KfK 4440
Juli 1988

Conversion of the Bravo3 CAD*I Preprocessor to Version 3.2

Ch. Qiu
Institut für Reaktorentwicklung

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Reaktorentwicklung

KfK 4440

Conversion of the Bravo3 CAD*I Preprocessor to Version 3.2

Qui Changhua

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Zusammenfassung

Umstellung des Bravo3 CAD*I Preprocessors auf Version 3.2

Das ESPRIT-Projekt 322: "CAD Interface" (CAD*I) begann 1984 mit der Definition der wichtigsten CAD/CAM-Schnittstellen für den Datenaustausch zu Datenbanken, Finite-Elemente- Analyse und neuen Modellierungstechniken. Das Gesamtprojekt hat eine geplante Laufzeit von 5 Jahren (1984-1989). 12 Organisationen aus 6 europäischen Ländern sind am Projekt beteiligt. Die Forschungs- und Entwicklungsarbeiten des Projektes sind in 8 Arbeitsgruppen organisiert. Dieser Bericht erläutert den Beitrag des Autors zur Arbeitsgruppe 2 (Volumenmodelle) während der Zeit seiner Delegation an das Kernforschungszentrum Karlsruhe seit dem 1. April 1987.

Abstract

Conversion of the Bravo3 CAD*I Preprocessor to Version 3.2

The ESPRIT project 322: "CAD Interfaces" (CAD*I) has been established in 1984 to define the most important interfaces in CAD/CAM systems for data exchange, data base, finite element analysis, and advanced modelling. The total scheduled project duration is 5 years (1984-1989). There are 12 participants in the ESPRIT project CAD*I, coming from 6 European countries. The research and development work within the CAD*I project is organized in 8 working groups. This article summarizes the author's contribution to working group 2 of the project (solid modelling) during his assignment to the Kernforschungszentrum Karlsruhe starting April 1, 1987.

TABLE OF CONTENTS

1. The Tasks 1

2. The architecture of the CAD*I preprocessor for BRAVO3 1

3. Essential enhancements of version 3.2 compared to version 2.1 . . . 1

4. STATE OF PRE-PROCESSOR DEVELOPMENT IN BRAVO3 6
4.1 Parallelepiped (identifier = BOX) 6
4.2 Polyhedron (identifier = ARB) 8
4.3 Right Angle Wedge (identifer = RAW) 10
4.4 Sphere (identifier = SPH) 11
4.5 Right Circular Cylinder (identifier = RCC) 12
4.6 Truncated Right Cone (identifier = TRC) 14
4.7 Solid Torus (identifier = STO) 14
4.8 Arbitrary Slab (identifier = ASL) 16
4.9 Surface of revolution (identifier = REV) 18

5. STRUCTURE OF THE CAD*I NEUTRAL FILE FOR CSG 20

6. REFERENCES 22

7. Acknowledgement 22

Appendix A. PRE-PROCESSOR SOFTWARE 23

Appendix B. The test part 35

Appendix C. The Synthavision file of The test part 36

Appendix D. The CAD*I neutral file of The test part 37

1. THE TASKS

The task of the pre-processor is to produce a neutral file from the information in a CAD system data base. In this case, the CAD system is BRAVO3 from APPLICON, which is the standard CAD system for solid modelling applications at KfK.Input to the preprocessor are the synthavision files which come from BRAVO3. The major tasks were:

1. Revising the BRAVO3 pre-processor based on "Specification of CAD*I Neutral File for Solid, Version 3.2". The old pre-processor was created based on "Specification of a CAD*I Neutral File for Solid, Version 2.1" /1/.

2. Developing new programs for the pre-processor which were required due to the wider scope of version 3.2 /2/ as compared to version 2.1.

3. Creating the test parts in the CAD system "Applicon BRAVO3", and then testing the pre- and post-processor and the Neutral File Format with the test parts.

2. THE ARCHITECTURE OF THE CAD*I PREPROCESSOR FOR BRAVO3

There are two implementation configurations for the BRAVO3 pre-processor: The first configuration is completely on VAX as shown in Figure 1 on page 2; The second one is mixed on VAX and IBM, as shown in Figure 2 on page 3.

3. ESSENTIAL ENHANCEMENTS OF VERSION 3.2 COMPARED TO VERSION 2.1

"Specification of a CAD*I Neutral File for CAD Geometry ,Version 3.2" is different (with respect to solid models) from Version 2.1 in several respects. Some typical differences are:

1. The attribute list of a primitive entity:

   - The placement information i.e. the description of the local coordination system of the primitive:

     In version 2.1, the ENTI <PLACEMENT> of a solid primitive is specified as:

     ENTI <placement> ::=GENERIC(TYPE OF DIM)
     STRUCTURE

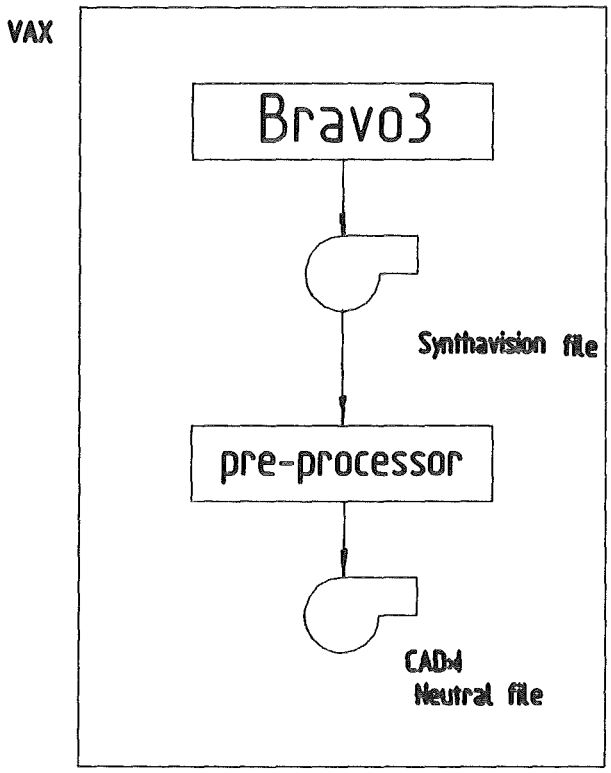


Figure 1. Software configuration in VAX

```
                        rotation    :  ROTATION(TYPE);
                        translation :  POINT(TYPE);
                      END;
ATTR  ROTATION = GENERIC(TYPE OF DIM)
                      CLASS(ROT_MATRIX(TYPE),
                            ROT_GLOBAL(TYPE),
                            ROT_AXIS);
ATTR ROT_MATRIX = GENERIC (TYPE OF DIM)
                      STRUCTURE
                       dirx  : DIRECTION(TYPE);
                       diry  : DIRECTION(TYPE);
                       dirz  : DIRECTION(TYPE);
                    END;
ATTR ROT_GLOBAL = GENERIC (TYPE OF DIM)
                      STRUCTURE
                        anglex  :  ANY(REAL);
                        angley  :  ANY(REAL);
                        anglez  :  ANY(REAL);
                     END;
ATTR ROT_AXIS    =  STRUCTURE
                        point   :  ANY(POINT(D3));
                        axis    :  ANY(DIRECTION(D3));
```

