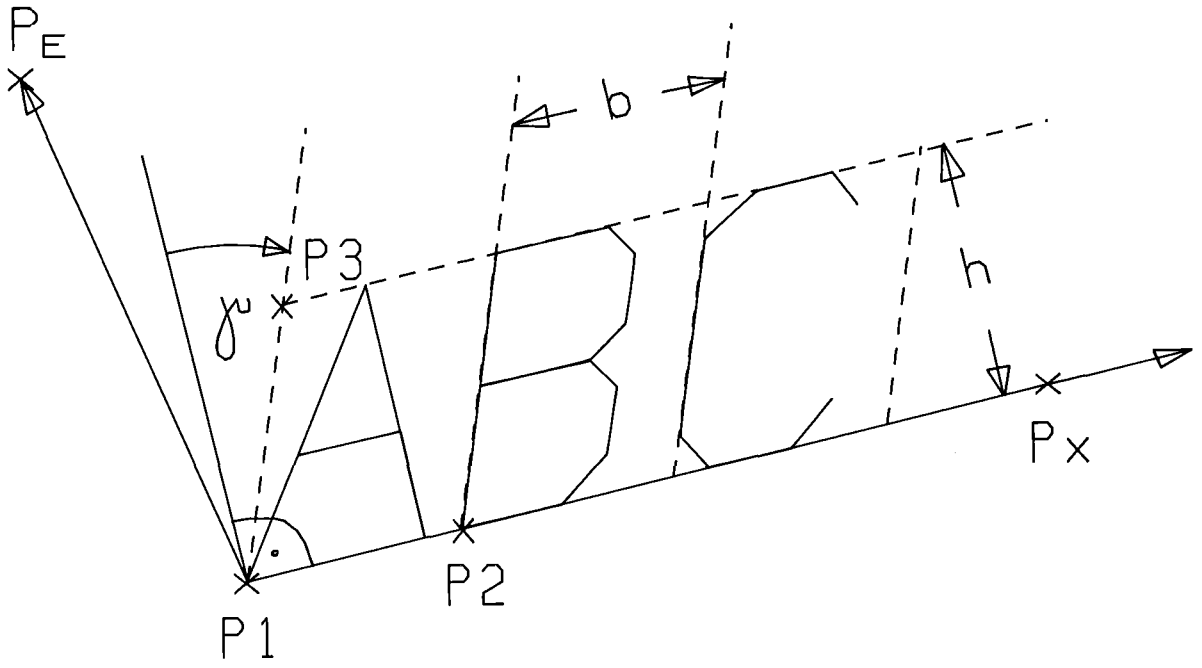


Abb. 9.6: Textattribute

Die Attribute höhe, breite und neigung können durch das EDIT-Statement noch nachträglich verändert werden.

Deklaration

DCL [level] ident [dim] TEXT(maxchar) [storage class] ;

maxchar: PL1_ex, maximale Anzahl von Zeichen im TEXT.

Siehe DCL.

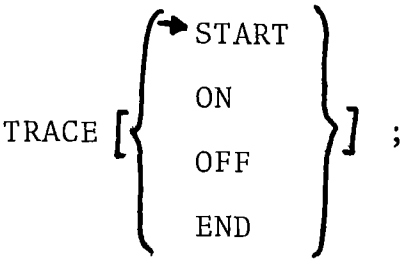
Beispiel

```
TRACE;                          Trace ein

PLOT(BODY(.....));

TRACE OFF;                      Trace Ende

TRACE START;                    Trace ein

SET.......

TRACE END;                      Trace Ende
```

TRACE

Statement

Syntax



Erläuterung

1) Das TRACE-Statement schaltet einen Modulaufruf-Kontrollausdruck ein und setzt zusätzlich eine Kontrollvariable in GIPSY so, daß vor allem bei der Körperanalyse ein umfangreicher Kontrollausdruck erfolgt. Das TRACE-Statement ist wichtig zur Analyse von GIPSY-Fehlern.

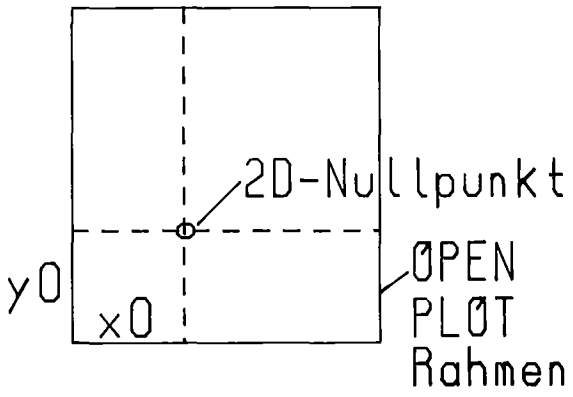
2) TRACE; TRACE START; oder TRACE ON; starten den Kontrollausdruck,

3) TRACE OFF; oder TRACE END; beenden den Ausdruck wieder.