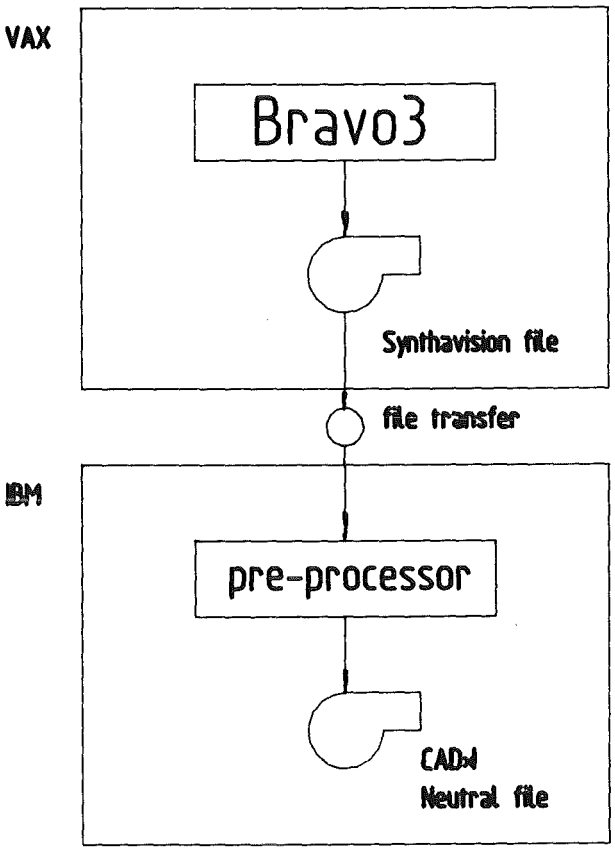


Figure 2. Alternative VAX and IBM

```
                        angle    :  ANY(REAL);
                      END;
ATTR DIRECTION = GENERIC (TYPE OF DIM)
                     STRUCTURE
                        x  :  ANY(RTEAL);
                        y  :  ANY(RTEAL);
                        z  :  ANY(RTEAL);
                      END;
ATTR POINT = GENERIC (TYPE OF DIM)
               CLASS( POINT_CONSTANT(TYPE)
                      POINT_VARIABLE(TYPE);
ATTR POINT_CONSTANT = GENERIC (TYPE OF DIM)
                        STRUCTURE
                          x : REAL;
                          y : REAL;
                          z : REAL;
                        END;
ATTR POINT_VARIABLE = GENERIC (TYPE OF DIM)
                        STRUCTURE
                          x : ANY(REAL);
                          y : ANY(REAL);
```

```
        z : ANY(REAL);
    END;
```

In version 3.2 the following entities are no longer needed in CSG:

ROT_AXIS
ROT_GLOBAL
ROT_MATRIX
PLACEMENT

The placement infomation for the solid primitives is embedded in the primitives themselves and consists of the attributes:

origin,
dir_x ,
dir_xz.

where origin indicates the origin of the local coordinate system,
dir_x to defines the local x-axis,
dir_xz lying in the local xz-plane.

The local y-axis is calculated by dir_y = dir_x * dir_xz and subsequently the local z-axis by dir_z = dir_x * dir_y.

For primitives with rotational symmetry the direction attributes may be missing.

- The order of the attribute list: In version 3.2, the placement comes first and then the shape information. In version 2.1, the shape information comes first and then the placement.

For example, The sphere which is situated in (x=100.0, y=100.0, z=170.0), and its radius is 130.00.

In version 2.1 it is expressed as follows:

```
SOLID_SPHERE(#6:+1.30000E+02,
             PLACEMENT(:ROT_MATRIX(:DIRECTION(:+1.00000E+00,
                                              +0.00000E+00,
                                              +0.00000E+00:),
                                   DIRECTION(:+0.00000E+00,
                                              +1.00000E+00,
                                              +0.00000E+00:),
                                   DIRECTION(:+0.00000E+00,
                                              +0.00000E+00,
                                              +1.00000E+00:):),
                       POINT_CONSTANT(:+1.00000E+02,
                                      +1.00000E+02,
                                      +1.70000E+02:):));
```

In version 3.2 it is expressed as follows:

```
SOLID_SPHERE(#6: POINT(+1.00000E+02,
                       +1.00000E+02,
                       +1.70000E+02),
                 +1.30000E+02);
```

2. Attribute WORLD_HEADER is increased by two values:

```
angle_test_value;
curvature_test_value;
```

3. COMPONENT

A component is the elementary constituent of an assembly. In version 2.1, components have geometric models associated via name. In version 3.2, components have geometric models associated via a GEOMETRY_ASSOCIATION.

For example: An assembly is composed of 3 parts: #2, #3, #4. In version 2.1, it is presented as follows:

```
COMPONENT(#40: #2; #41: #3; #42: #4);
```

In version 3.2, it is presented as follows:

```
COMPONENT(#40:);
GEOMETRY_ASSOCIATION(#40, #2,);
COMPONENT(#41:);
GEOMETRY_ASSOCIATION(#41, #3,);
COMPONENT(#42:);
GEOMETRY_ASSOCIATION(#42, #4,);
```

4. INDEX_ENTRY.

In version 2.1, it is presented as follows:

```
INDEX_ENTRY("PART0001",#42; "PART0002",#43; "PART0003", #44);
```

In version 3.2, it is presented as follows:

```
INDEX_ENTRY( user_defined_name, name ); For example:
INDEX_ENTRY("PART0001",#42);
INDEX_ENTRY("PART0002",#43);
INDEX_ENTRY("PART0003",#44);
```

5. MATERIAL_PROPERTY_LIST.

In version 2.1, it is presented as follows:

```
MATERAL(2,(/#44,#45,#49/);4,(/#46,#47,#48/);7,(/#50,#52,#55/));
```

In version 3.2, the parts which are made with the same material are presented together. For example:

```
MATERAL(2,(#44,#45,#49));
MATERAL(4,(#46,#47,#48));
MATERAL(7,(#50,#52,#55));
```

4. STATE OF PRE-PROCESSOR DEVELOPMENT IN BRAVO3

The BRAVO3 CAD*I processor (pre- and post-processor) are now based on the Specification of a CAD*I Neutral File for CAD Geometry, Version 3.2. Due to the fact that BRAVO3 is a typical CSG modeler, the processors know only the CSG entities. The state of pre-processor development in BRAVO3 is illustrated in table 1:

We will now discuss this mapping in more detail.

4.1 PARALLELEPIPED (IDENTIFIER = BOX)

In BRAVO3 BOX is created typically as follows: Input origin point and three orthogonal length vectors.

On the synthavision file of the BOX will appear as:

	BOX		Vx	Vy	Vz	L1x	L1y	L1z
			L2x	L2y	L2z	L3x	L3y	L3z

Where V is one corner of the BOX; L1, L2, L3 are three orthogonal length vectors of the BOX.
The corresponding CAD*I specification is :

```
ENTITY BOX = STRUCTURE
             origin          : ANY(POINT(D3));
             dir_x           : ANY(DIRECTION(D3));
             dir_xz          : ANY(DIRECTION(D3));
             dimension_x     :ANY(REAL);
             dimension_y     :ANY(REAL);
             dimension_z     :ANY(REAL);
      END;
```

For example:

BRAVO3 shapes	Synthavision body library	CAD*I entities
BOX	BOX	BOX
CONE	TRC	TRUNCATED_CONE
CYLINDER	RCC	SOLID_CYLINDER
EXTRUSION	ASL	LINEAR_SWEEP
PRISM	ARB	SOLID_CYLINDERS
PYRAMID	ARB	SOLID_CYLINDERS
REVOLUTION	REV	ROTATIONAL_SWEEP
SPHERE	SPH	SOLID_SPHERE
SWEPT	SCU	no mapping
TUBE	TOR*ARB TOR-ARB	SOLID_CYLINDERS SOLID_TORUS
WEDGE	RAW	LINDER_SWEEP
ELLIPSOID	ELL	no mapping
Truncated Elliptic Cone	TEC	no mapping
SOLID TORUS	STO	SOLID_TORUS

TABLE1: MAP OF THE PRE-PROCESSING WITH BRAVO3 PROCESSORS

The data of a BOX on the synthavision file is as follows:

```
-1.9000E+01 -4.5000E+00 6.0000E+00 6.0000E+00 0.0000E+00 0.0000E+00
 0.0000E+00  9.0000E+00 0.0000R+00 0.0000E+00 0.0000E+00 8.0000E+00
```

The neutral file format of this BOX is as follows:

```
BOX( #3:POINT(-1.9000E+01,-4.5000E+00,+6.0000E+00),
        DIRECTION(6.0000E+00,            +0.0,       +0.0),
        DIRECTION(      +0.0,            +0.0,+8.0000E+00),
        +6.0000E+00,+9.0000E+00,+8.0000E+00;
```

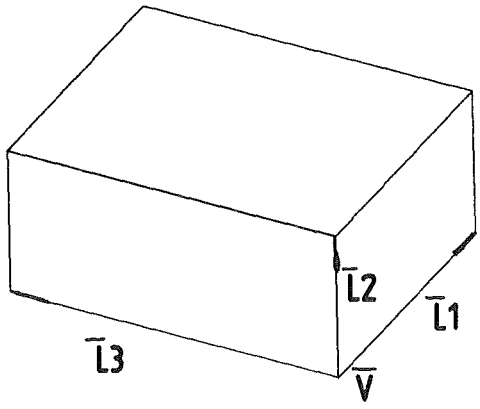


Figure 3. The BOX in Synthavision

4.2 POLYHEDRON (IDENTIFIER = ARB)

In BRAVO3 the polyhedron is created typically as follows: Input the coordinates of 8 (or 7, 6, 5, 4) vertex points.
On the synthavision file the Polyhedron will appear as :

	ARB		V1x	V1y	V1z	V2x	V2y	V2z
			V3x	V3y	V3z	V4x	V4y	V4z
			V5x	V5y	V5z	V6x	V6y	V6z
			V7x	V7y	V7z	V8x	V8y	V8z
			F1	F2	F3	F4	F5	F6

Where V1,V2,V3,V4,V5,V6,V7,V8 are 8 vertex points.They must always be supplied, but those not appearing in any face description, F, are ignored by the code. Unused vertex coordinates can be 0.0.

F1,F2,F3,F4,F5,F6 are the faces of the polyhedron. When the number of faces is less than 6, the remaining F's must be 0.0 and must appear at the end of the list. If a face has only three vertices, the last digit should be a repetition of the first one (e.g., 1231).

For example, the data of an ARB on the synthavision file is as follows:

```
+ARB
-5.0000E+01 0.0000E+00 0.0000E+00 -5.0000E+00 2.0000E+01 0.0000E+00
-3.0000E+01 0.0000E+00 0.0000E+00 -4.0000E+01 1.0000E+01 2.0000E+01
 0.0000E+00 0.0000E+00 0.0000E+00  0.0000E+00 0.0000E+00 0.0000E+00
 0.0000E+00 0.0000E+00 0.0000E+00  0.0000E+00 0.0000E+00 0.0000E+00
 0.1231E+04 0.1241E+04 0.2342E+04  0.3143E+04 0.0000E+00 0.0000E+00
```

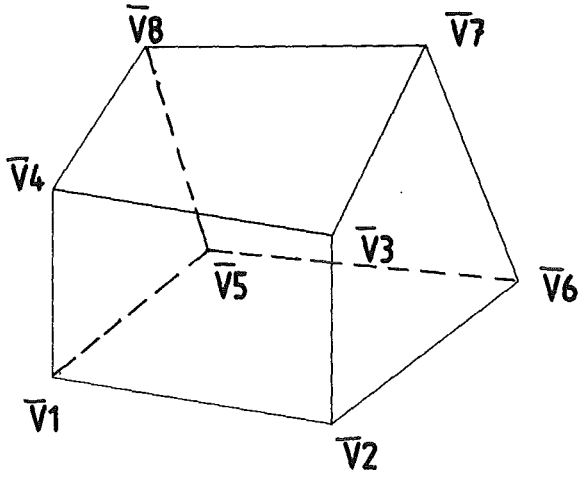


Figure 4. The polyhedron in Synthavision

The ARB is defined by the coordinates of 4 vertices points, and composed of 4 faces. Each face is composed of 3 vertices.

There is no entity on the CAD*I neutral file format for this type of polyhedron. We express it with cylinder intersections. Each cylinder is defined by using each face as a top face and the direction points to the center of the polyhedron.

For the above example, the neutral file format of the ARB is:

```
CONSTRUCT( #2:OPEN);
  SCOPE;
  SOLID_CYLINDER( #3:
          POINT(-4.0000E+01,+1.0000E+01,+2.0000E+01),
          DIRECTION(   +0.0,       +0.0,   +1.0000E+00),
          +3.8105E+01,+3.8105E+01);
  SOLID_CYLINDER( #4:
          POINT(-3.0000E+01,       +0.0,      +0.0),
          DIRECTION(+8.9443E-01    +0.0,-4.4721000E-01),
          +3.8105E+01,+3.8105E+01);
  SOLID_CYLINDER( #5:
          POINT(-5.0000E+01,       +0.0,       +0.0),
          DIRECTION(-7.0711E-01,,-7.0711E-01,     +0.0),
          +3.8105E+01,+3.8105E+01);
  SOLID_CYLINDER( #6:
          POINT(-5.0000E+01,+2.0000E+01,       +0.0),
          DIRECTION(       +0.0,+8.9443E-01,-.4721E-01),
          +3.8105E+01,+3.8105E+01);
  BOOLEAN( #7:INTERSECTION, #3, #4);
  BOOLEAN( #8:INTERSECTION, #7, #5);
  BOOLEAN( #9:INTERSECTION, #8, #6);
  END_SCOPE;
  CONSTRUCT_RESULT( #9);
CONSTRUCT( #2,CLOSE);
```

4.3 RIGHT ANGLE WEDGE (IDENTIFER = RAW)

The RAW input is identical to that for the BOX, with the restriction that the vectors forming the legs of the right triangle (L1, and L2) must be given first.

On the Synthavision file, the RAW will appear as:

	RAW		Vx	Vy	Vz	L1x	L1y	L1z
			L2x	L2y	L2z	L3x	L3y	L3z

There is no entity on the CAD*I neutral file format for RAW. So we express it as a LINEAR_SWEEP. Where vectors L1 and L2 form the face, and L3 as the depth of movement for the face. For example: The data of a RAW on the Synthavision file is as follows:

```
0.0000E+00 0.0000E+00 0.0000E+00 1.0000E+01 0.0000E+00 0.0000E+00
0.0000E+00 1.0000E+01 0.0000E+00 0.0000E+00 0.0000E+00 5.0000E+00
```

The Neutral File format is:

```
LINEAR_SWEEP( #3:OPEN);
```

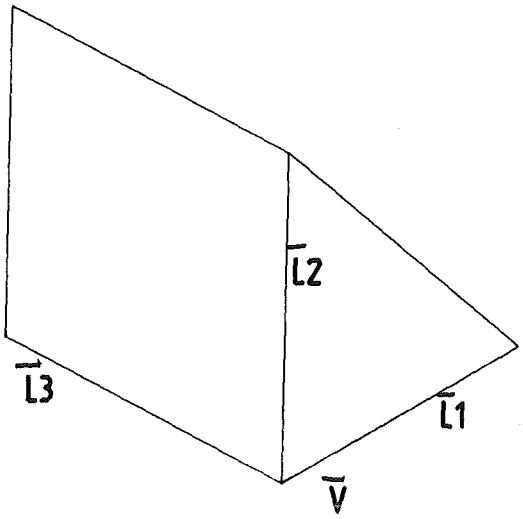


Figure 5. The right angle wedge in Synthavision

```
SCOPE;
POINT( #4:        +0.0,        +0.0,        +0.0);
POINT( #5:        +0.0,+1.0000E+01,         0.0);
POINT( #6:+1.0000E+01,        +0.0,         0.0);
POINT( #7:        +0.0,        +0.0,        +0.0);
DIRECTION(#8:         +0.0,+1.0000E+00,         +0.0);
LINE(#9:#4,#8);
CONTOUR_ELEMENT(#10:#9,#4,#5);
DIRECTION(#11:+7.0711E-01,-7.0711E-01,         +0.0);
LINE(#12:#5,#11);
CONTOUR_ELEMENT(#13:#12,#5,#6);
DIRECTION(#14:-1.0000E+00,        +0.0,         +0.0);
LINE(#15:#6,#14);
CONTOUR_ELEMENT(#16:#12,#6,#7);
END_SCOPE;
LINEAR_SWEEP_RESULT(((#10,,#13,#16)),
  +5.0000E+01,
  DIRECTION(        +0.0,       +0.00,+1.0000E+00);
  POINT(        +0.0,        +0.0,        +0.0),
  DIRECTION(+1.0000E+00,       +0.0,         +0.0);
  DIRECTION(        +0.0,       +0.0,+1.0000E+00);
LINEAR_SWEEP(#3,CLOSE);
```

4.4 SPHERE (IDENTIFIER = SPH)

In BRAVO3 the SPH is created typically as follows:

The SPH is defined by its center V, and radius R.