Beispiel

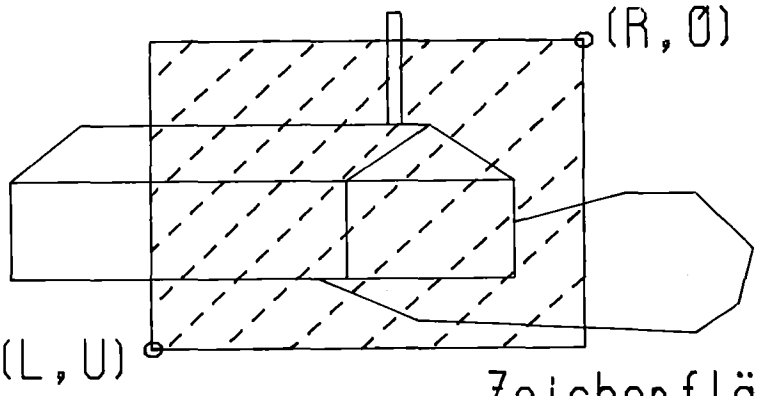
1) VIEW SHIFT(X_o,Y_o);

Lage des 2D-Null-punktes im Bild-rahmen



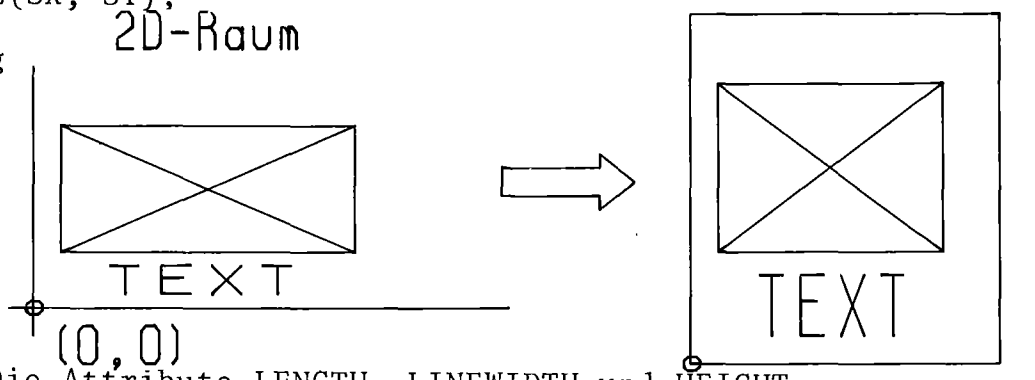
2) VIEW CLIPP(L,U,R,O);

Nur schraffierter Bereich gezeichnet



3) VIEW SCALE(SX, SY);

Skalierung bezüglich 2D-Nullpunkt



Achtung: Die Attribute LENGTH, LINEWIDTH und HEIGHT für Symbole werden dabei mit $\sqrt{sx \cdot sy}$ skaliert!

4) Vergrößerung/Verkleinerung einer gesamten Zeichnung

```
DCL ( SX,SY) DEC FLOAT(6); Vergrößerungsfaktoren
OPEN PLOT SIZE(SX*X, SY*Y);
VIEW SCALE(SX, SY);
PLOT(.....);
  :
  :
```

VIEWING
TRANSFORMATION
Statement

Syntax

VIEWING [TRANSFORMATION] [SHIFT(x,y)]
[CLIPP(l,u,r,o)] [SCALE(sx[, sy])] ;

Erläuterung

VIEWING TRANSFORMATION legt die Abbildungstransformation vom 2D-GIPSY-Raum auf die Zeichenfläche fest. Dabei kann der Nullpunkt des 2D-Raumes auf der Zeichenfläche plaziert werden (SHIFT), ein Bildausschnitt kann angegeben werden (CLIPP) und ein Abbildungsmaßstab kann spezifiziert werden (SCALE). x und y sind die Koordinaten des 2D-Nullpunktes auf der Zeichenfläche. l, u, r, o sind die Koordinaten der linken unteren und rechten oberen Ecke des Bildausschnittes im 2D-GIPSY-Raum. Nur innerhalb des CLIPP-Ausschnittes liegende Objekte oder Objektteile werden gezeichnet. Alles andere wird ohne Fehlermeldung abgeschnitten. sx und sy sind Maßstäbe in x- und y-Richtung. Wird sy nicht angegeben,so ist sy=sx. Siehe Kap.6.3.3 und 8.1.

Beispiel

Ein Feld DRUCK enthalte Druck-Werte P_i, ein Feld ZEIT die zugehörigen Zeitpunkte t_i, $P_i = P(t_i)$, i = 1....n. Der Druckverlauf soll als Diagramm dargestellt werden. Einfache Möglichkeit mit CURVE2, XAXIS und YAXIS:

```
DCL (A1, A2) AXIS2(12);                          /* Achsen */
DCL (DRUCK, ZEIT) (N) DEC FLOAT(6);
DRUCK(I) =.....;  ZEIT(I) = .....;               /* Werte */
DCL P POLY2(N);                                  /* für Kurve */

OPEN PLOT DIN A(4) BROAD;                        /* Bildgröße */
SET A1 = XAXIS(ZEIT);                            /* Zeitachse */
SET A2 = YAXIS(DRUCK);                           /* Druckachse */
SET P = CURVE2(ZEIT, DRUCK, A1, A2);             /* Kurve */
PLOT (A1, A2, P);                                /* Zeichnen */
```

XAXIS

Funktion

Syntax

XAXIS(xfeld, [abstand],[text] , [texthöhe] , [options])

Erläuterung

xfeld : PL1_ex, eindimensionales Feld von x-Werten, zu dem eine Achse erzeugt werden soll.

abstand: 1value, Abstand der Skalenstriche in Papierkoordinaten

text : PL1_ex Textstring zur Achsenbeschriftung. Seine maximale Länge wird in der Deklaration von AXIS2 festgelegt. Standardwert: 'X-AXIS'

texthöhe:1value, Texthöhe

options: $[\{'LIN' \mid 'LOG'\}] \,||\, ['EX'] \,||\, [\{'RIGHT' \mid 'LEFT'\}]$

Options besteht aus einem Textstring für Angaben zu den Achsen.

XAXIS erzeugt zu einem Feld x_i, i=1,....N von DEC FLOAT(6)-Werten und zu dem gerade gültigen OPEN-PLOT-Rahmen eine Standardachse, siehe Abb.9.7. S.354

Zu den übrigen Angaben siehe AXIS2.

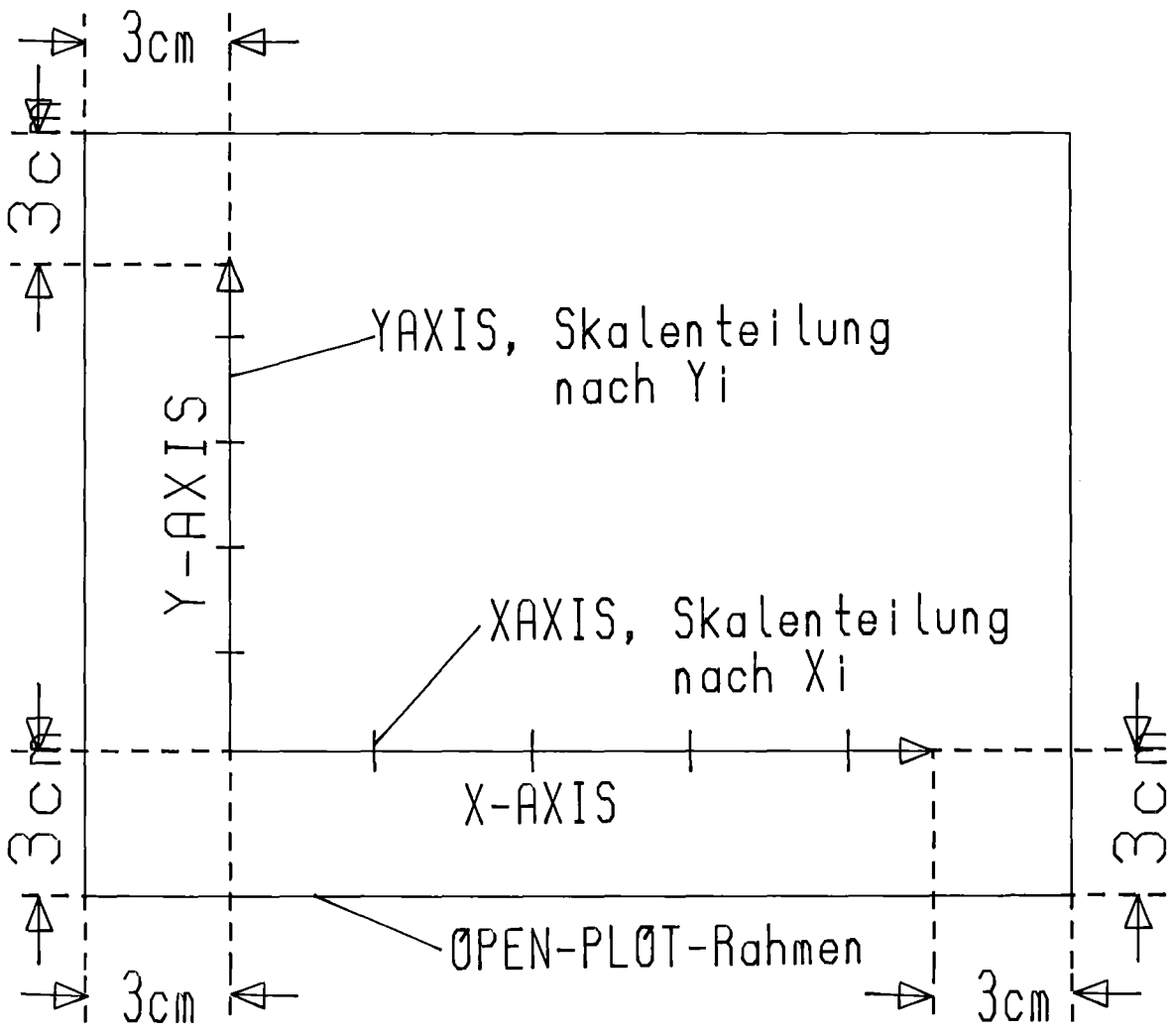


Abb. 9.7: Standardachsen XAXIS und YAXIS

YAXIS

Funktion

Syntax

YAXIS(y-feld, [abstand] , [text] , [texthöhe] , [options])