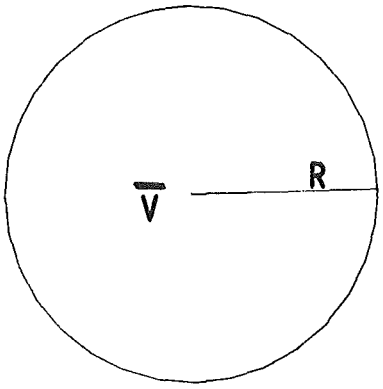


Figure 6. The sphere in Synthavision

On the Synthavision file, the SPH will appear as:

	SPH		Vx	Vy	Vz	R		

For example, The Sphere is located (x=10.0, y=12.0, z=12.0) and its radius is 120.0. The data of a SPH on the synthavision file is as follows

```
+SPH  1.0000E+01, 1.2000E+01, 1.2000E+01,1.2000E+02
```

The corresponding CAD*I neutral file specification is:

```
ENTITY SOLID_SPHERE = STRUCTURE
                        center : ANY(POINT(D3));
                        radius : ANY(REAL);
                      END;
```

The CAD*I neutral file format of the sphere is:

```
SOLID_SPHERE(#3: POINT( 1.0000E+01, 1.2000E+01, 1.2000E+01),
                 1.2000E+02);
```

4.5 RIGHT CIRCULAR CYLINDER (IDENTIFIER = RCC)

In RRAVO3 the RCC is created typically as follows:

A RCC is defined by the coordinates of the center of the base (V), the length vector (H) directed from V to the opposite base, and the radius(R).

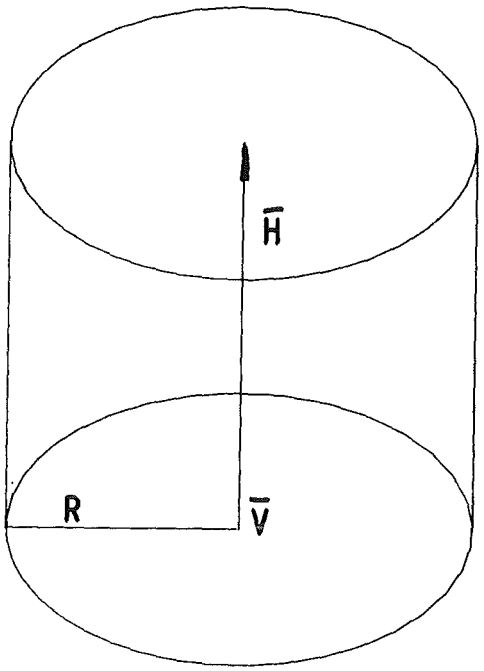


Figure 7. The cylinder in Synthavision

On the Synthavision file, the RCC will appear as:

	RCC		Vx	Vy	Vz	Hx	Hy	Hz
			R					

The corresponding CAD*I neutral file specification is:

```
ENTITY SOLID_CYLINDER = STRUCTURE
                          origin : ANY(POINT(D3));
                          dir_z  : ANY(DIRECTION(D3));
                          radius : ANY(REAL);
                          height : ANY(REAL);
                        END;
```

For example: A RCC' synthavision file is as follows:

```
+RCC 2.0000E+00 4.0000E+00 1.0000E+01 0.0000E+00-8.0000E+00 0.0000E+00
     4.0000E+00
```

The CAD*I neutral file of the RCC is as follows:

```
SOLID_CYLINDER(#3:POINT(+2.0000E+00,+4.0000E+00,+7.0000E+00),
                  DIRECTION(       +0.0,-8.0000E_01,   +0.0),
                  +4.0000E+00, +8.0000E+00);
```

4.6 TRUNCATED RIGHT CONE (IDENTIFIER = TRC)

A TRC is similar to an RCC, with the addition of a second radius, R2. It is required that V lies on the larger base (R1) and R1>R2 and R2>0.

On the Synthavision file, the TRC will appear as:

	TRC		Vx	Vy	Vz	Hx	Hy	Hz
			R1	R2				

The corresponding CAD*I neutral file specification is:

```
ENTITY TRUNCATED_CONE = STRUCTURE
                          origin  : ANY(POINT(D3));
                          dir_z   : ANY(DIRECTION(D3));
                          radius1 : ANY(REAL);
                          radius2 : ANY(REAL);
                          height  : ANY(REAL);
                        END;
```

For example, the data of a TRC on the synthavision file is as follows:

```
+TRC 2.0000E+00 4.0000E+00 1.0000E+01 0.0000E+00-8.0000E+00 0.0000E+00
     4.0000E+00 2.0000E+00
```

The CAD*I neutral file format of the TRC is as follows:

```
TRUNCATED_CONE(#3:POINT(+2.0000E+00,+4.0000E+00,+7.0000E+00),
                  DIRECTION(       +0.0,-8.0000E-01,   +0.0),
                  +4.0000E+00, +2.0000E+00 +8.0000E+00);
```

4.7 SOLID TORUS (IDENTIFIER = STO)

In BRAVO3 STO is created typically as follows:

The input is defined as follows:
V-- center
N-- a unit normal giving the direction of the axis.
R1--large radius
R2--small radius
R3--small radius
IF R2=R3, it has a circular cross section, and only this case is converted to the neutral file.

On the Synthavision file, the STO will appear as:

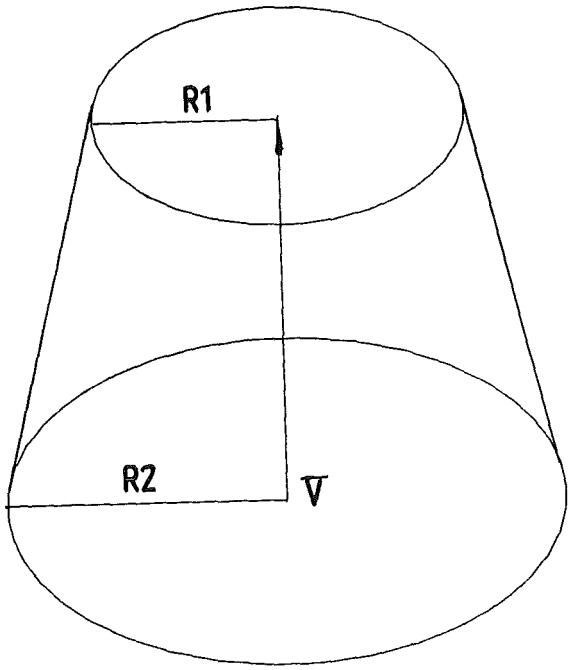


Figure 8. The cone in Synthavision

	STO		Vx	Vy	Vz	Nx	Ny	Nz
			R1	R2	R3			

The corresponding CAD*I neutral file specification is:

```
ENTITY SOLID_TORUS = STRUCTURE
                       origin       : ANY(POINT(D3));
                       dir_z        : ANY(DIRECTION(D3));
                       radius_large : ANY(REAL);
                       radius_small : ANY(REAL);
                     END;
```

For example, the data of a STO on the synthavision file is as follows:

```
2.5000E+01            0.0        0.0        0.0       0.0 -1.0000E+00
2.0000E+01 5.0000E+00 5.0000E+00
```

The CAD*I neutral file of the STO is as follows:

```
SOLID_TORUS(#23:
      POINT(+2.5000+01,     +0.0,     +0.0),
      DIRECTION(    0.0,    0.0, -1.0000E+00),
      +2.0000E+01,+5.0000E+00);
```

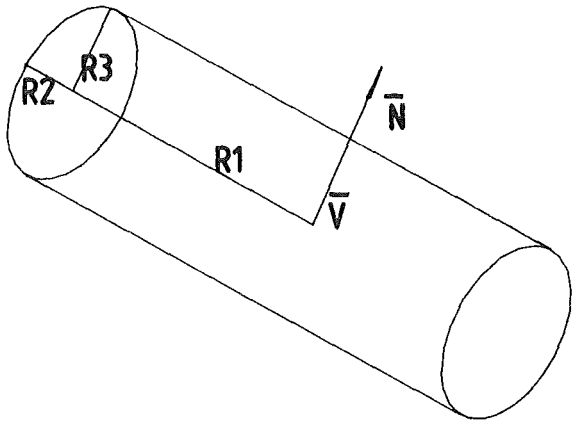


Figure 9. The torus in Synthavision

4.8 ARBITRARY SLAB (IDENTIFIER = ASL)

In BRAVO3 the ASL is created typically as follows:

The body is defined by a set of connected curve segments which form a closed curve in 2_dimensions.

On the synthavision file of the ASL will appear as:

Line1	ASL	Vx	Vy	Vz	T	N	-
Line2		Xx	Xy	Xz	Yx	Yy	Yz
Line3		X	Y	R	F	-	-

where V is the origin of the body. T is the thickness of the body. N is the number of points to be defined (equal to the number of curve segments). X, Y are two orthogonal unit vectors. Line3 is used to define each point on the curve. The coordinates (X,Y) are relative to the local body origin (V). R is the radius of curvature of the curve segment. F may be assumed as any value fmom 0 to 8.

The corresponding CAD*I specification is presented as a LINEAR_SWEEP as follows:

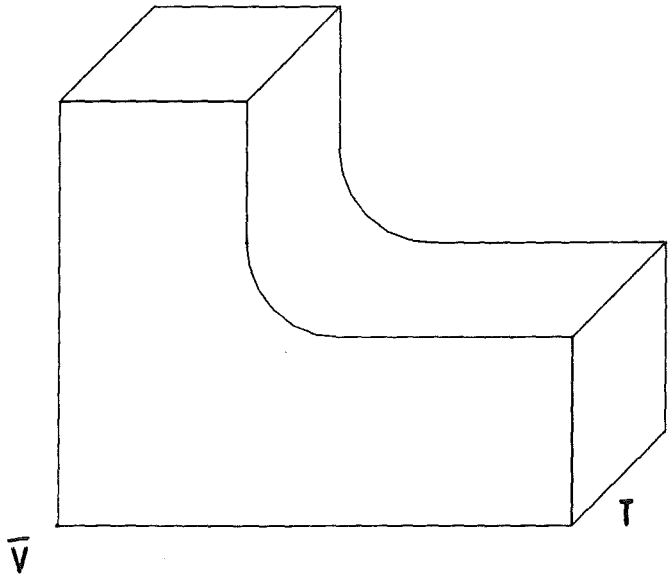


Figure 10. The arbituary slab in Synthavision

```
ENTITY LINEAR_SWEEP =
    STRUCTURE
          SCOPE;
                DIRECTION(D2);
                POINT(D2);
                ELEMENTARY_CURVE(D2);
                CONTOUR_ELEMENT;
          END_SCOPE;
    contour_sequence : LIST OF REF_ONLY(CONTOUR_ELEMENT);
    shift_length     : ANY(REAL);
    shift_direction  : ANY(DIRECTION(D3));
    origin           : ANY(POINT(D3));
    dir_x            : ANY(DIRECTION(D3));
    dir_xz           : ANY(DIRECTION(D3));
  END;
```

For example, the data of an ASL on the Sythavision file is as follows:

```
-3.2000E+02 6.6500E+02-1.2000E+02 2.4000E+02 4.0000E+00
 1.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 1.0000E+00 0.0000E+00
 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
 1.5000E+02 0.0000E+00 0.0000E+00 0.0000E+00
 0.0000E+00-5.0000E+01 0.0000E+00 0.0000E+00
 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
```

The CAD*I neutral file format of the body is presented as follows:

```
LINEAR_SWEEP(#83:OPEN);
SCOPE;
POINT(#84:       +0.0,      +0.0,      +0.0);
POINT(#85:+1.5000E+02,      +0.0,      +0.0);
POINT(#86:       +0.0,-5.000E+01,      +0.0);
POINT(#87:       +0.0,      +0.0,      +0.0);
DIRECTION(#88:+1.0000E+00,       +0.0,       +0.0 );
LINE(#89:#84,#88);
CONTOUR_ELEMENT(#90:#89,#84,#85);
DIRECTION(#91:-9.4868E-01,-3.1623E-01,       +0.0 );
LINE(#92:#85,#91);
CONTOUR_ELEMENT(#93:#92,#85,#86);
DIRECTION(#94:       +0.0,+1.0000E+00,       +0.0 );
LINE(#95:#86,#94);
CONTOUR_ELEMENT(#96:#95,#86,#87);
END_SCOPE
LINEAR_SWEEP_RESULT(((#90,93,96)),
  +2.4000E+02,
  DIRECTION(       +0.0,      +0.00,+1.0000E+00);
  POINT(-3.2000E+02,+6.6500E+02,-1.2000E+02),
  DIRECTION(+1.0000E+00,      +0.0,       +0.0);
  DIRECTION(       +0.0,      +0.0,+1.0000E+00);
  LINEAR_SWEEP(#83,CLOSE);
```

4.9 SURFACE OF REVOLUTION (IDENTIFIER = REV)

In BRAVO3 a REV is created typically as follows:

The body is defined by a set of connected curve segments which form a closed curve in 2-dimensions. The enclosed area is rotated about an axis to form a solid. The input format, given below, is identical to that for the arbitrary slab (ASL).

On the Synthavision file of the REV will appear as:

Line1	REV	Vx	Vy	Vz	T	N	-
Line2		Xx	Xy	Xz	Yx	Yy	Yz
Line3		X	Y	R	F	-	-

The corresponding CAD*I neutral file specification is:

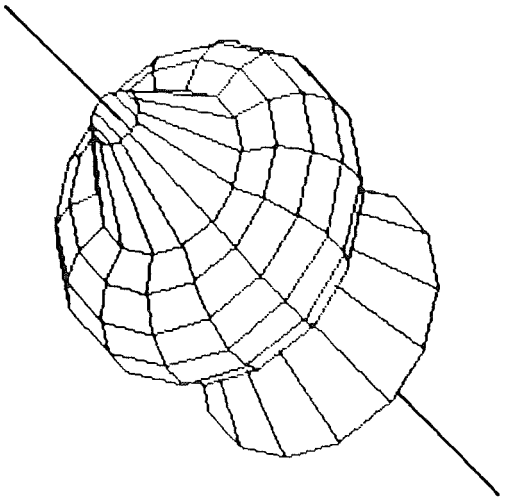


Figure 11. The surface of revolution in Synthavision

```
ENTITY ROTATIONAL SWEEP =
    STRUCTURE
          SCOPE;
                DIRECTION(D2);
                POINT(D2);
                ELEMENTARY_CURVE(D2);
                CONTOUR_ELEMENT;
          END_SCOPE;
    contour_sequence : LIST OF REF_ONLY(CONTOUR_ELEMENT);
    angle            : ANY(REAL);
    origin           : ANY(POINT(D3));
    dir_x            : ANY(DIRECTION(D3));
    dir_xz           : ANY(DIRECTION(D3));
  END;
```

For example, the data of a REV on the Synthavision file is as follows:

```
 5.4749E+01 5.8536E+01 3.0000E+01 0.0000E+00 3.0000E+00
-7.0711E-01-7.0711E-01 0.0000E+00-7.0711E-01-7.0711E-01 0.0000E+00
 1.0000E+01 2.0000E+01 1.0000E+01 6.0000E+00
 3.0000E+01 2.0000E+01 1.0000E+01 6.0000E+00
 1.0000E+10 2.0000E+01 0.0000E+00 0.0000E+00
```

The CAD*I neutral file format of a REV is as follows:

```
ROTATIONL_SWEEP(#3:OPEN);
SCOPE;
POINT(#4:+1.0000E+01,+2.0000E+01,        0.0);
POINT(#5:+3.0000E+01,+2.0000E+01,       +0.0);
POINT(#6:+1.0000E+01,+2.0000E+01,       +0.0);
CIRCLE(#7:+1.0000E+01,
  POINT(+2.0000E+01,+2.0000E+01,_8.1833E+05),
  DIRECTION(      +0.0,      +0.0,+1.0000E+00),
  POINT(+1.0000E+01,+2.0000E+01,       +0.0);
CONTOUR_ELEMENT(#8:#7,#4,#5);
CIRCLE(#9:+1.0000E+01,
  POINT(+2.0000E+01,+2.0000E+01,_8.1833E+05),
  DIRECTION(      +0.0,      +0.0,+1.0000E+00),
  POINT(+3.0000E+01,+2.0000E+01,       +0.0);
CONTOUR_ELEMENT(#10:#9,#5,#6);
END_SCOPE;
ROTATIONAL_SWEEP_RESULT(((#8,#10)),
  1.0472E+00,
  POINT(+5.4749E+01,+5.8536E+01,+3.0000E+01),
  DIRECTION(-7.0711E-01,-7.0711E-01,      +0.0);
  DIRECTION(       +0.0,      +0.0,-1.0000E+00);
ROTATION_SWEEP(#3,CLOSE);
```