Erläuterung

y-feld : PL1_ex, Feld von DEC FLOAT(6)-Werten, zu denen eine Standardachse erzeugt werden soll.

abstand: 1value, Abstand der Skalenstriche

text : PL1_ex, Textstring zur Achsenbeschriftung. Seine Länge wird in der Deklaration von AXIS2 festgestellt. Standardwert: 'Y-AXIS'

texthöhe: 1value, Texthöhe

options: [{'LIN' / 'LOG'}] || ['EX'] || [{'RIGHT' / 'LEFT'}]

Options besteht aus einem Textstring für Angaben zu den Achsen.

AXIS erzeugt zu dem Feld y_i, i=1...m von DEC FLOAT(6)-Werten eine Standard-y-Achse.

Siehe auch AXIS2 und XAXIS.

Beispiel siehe S.352

10. Fehlermeldungen

In diesem Kapitel sind die Fehlermeldungen von GIPSY und von einigen Plotfile-Interpretierern alphabetisch aufgeführt. Die GIPSY-Fehlermeldungen sind aufgeteilt in folgende Klassen:

W - Warning	Warnung
E - Error	Fehler
S - Severe Error	Schwerer Fehler

Darüberhinaus gibt es rein informative Meldungen in den Klassen:

I - Informative	Ablaufmeldungen
D - Debug	Ablaufmeldungen des REGENT-Systems

10.1 Fehlermeldungen während der Übersetzung eines GIPSY-Programms

```
W  PLOT DIAGRAM OBSOLETE.USE CURVE INSTEAD
W ( MISSING AFTER OPTION .....
W ) MISSING AFTER OPTION .....
W IN FUNCTION BODY NO COMMA ALLOWED AS SEPARATOR. + ASSUMED
W IN FUNCTION COLLECTION NO + ALLOWED AS SEPARATOR.USE COMMA.
W IN STATEMENT EMPTY ( MISSING
W IN STATEMENT EMPTY ) MISSING
W IN STATEMENT OPEN PLOT AFTER OPTION  DIN A  ( MISSING
W IN STATEMENT OPEN PLOT AFTER OPTION  DIN A  ) MISSING
W IN STATEMENT OPEN PLOT OPTION  DIN A  OR SIZE MISSING.DIN A(4) ASSUMED
W IN STATEMENT PLOT RELIEF "OF" REQUIRED.
W USE POLY(A,B,C) INSTEAD OF POLY(COLL(A,B,C))
E DECLARATIONLIST TOO LONG.ONLY 20 CARDS ALLOWED.FINISH AT WORD .....
E FUNCTION POINT REQUIRES 3 ARGUMENTS
E FUNCTION POINT2 REQUIRES 2 ARGUMENTS
E IN DCL-STATEMENT MORE THAN 50 NAMES IN PARENTHESIZED LIST
E IN DCL-STATEMENT DECLARATIONLIST DOES NOT START WITH NAME,NUMBER OR (
E IN DCL-STATEMENT DIMENSION MISSING.10 ASSUMED
E IN DECLARATATIONLIST ...WRONG DELIMITER OR MISSING , OR )
E IN FUNCTION SPACE ERROR IN ARGUMENT LIST STARTING AT .....
E IN STATEMENT SET ) FOUND AT POSITION .... AND IGNORED
E IN STATEMENT DCL STATIC OR EXTERNAL ATTRIBUTE FOR GIPSY OBJECT INVALID
E IN STATEMENT  LOAD  WRONG OPTION FOUND AFTER LOAD.ALL ASSUMED.
E IN STATEMENT  PLOT SHADE  OPTION SURFACE OR DIRECTION MISSING
E IN STATEMENT ... COMMA MISSING IN LIST AFTER OPTION .....
E IN STATEMENT ... COMMA MISSING IN OPTION .....
E IN STATEMENT ... COMMA REQUIRED TO SEPARATE VALUES.
E IN STATEMENT ... WRONG KEYWORD NEAR .... FOUND.STATEMENT IGNORED.
E IN STATEMENT SET COORD LIST CONTAINS MORE THAN 3 ELEMENTS.
```

```
E IN STATEMENT .... OPTION .... SPECIFIED TWICE.UNDEFINED SELECTION MADE
E IN STATEMENT .....  WRONG KEYWORD FOUND
E IN STATEMENT ..... ( MISSING BEFORE OPTION .....
E IN STATEMENT ..... ( REQUIRED AFTER OPTION .....
E IN STATEMENT ..... ) MISSING BEFORE OPTION .....
E IN STATEMENT ..... ) REQUIRED AFTER OPTION .....
E IN STATEMENT ..... ) REQUIRED BEFORE POSITION .....
E IN STATEMENT ..... = MISSING AFTER ELEMENTLIST.
E IN STATEMENT ..... AT POSITION ..... ERROR IN SYNTAX
E IN STATEMENT ..... COMMA MISSING TO SEPARATE LISTVALUES.
E IN STATEMENT ..... COMMA REQUIRED BEFORE POSITION .....
E IN STATEMENT ..... NO OPTION SPECIFIED.REQUIRED ..... OR .....
E IN STATEMENT ..... NO VALID OPTION FOUND
E IN STATEMENT ..... WRONG SYNTAX IN OR AFTER OBJECTLIST STARTING .....
E IN STATEMENT ...WRONG DELIMITER OR MISSING , OR )
E IN STATEMENT BUILD = MISSING
E IN STATEMENT CHANGE PROJ BOTH PARALLEL AND CENTRAL SPECIFIED.UNDEFINED
E IN STATEMENT CHANGE PROJECTION NO VALID OPTION FOUND
E IN STATEMENT DCL , OR ; MISSING AFTER .....
E IN STATEMENT DCL BUILTIN NAME ... NOT ALLOWED AS OBJECT NAME,IGNORED
E IN STATEMENT DCL NAME ..... TOO LONG BUT HAS NOT BEEN REPLACED
E IN STATEMENT EDIT COMMA MISSING IN OBJECTLIST
E IN STATEMENT FILL ( REQUIRED AFTER WITH
E IN STATEMENT FILL ) MISSING
E IN STATEMENT FILL KEYWORD  WITH  MISSING
E IN STATEMENT GIPSY ALLOC INVALID SYNTAX
E IN STATEMENT GIPSY FREE INVALID SYNTAX
E IN STATEMENT OPEN PLOT ( MISSING AFTER OPTION SIZE
E IN STATEMENT OPEN PLOT ) MISSING AFTER OPTION SIZE
E IN STATEMENT OPEN PLOT BOTH DIN AND SIZE SPECIFIED.UNDEFINED SELECTION
E IN STATEMENT SET COORD ) MISSING
E IN STATEMENT SET MISSING = ASSUMED
E IN STATEMENT SET NO ( ALLOWED AFTER =
E IN STATEMENT SET NO ( ALLOWED BEFORE OBJECT NAME
E IN STATEMENT STATUS WRONG KEYWORD ..... FOUND.IGNORED
E LESS ARGUMENTS SPECIFIED THAN REQUIRED
E MORE THAN 60 ARGUMENTS IN ARGUMENT LIST OF FUNCTION SPACE
E ONE ) TOO MANY AT END OF STATEMENT
S EXPRESSION AFTER OPTION  .....  TOO LONG
S EXPRESSION STARTING ..... TOO LONG.STATEMENT IGNORED
S GIPSY OBJECT HAS MORE THAN 6 DIMENSIONS,EXECUTION IMPOSSIBLE
S IN STATEMENT .... EXPRESSION BEFORE POSITION .... TOO LONG.IGNORED
S IN STATEMENT .... EXPRESSION TOO LONG AT POSITION ....
S IN STATEMENT ..... END OF STATEMENT AFTER = .STATEMENT IGNORED
S IN STATEMENT ..... OPTION ..... INVALID AND HAS BEEN IGNORED
S IN STATEMENT ..... SEMICOLON ENCOUNTERED BEFORE END OF STATEMENT.
S IN STATEMENT ..... SEVERE ERROR DETECTED BEFORE POSITION ..... IGNORED
S IN STATEMENT FILL WRONG KEYWORD FOUND STARTING ..... STATEMENT IGNORED
S IN STATEMENT EDIT NAME IN OBJECTLIST TOO LONG.STATEMENT IGNORED       AT ....
S NO EXPRESSION OR REAL VALUE FOUND IN VALUE-LIST AT POSITION .....
S STRINGSIZE CONDITION RAISED AT  .....
```