5. STRUCTURE OF THE CAD*I NEUTRAL FILE FOR CSG

The structure of the neutral file for CSG is shown below in fig.12:

The file header is composed of 14 parameters. They are:

1 author, the person who generated the file
2 company, the full address
3 computer hardware on which the file was generated
4 operating system, full identification
5 pre-prrocessor identification including release number
6 data and time of pre-processing, format 'yyyymmddhhnnss'
7 disclaimer (optional, the null string is assumed as default)
  minimum schema levels required for post-preprocessor
8 geometry, allowed values: 0, 1, 2, 3a,3b, 3c, 4, 5
9 assembly, allowed values: 0, 3, 8
10 parametric, allowed values: 0, 1a, 1b, 2
11 referencing, allowed values: 0, 1, 2, 3
12 maximum number of digits for integers on the file
13 maximum number of significant digits in real mantissa
14 maximum number of digits for exponents

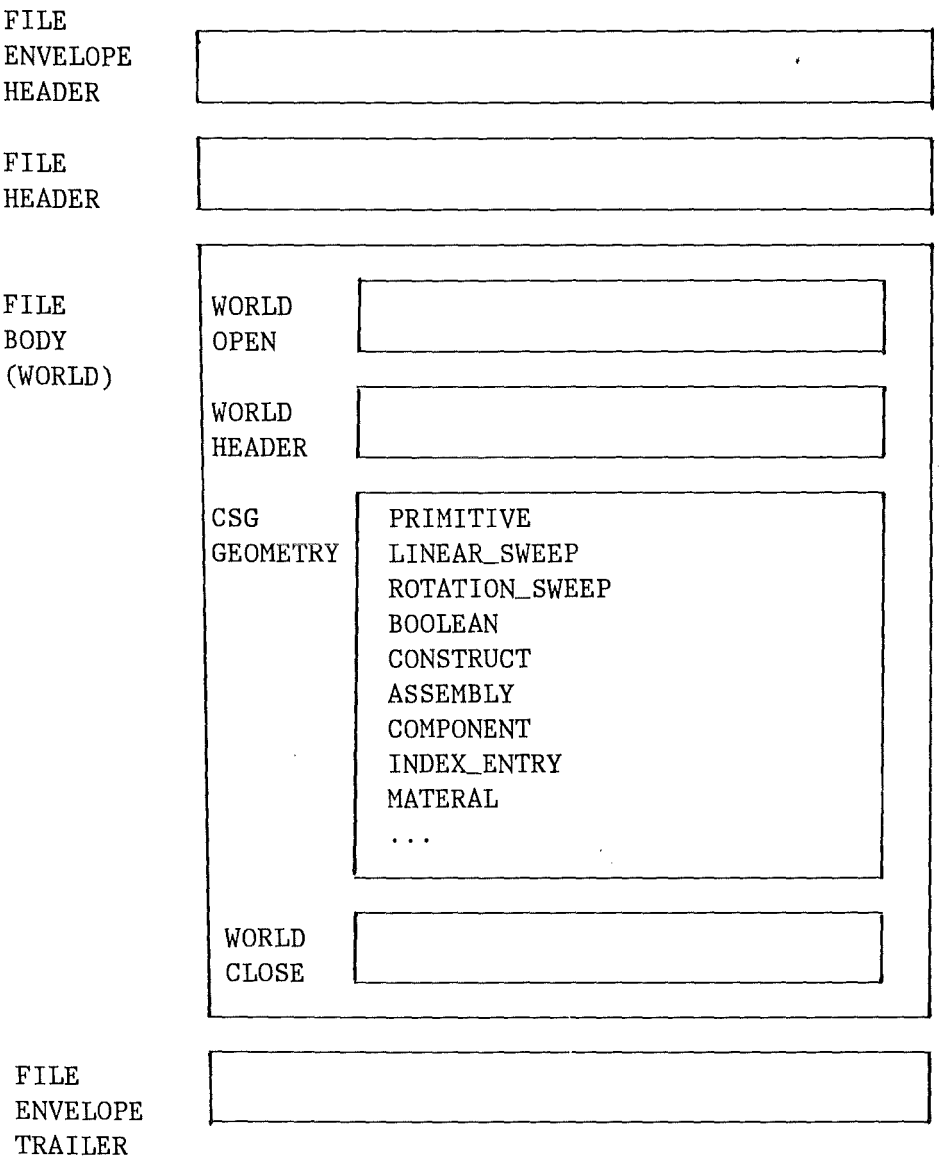


Figure 12. The structure of CAD*I neutral file for CSG

The file body is the main content of the neutral file. It is composed of worlds. The world is the highest level entity in the schema. Within its scope are all other entities which together define the contents of a CAD database. The neutral file mechanism transfers a world between different universes.

The world begins with WORLD(name:OPEN) and ends with WORLD(name,CLOSE).

The WORLD_HEADER is composed of 6 parameters:

1 the length unit given by the length_unit_factor.
2 the angle_unit such that a real value corresponds to angle_unit times one radian.

3 the world_size.
4 the distance_test_value.
5 the angle_test_value.
6 the curvature_test_value.

The primitives and booleans build the Constructive Solid Geometry (CSG) technique. This technique consists of describing complex structures by performing Boolean set operators (intersection, union, and difference). The elementary solid named 'primitives' are used as initial generators, amongst them boxes, cylinders, cones etc. In BRAVO3, sweeping (rotation, translation), polyhedron, etc are used also as 'primitives'. A CSG representative of a solid is based on a tree structure where non-terminal nodes represent operators and the leaves are the 'primitives'.

The assembly is composed of parts. It is represented with component, INDEX_ENTRY, and material.

The following example shows a CAD*I neutral file according to version 3.2 of specification written by the pre-processor of BRAVO3.

The test part is an assembly comlosed of 2 parts. One part is made from copper, the other from aluminium.

The test part is shown in APPENDIX B; The Synthavision file is shown in APPENDIX C; The CAD*I neutral file is shown in APPENDIX D.

6. REFERENCES

/1/ E.G.Schlechtendahl (Editor), Specification of a CAD*I Neutral File for CAD Geometry, Version 2.1 May 1986 , Springer
/2/ E.G.Schlechtendahl (Editor), Specification of a CAD*I Neutral File for CAD Geometry, Version 3.2 June 1987, Springer

7. ACKNOWLEDGEMENT

This work was performed under a grant provided by Friedrich-Ebert-Stiftung.

The auther gratefully acknowledges the technical support which he has received from M.F.Katz and Dr.E.G.Schlechtendahl while performing this work.

APPENDIX A. PRE-PROCESSOR SOFTWARE

Now we present in alphabetical order all routines of the CAD*I pre-processor for BRAVO3 which were converted from Version 2.1 to Version 3.2 or which were newly developed for Version 3.2. 43 routines in Version 2.1 are in Version 3.2 no longer needed. 13 COMMON blocks are used to share data between two or more program units. They are commented as follows:

BOOL: This COMMON block contains 4 parameters. They are: BFIRST, IBOOL, OPRTOR, OPRAND.

BOX: This COMMON contains 4 parameters: SBODAT, CBODAT, SRADAT, CRADAT. They are used for reading data from CARDI and LUI.

BUFFER: This COMMON area manage the buffer. It contains two COMMONs BUFFER1 and BUFFER2. The parameter in BUFFER1 is USED. The parameters in BUFFER2 are CARDI and CARD.