10.2 Fehlermeldungen während der Ausführung eines GIPSY-Programmes

```
D END GIPSY LEAVE
D ENTER GIPSY REENTER
I **** ENTER GIPSY **** RELEASE ......
I ***** END GIPSY ***** RELEASE ....
I LOAD ALL PERFORMED
I LOAD 2D PERFORMED
I LOAD 3D PERFORMED
I RELEASE PERFORMED
W CHARACTER HEIGHT=.....>AXIS LENGTH=..... AND SET TO STANDARD VALUE=...
W CHARACTER HEIGHT=0 AND SET TO STANDARD VALUE=.....
W EDITED OBJECT IS YET UNINITIALIZED
W END OF TEXT ..... STARTING : .... OUTSIDE PLOT FRAME
W MIN VALUE = MAX VALUE FOR AXIS2.MAX VALUE SET TO .....
W PART OF POLY2/ARC2/CIRCLE2/AXIS2 ..... OUTSIDE PLOT FRAME
W PARTS OF POLY2 .... OUTSIDE PLOT FRAME
W PARTS OF TEXT2 .... STARTING: .... OUTSIDE PLOT FRAME
W POINT ..... OUTSIDE PLOT FRAME
W SCALE MARK DISTANCE=.....>AXIS LENGTH=..... AND SET TO STANDARD VALUE=
W SCALE MARK DISTANCE=0 AND SET STANDARD VALUE=..... IN AXIS2
W SPECIFIED INDEX > POINT NUMBER IN POLY
W START OF TEXT ..... STARTING : .... OUTSIDE PLOT FRAME
W TEXT2 STARTING .... OUTSIDE PLOT FRAME
W VALUES,PRECISELY LYING ON H ...,WERE CHANGED BY ...IN ISO/ISO2/NIVEAU
E ANGLE BETWEEN AXES FOR CURVE2<1 DEGREE,NO CURVE GENERATION
E CHARACTER HEIGHT=0,NO PLOT POSSIBLE
E CHARACTER WIDTH=0,NO PLOT POSSIBLE
E DISTANCE BETWEEN POINTS IS GREATER THAN ARCLENGTH SPECIFIED IN ARC2
E EMPTY APPLIED TO UNDEFINED OBJECT
E EMPTY APPLIED TO UNDEFINED OBJECT( NO COLLECTION,SPACE OR BODY)
E IN BODY NO OR MORE THAN 60 SUBOBJECTS SPECIFIED
E IN COLLECTION NO OR MORE THAN 60 SUBOBJECTS SPECIFIED
E INVALID ARGUMENT PASSED TO OPERATION
E INVALID ARGUMENT PASSED TO OPERATION .....
E INVALID ARGUMENT PASSED TO POLY-OPERATION
E INVALID ARGUMENT PASSED TO POLY2-OPERATION
E INVALID ARGUMENT PASSED TO SPACE OPERATION
E INVALID CENTER OBJECT PASSED TO SCALE/ROTATE IGNORED
E INVALID DESTINATION OBJECT PASSED TO MOVE
E INVALID DESTINATION OBJECT PASSED TO MOVETO
E INVALID FIRST ARGUMENT PASSED TO INTERSECTION
E INVALID OBJECT ( NO COLLECTION) PASSED TO CARDCOL
E INVALID OBJECT (SPACE) PASSED TO COORD
E INVALID OBJECT (SPACE) PASSED TO SET COORD
E INVALID OBJECT PASSED TO BODY
E INVALID OBJECT PASSED TO COORD
E INVALID OBJECT PASSED TO FILL
E INVALID OBJECT PASSED TO INTERSECTION
E INVALID OBJECT PASSED TO SETCOORD
E INVALID OBJECT( NO COLLECTION) PASSED TO CARDCOL
E INVALID OBJECT( NO GIPSY-OBJECT) PASSED TO SET COORD
E INVALID OBJECT(BODY) PASSED TO SET COORD
E INVALID OBJECT(COLL) PASSED TO SET COORD
E INVALID SECOND ARGUMENT PASSED TO INTERSECTION
E LENGTH OF AXIS 0 IN FUNCTION AXIS2
E LENGTHS OF FIELDS ARE DIFFERENT
```

```
E MAXIMUM X POSITION OF PLOT FRAME=....>PAPER WIDTH,PLOTTING TERMINATED
E MAXIMUM Y POSITION OF PLOT FRAME=....>PAPER HEIGHT,PLOTTING TERMINATED
E MORE VALUES SPECIFIED IN STATEMENT FILL THAN FIELD LENGTH
E N=..... OUT OF RANGE IN FUNCTION NCOLL
E NO CIRCLE POSSIBLE THROUGH IDENTICAL POINTS
E NO PLANE POSSIBLE THROUGH IDENTICAL POINTS
E PLOT DIAGRAM STATEMENT OBSOLETE,USE CURVE2 INSTEAD
E POINT NUMBER .....>MAX POINT NUMBER ..... OF POLY IN SETCOORD
E POINT NUMBER .....>MAX POINT NUMBER ..... OF POLY2 IN SETCORRD
E POINTS SPECIFIED ON STRAIGHT LINE
E POLY CONTAINS LESS THAN ..... POINTS IN COORD