HOWTPR: This COMMON area controls the formatting. It contains a LOGICAL PRETTY and two parameters DPI and DPM. The PRETTY controls whether the file is to be formatted for easy to read. The parameter DPM presents the real numbers as 1PEw.d where d=DPM and w=DPM+7, DPI gives the format for integers.

INDEX: This COMMON contains two COMMONs CINDI,and CINDC. In CINDI contains two parameters INDLEN, and INDREF. They are the length and references of INDEX. CINDC contains the parameter UDN. It is USER_DEFINED_NAME .

INTPR: This COMMON area contains parameters INTDIG, SIGDIG, EXPDIG

INTDIG - Maximum number of digit for integer.
EXPDIG - Maximum number of digit for exponent.
SIGDIG - Maximum significant digit of real mantissa.

IOFILS: This COMMON area contains 6 logical units for the pre-processor input and output. They are:

LUI - The logical unit associated with the native data base or native sequential file.
LUE - The logical unit associated with the error and log file.
LUO - The logical unit associated with the neutral file to be generated.
LTI - The logical unit associated with the user directives.
LTO - The logical unit associated with the system prompts.
LST - The logical unit associated with the statistics output.

MATER: This COMMON contains 4 parameters MATTAB, MATLIS, LMATTA, and NUPART:

MATTAB - For 7 materials and 50 entries per material.
MATLIS - Material list.
LMATTA - Number of entries in a material list.
NUPART - Number of parts.

MISC: This COMMON area contains two COMMONs MISL,and MISI.The COMMON area MISL contains parameter TRACE, and the COMMON area MISI contains parameters INAME, NAMOPE, LASBOO, STLENG, INSASS, DIM, PMODE, WORNAM:

TRACE - Writes begin and end of a routine onto message file.
INAME - Currently used name.
NAMOPE - Name of open entity.
LASBOO - Last created boolean.
STLENG - String length for routine ADDSTR.
INSASS - Number of instances in the assembiy result.
DIM - Length of an array.
PMODE - Print mode.
WORNAM - Name of WORLD.

SPHERE: This COMMON area contains parameters SSPDAT and CSPDAT for reading data from CARDI and LUI for sphere.

STTICS: This COMMON area controls the the statistics. It contains a parameter NLINES which indicates the number of 80-character records actually written to the neutral file.

SWEEPA: This COMMON area contains 6 parameters. They are:

NASL - The actual number of linear sweeps in a model.
NRAW - The actual number of linear sweeps of RAW in a model.
NREV - The actual number of rotation sweeps in a model.
NPRIM - Number of primitives in a model.
ANGLE - Angle of rotation of the ROTATION_SWEEP.
ASLANG - True if a ASL is used to get the angle of a ROTATION_SWEEP.

WORLD: This COMMON area controls the world header attributes ,It contains 6 parameters. They are:

LENGUF - The length unit given by the length_unit_factor.
ANGLUN - The angle unit.
WORSIZ - The size of the world.
MINDIS - The minimal allowable distance value.
ANGTVE - The angle_test_value which indicates the allowable angle value between curves.
CURTVE - The curvature_test_value which indicates the allowable curvature difference between curves.

All the routines in the pre-processor are written in FORTRAN-77 and have been developed and classified into 5 levels as follows:

- Level 1 routines manage an 80 character internal buffer. They add simple tokens to the buffer. They are:

```
ADDINT( IVAL, ISTAT ) ;
ADDREA( RVAL, ISTAT );
ADDSTR( TEXT ,ILEN, MODE, ISTAT ).
```

  Level 1 routines are basic routines. For example: The routine ADDSTR is called in other 43 routines of all 75 routines in pre-processor. The routine PROASL(ISTAT) calls the routine ADDSTR 13 times.

- Level 2 routines use level 1 routines to build complete sequences of tokens separated by delimiters. The most important routines for level 2 are:

```
NAMINT(IVAL,ISTAT);
REFANY(IVAL,ISTAT);
READIM(RARR,IDIM,ISTAT);
REFDIM(IARR,IDIM,ISTAT);
SCOPE (MODE,ISTAT);
WRCMND(TEXT,ISTAT);
OPCLEN(KEYWRD,INAM,MODE,ISTAT);
MFBGEN(MODE,TEXT,ISTAT);
NFBGEN(MODE,TEXT,ISTAT);
```

- Level 3 routines employ level 1 and level 2 routines to build up a complete entity, property, or attribute definition. For example:

```
WDIREC(MODE,INAM,COOR1,COOR2,ISTAT);
WPOINT(PDATP,ISTAT);
et al.
```

- Level 4 routines mainly employ level 3 routines to build up a entity and property list. For example:

```
WLBOOL(ISTAT);
WINDEX(ISTAT);
et al.
```

- LEvel 5 ROUTINES are interface routines which read data from the Synthavision file in BRAVO3 and write entities onto neutral file. For example:

```
PROBOX(ISTAT);
PROPSM(ISTAT);
PRORAW(ISTAT);
PRORCC(ISTAT);
et al.
```

In the 5 levels, level 1 and level 2 are used for all conditions, level 3 and level 4 are used for CAD*I, and level 5 is used only for BRAVO3.

The routines which are developed into 5 levels have many advantages : They are convenient for managing, easy to reveise, have higher commonabilities and smaller volume, program development costs and time to reduce.

The routine EXA1 is the main program in BRAV03 pre-processor,and the others are subroutines.

The routines are presented in the following format:

- subroutine name
- header comment
- called subroutines
- commons

EXA1: To write the beginning and ending of a CAD*I Neutral File; To convert synthavision file to CAD*I format.

Subroutines: INTVAR;MFBGEN;NFBGEN;HEADER;MFBGEN; BWORLD;PAARCR;PAGROU;EWORLD;NFBGEN;FORCEO.
commons : MISC;IOFILS.

ADDINT: To write an integer to the output buffer.

Subroutines: BMVBLN;ADDSTR;ERROUT.
commons : MISC.

ADDREA: To write a real to the output buffer.

Subroutines: ADDSTR;ERROUT.
commons : MISC;WORLD;HOWTPR.

ADDSTR: To write a substring to the output buffer.

Subroutines: FORCEO;ERROUT.
commons : MISC;BUFFER;IOFILS.

ERROUT: To printout a message in LUE,and in additional LU's.

Subroutines: None.
commons : MISC;IOFILS.

FORCEO: To empty the buffer to LUO file.

Subroutines: TYPOUT;ERROUT.
commons : MISC;IOFILS;BUFFER;STTICS.

GETINT: To get an integer from unit LUTINP.

Subroutines: None.
commons : MISC;IOFILS.

INTVAR: To initiate files and common them.

Subroutines: None.
commons : ALL.

MFBGEN: To write metafile brackets.

Subroutines: ADDSTR.
commons : MISC.

NAMINT: To write a name(#inam) to the output buffer.

Subroutines: ADDSTR;REFANY;ERROUT.
commons : MISC;STTICS.

NFBGEN: rite neutral file brackets.

Subroutines: ADDSTR.
commons : MISC.

OPCLEN: To write a text(#inam:open);or text(#inam,close) to the CARD buffer.

Subroutines: ADDSTR;REFANY.
Commons : MISC;HOWTPR.

PROMPT: To write a prompting text to unit LUTOUT.

Subroutines: None.
Commons : MISC;IOFILS.

READIM: To write a sequence of reals to the output buffer(the reals are seperated by commas).

Subroutines: ADDREA;ADDSTR.
Commons : MISC;HOWTPR.

REFANY: To write a reference(#inam) to the output buffer.

Subroutines: RMVBLN;ADDSTR;ERROUT.
Commons : MISC;HOWTPR.

REFDIM: To write a sequence of references to the output buffer(the reference are separated by commas).

Subroutines: REFANY;ADDSTR.
Commons : MISC.

RMCRLF: To substitute CR/LF characters to blank in a text.

Subroutines: None.

Commons : MISC.

RMVBLN: To remove leading and trailing blanks from a string.

Subroutines: None.
Commons : MISC.

SCOPE: To write a scope or end_scope to the neutral file.

Subroutines: ADDSTR.
Commons : MISC;HOWTPR.

TYPOUT: To type out a string as a new line in file LU.

Subroutines: ADDSTR.
Commons : MISC.