E POLY FOR CIRCLE AXIS HAS LESS THAN TWO POINTS
E POLY FOR CONE AXIS HAS LESS THAN 2 POINTS
E POLY FOR CYLINDER AXIS HAS LESS THAN 2 POINTS
E POLY FOR PLANE NORMAL HAS LESS THAN 2 POINTS
E POLY SPECIFIED CONTAINS NO POINTS
E POLY2 CONTAINS LESS THAN ..... POINTS IN COORD
E PROJECTION DIRECTION PARALLEL TO PROJECTION PLANE
E PROJECTION OF SURFACES IS NOT POSSIBLE IN REDUCE
E SELECT NOT SPECIFIED IN COORD(POLY),1 ASSUMED
E SELECT NOT SPECIFIED IN COORD(POLY2) 1 ASSUMED
E SELECT NOT SPECIFIED IN SETCOORD (POLY),1 ASSUMED
E SELECT NOT SPECIFIED IN SETCOORD (POLY2),1 ASSUMED
E SINGLE SURFACES CANNOT BE PLOTTED
E SOURCE OF ASSIGNMENT CONTAINS MORE POINTS IN POLY THAN TARGET
E SOURCE OF ASSIGNMENT CONTAINS MORE POINTS IN POLY2 THAN TARGET
E SOURCE OF ASSIGNMENT CONTAINS MORE SPACE SURFACES THAN TARGET
E SPACE ELEMENT EMPTY IN ...
E SPECIFIED DIN-FORMAT <1 OR >8 , 4 ASSUMED
E SPECIFIED INDEX > NUMBER OF POINTS IN POLY
E SURFACE NAME=.....
E TEXT PLANE POINT ON TEXT AXIS
E UNDEFINED FIRST ARGUMENT PASSED TO MOVE
E UNDEFINED FIRST ARGUMENT PASSED TO MOVETO
E UNDEFINED OBJECT PASSED TO ...
E UNDEFINED OBJECT PASSED TO .....
E UNDEFINED OBJECT PASSED TO ARC2
E UNDEFINED OBJECT PASSED TO ASSIGNMENT
E UNDEFINED OBJECT PASSED TO EDIT
E UNDEFINED OBJECT PASSED TO INTERSECTION
E UNDEFINED OBJECT PASSED TO NCOLL
E UNDEFINED OBJECT RECEIVED DURING SPACE ANALYSIS
E UNDEFINED SECOND ARGUMENT OF INTERSECTION DOES NOT MATCH POLY ARGUMENT
E UNINITIALIZED OBJECT PASSED TO ASSIGNMENT
E UNINITIALIZED OBJECT PASSED TO MOVE
E UNINITIALIZED OBJECT PASSED TO MOVETO

E INVALID OBJECT PASSED TO SET COORD
E OBJECTTYPES IN ASSIGNMENT ARE DIFFERENT
E OPEN PLOT MISSING,DIN A(4) ASSUMED
E UNDEFINED OBJECT PASSED TO PLANE
E UNDEFINED SECOND ARGUMENT PASSED TO MOVE
E UNDEFINED SECOND ARGUMENT PASSED TO MOVETO

E UNDEFINED OBJECT PASSED TO AXIS2
E UNDEFINED OBJECT PASSED TO BODY OPERATION
E UNDEFINED OBJECT PASSED TO CARDCOL
E UNDEFINED OBJECT PASSED TO COLL OPERATION
E UNDEFINED OBJECT PASSED TO FILL
```

```
E UNDEFINED OBJECT PASSED TO LINE.NO SHADE POSSIBLE.DISTANCE VALUE ....
E UNDEFINED OBJECT PASSED TO PLOT OPERATION
E UNDEFINED OBJECT PASSED TO PLOT/SAVE OPERATION
E UNDEFINED OBJECT PASSED TO PRINT
E UNDEFINED OBJECT PASSED TO REDUCE
E UNDEFINED OBJECT PASSED TO SPACE ANALYSIS
E UNDEFINED OBJECT PASSED TO SPACE OPERATION
E UNDEFINED OBJECT PASSED TO TRANSFORMATION
E UNINITIALIZED OBJECT PASSED TO .....
E UNINITIALIZED OBJECT PASSED TO ASSIGNMENT
E UNINITIALIZED OBJECT PASSED TO BODY OPERATION
E UNINITIALIZED OBJECT PASSED TO COLLECTION OPERATION
E UNINITIALIZED OBJECT PASSED TO PLOT OPERATION
E UNINITIALIZED OBJECT PASSED TO PLOT/SAVE OPERATION
E UNINITIALIZED OBJECT PASSED TO REDUCE
E UNINITIALIZED OBJECT PASSED TO SPACE OPERATION
E UNINITIALIZED OBJECT PASSED TO TRANSFORMATION
E X AND Y FIELDS FOR CURVE2 HAVE DIFFERENT LENGTH
E X VALUE FOR LOG CURVE<=0,NO CURVE GENERATION
E Y VALUE FOR LOG CURVE<=0,NO CURVE GENERATION
S ERROR IN GIPSY OBJECT STRUCTURE.OBJECT NOT PRINTABLE.CALL SYSTEM PROGR
S MAX VALUE FOR LOG AXIS<=0, =..... IN AXIS2
S MIN VALUE FOR LOG AXIS<=0  =..... IN AXIS2
S ORTHONORMAL BASE CANNOT BE DETERMINED FOR NULLVECTOR
```

10.3 Fehlermeldungen am Terminal 4014 bei der Plotfile-Interpretation

Es folgt eine alphabetische Liste der Fehlermeldungen mit Erläuterungen.

COMMAND NOT IN SYSTEM

Falsches Kommando

CONVERSION RAISED FOR SIZE CONSTANT

Bei Eingabe der Größe eines Windows wurde eine nicht als Zahl interpretierbare Zeichenfolge festgestellt. Kommando wiederholen.

ENTER DASHED, THROUGH OR OMITTED

Ungültiger oder fehlender Parameter nach INVISIBLE.

ENDILE RAISED ON INPUT PLOTFILE

Bei PICTURE NEXT, SKIP oder NR wurde das Ende des Plotfiles erreicht. Es muß danach durch PICTURE NR n eine gültige Bildnummer ausgewählt werden.

ENTER MAX OR SIZE

Nach VIEWPORT falscher oder fehlender Parameter

ENTER MAX, PLOTFILE OR SIZE

Nach Window falscher oder fehlender Parameter

HEADER 1 MISSING

Der Plotfile enthält am Start keine gültige File-Information. Die Barbeitung wird abgebrochen.

INVALID DATA ON PLOTFILE

Der Plotfile enthält nicht interpretierbare Information. Die Bearbeitung wird abgebrochen.

LINE <2 OR >21

Ungültige Menüfeldhöhe.

MORE THAN 12 GAPS + DASHES IN LINE DEFN

Der Plotfile enthält eine Linientyp-Definition mit mehr als 12 Strichen und Lücken. Die restlichen Striche und Lücken werden durch PFTEKT nicht mehr beachtet. (In GIPSY gibt es höchstens 2 Striche und Lücken, für strichpunktierte Linien $\frac{}{1}\ \frac{\cdot}{2}\ \frac{}{1}\ \frac{:}{2}$).

NO RASTERGRAPHICS ON THIS DEVICE

Der Plotfile enthält statt Liniengraphik Rastergraphik. Die Bearbeitung wird abgebrochen.

PARAMETERS MISSING, REENTER

Nach einem Kommando fehlen nötige Parameter oder aber Parameter sind falsch. Nach REENTER müssen die fehlenden Parameter einzeln eingegeben werden oder die Kommandoabarbeitung muß durch Ausrufezeichen unterbrochen werden.

PICTURE HEADER MISSING

Auf dem Plotfile fehlt eine Bildinformation. Die Bearbeitung wird abgebrochen.

SEGMENT NAME LONGER THAN 32 CHARS

Der Name eines graphischen Objektes war länger als 32 Zeichen (GIPSY-Namen sind stets ≤ 32 Zeichen lang).

10.4 Fehlermeldungen des Plotfile-Interpretierers für Plotter-Ausgabe

```
 1    FILE HEADER MISSING OR WRONG
 2    NO RASTERGRAPHICS WITH THIS PROGRAM POSSIBLE
 3    PICTURE HEADER 1 MISSING OR WRONG
 4    RASTERGRAPHICS RECORD NOT PLOTTED
 5    VERSION NUMBER>1 NOT SUPPORTED BY THIS PROGRAM
11    VIEWPORT HEIGHT>PAPER HEIGHT
12    VIEWPORT WIDTH>PAPER WIDTH
21    VIEW SPECIFIED BUT BREITEVIEW, HOEHEVIEW MISSING
22    MASST SPECIFIED BUT HOEHEVIEW MISSING
23    UPPER RIGHT OF WINDOW<=LOWER LEFT
```

Literatur

/ 1 / Schuster, R.: System und Sprache zur Behandlung graphischer Information im rechnergestützten Entwurf (GIPSY), KFK 2305, 1976

/ 2 / Schlechtendahl, E.G.: Das integrierte CAD-System REGENT, KFK-CAD 2, 1975

/ 3 / Schlechtendahl, E.G.: Grundzüge des integrierten CAD-Systems REGENT, Angewandte Informatik 18(1976) 11, S.490-496

/ 4 / Enderle, G.: Problemorientierte Sprachen im REGENT-System, Angewandte Informatik 18(1976) 12, S.543-549

/ 5 / Leinemann, K.: Dynamische Datenstrukturen des integrierten CAD-Systems REGENT, Angewandte Informatik 19 (1977) 1, S.26-32

/ 6 / Schuster, R.: GIPSY-Graphische Fähigkeiten als Bestandteil eines Systems für den rechnergestützten Entwurf, Angewandte Informatik 19(1977) 4, S.155-167

/ 7 / Schlechtendahl, E.G., Bechler, K.-H., Enderle, G., Leinemann, K., Olbrich, W.: REGENT-Handbuch, KFK 2666 und KFK-CAD 71, 1978

/ 8 / Schlechtendahl, E.G., Enderle, G., Leinemann, K., Schuster, R.: Das Programmsystem REGENT im praktischen Einsatz.
GI-Fachtagung: Methoden der Informatik für Rechnergestütztes Entwerfen und Konstruieren, München, 19.-21.Oktober 1977

/ 9 / Enderle, G.: Definition, Übersetzung und Anwendung benutzerorientierter Sprachen im CAD-System REGENT, KFK 2204, 1975

/ 10 / Enderle, G., Giese, I., Krause, M., Meinzer, H.P.:
AGF-Plotfile. Eine Datei zum Speichern und Transportieren graphischer Information.
KFK 2776, 1979