VECOPE: 6 subroutines to perform vector operations(VECADD;VECSUB; VECDOT;VECCRO;VECLEN;VECUNI).

Subroutines: None.
Commons : None.

VECADD: Add two vectors.

Subroutines: None.
Commons : HOWTPR;MISC.

VECSUB: Subtract two vectors.

Subroutines: None.
Commons : HOWTPR;MISC.

VECDOT: Dot product of two vectors.

Subroutines: None.
Commons : HOWTPR;MISC.

VECCRO: Cross product of two vectors.

Subroutines: None.
Commons : HOWTPR;MISC.

VECLEN: Lenth of a vector.

Subroutines: None.
Commons : HOWTPR;MISC.

VECUNI: Normalize a vector.

Subroutines: None.

Commons : HOWTPR;MISC.

WRCMNT: To write a comment to the output buffer.

Subroutines: ADDSTR;RMVBLN.
Commons : HOWTPR;MISC.

ARBDIR: Direction of a cylinder.

Subroutines: None.
Commons : HOWTPR;MISC.

ARCORE: To gather the arcs of the revolution(rot.sweeps).

Subroutines: RMVBLN;PROPAR;GETANG.
Commons : HOWTPR;MISC;PARAME;IOFILS; BUFFER;BOOL;SWEEPA;STTICS.

BASSE: To write the assembly open ,name and scope onto the neutral file

Subroutines: OPCLEN;WINDEN;SCOPE.
Commons : HOWTPR;MISC;INDEX;BUFFER.

BPART: To write the part open ,name and scope onto the neutral file.

Subroutines: PCLEN;WINDEN;SCOPE.
Commons : HOWTPR;MISC;INDEX;BUFFER.

BWORLD: To write end_scope and world end onto the neutral file.

Subroutines: ADDSTR;WINDEN;READIM;OPCLEN;TERREA;SCOPE.
Commons : HOWTPR;MISC;BUFFER;WORLD;INDEX.

DIRDIR: To write dir_x and dir_xz as the attributes onto the neutral file.

Subroutines: ADDSTR;WINDEN;READIM.
Commons : HOWTPR;MISC;BOX;BUFFER;STATIS.

EASSE: To write the assembly close onto the neutral file.

Subroutines: ADDSTR;WINDEN;READIM.
Commons : HOWTPR;MISC;BUFFER.

EPART: To write the part close onto the neutral file.

Subroutines: ADDSTR;POCLEN.
Commons : HOWTPR;MISC;BUFFER.

EWORLD: To write end_scope and world end onto the neutral file.

Subroutines: ADDSTR;POCLEN.

Commons : HOWTPR;MISC;BUFFER.

FIBOFA: To find box points on a face containing a closed polygon with 3 or 4 points and 1 to 4 points not in the face.

Subroutines: VECSUB;VECDOT;VECLEN.
Commons : MISC;IOFILS.

FILLBO: To fill the boolean list of the open construct.

Subroutines: None.
Commons : MISC;BOOL;BUFFER.

GETANG: To read the data of a synthavision sweep entity,and use it to get the rotational angle of a revolution entity.

Subroutines: None.
Commons : MISC;BOX;BUFFER;IOFILS;HOWTPR;STTICS.

GETCEN: To get the center of aN ARC defined by two points,radius and a form number,z_values of points are assumed to be 0.

Subroutines: None.
Commons : MISC;BOX.

HEADER: To write the header of a neutral file on to the output buffer.

Subroutines: WINDEN;ADDSTR;PROMPT;TERTEX;TERINT.
Commons : MISC;HOWTPR;IOFILS.

NORMAL: To find the normal direction of two vectors.

Subroutines: None.
Commons : MISC;HOWTPR.

PAARCR: To read cards from sythavision file;to look for linear sweeps that define angles of rotation sweeps.lists are filled with angles and states of the sweeps.

Subroutines: ARCORE.
Commons : MISC;PARAME;IOFILS;BUFFER; STTICS;SWEEPA.

PAASS: To process assembly input.

Subroutines: EASSE;REFANY;ADDSTR;BASSE.
Commons : MISC;HOWTPR;IOFILS;BUFFER; STTICS;INDEX.

PAGROU: To read card from the synthavision file.if statements part, assembly or group are found control is given to one of the sub-routines for further processing.

Subroutines: PAPART;PAASS;WLCOMPWINDEX;WMATER.
Commons : MISC;PARAME;IOFILS;BUFFER; STTICS;SWEEPA.

PAPART: To process part input.

Subroutines: BPART;PROBOX;PROSPH;PROPAR;PROROS;
PRORCC;PROTRC;PROTOR;PROPSM;PPOASL;
WLBOOL;SCOPE;ADDSTR;REFANY;EPART.
Commons : MISC;BOOL;IOFILS;BUFFER;STTICS;HOWTPR.

PROASL: Read the data of a synthavision ASL entity. Make an entry to the boolean list; Write a linear_sweep entity onto the neutral file.

Subroutines: FILLBO;OPCLEN;SETLOS;WLPOIN;WLDIRE;
WLLINE;GETCEN;WLCIR2;WLCELE;ADDSTR; REFA-
NY;ADDREA;NORMAL;READIM.
Commons : MISC;PARAME;IOFILS;BUFFER; BOX;HOWTPR;SWEEPA.

PROBOX: Read the data of a synthavision BOX entity. Make an entry to the boolean list; Write a box entity onto the neutral file.

Subroutines: FILLBO;OPCLEN;NAMINT;VECUNI;VECCRO; VEC-
DOT;VECLEN;WDIREC;ADDSTR;READIM.
Commons : MISC;BOX;IOFILS;BUFFER;HOWTPR.

PROMPT: Write a promting text to unit LUTOUT: Read a whole line of text from unit LUTINP; Look if first letter of the input is Y: Set YES = . TRUE .

Subroutines: RMVBLN;RMCRLF.
Commons : MISC;IOFILS.

PROPAR: Write a part reference to the boolean list.

Subroutines: REFANY.
Commons : MISC;IOFILS;BOX;BOOL;BUFFER;INDEX.

PROPSM: Read the data of a synthavision ARB entity; Make an entry to the boolean list; Write sylinder entities onto the neutral file.

Subroutines: ADDSTR;NAMINT;FILLBO; WPOINT;WDIREC;READIM.
Commons : MISC;IOFILS;BOX;ARB;BUFFER; PAR-
AME;STATIS;HOWTPR;SWEEPA.

PRORAW: Read the data of a synthavision RAW entity; Make an entry to the boolean list; Write a linear_sweep entity onto the neutral file.

Subroutines: OPCLEN;FILLBO;SCOPE;WLPOIN;READIM;
ADDSTR;REFANY;WLCELE.
Commons : MISC;IOFILS;BOX;BOOL;BUFFER; HOWTPR;SWEEPA.

PRORCC: Read the data of a synthavision right circular cylinder entity; Make an entry to the boolean list; Write a cylinder entity onto the neutral file.

Subroutines: ADDSTR;NAMINT;FILLBO; WPOINT;WDIREC;READIM.
Commons : MISC;IOFILS;BOX;BOOL;BUFFER;HOWTPR.

PROROS: Read the data of a synthavision REV entity; Make an entry to the boolean list; Write a rotational_sweep entity onto the neutral file.

Subroutines: FILLBO;OPCLEN;SETLOS;WLPOIN;WLDIRE; WLLINE;GETCEN;WLCIR2;WLCELE;ADDSTR; REFA-NY;ADDREA;NORMAL;READIM.
Commons : MISC;PARAME;IOFILS;BUFFER; BOX;HOWTPR;SWEEPA.

PROSPH: Read the data of a synthavision SPHERE entity; Make an entry to the boolean list; Write a sphere entity onto the neutral file.

Subroutines: ADDSTR;NAMINT;FILLBO; WPOINT;READIM.
Commons : MISC;IOFILS;BOX;BOOL;BUFFER;HOWTPR.

PROTOR: Read the data of a synthavision TORUS tity; Make an entry to the boolean list; Write a SOLID_TORUS entity onto the neutral file.

Subroutines: ADDSTR;NAMINT;FILLBO; WPOINT;READIM;WDIREC.
Commons : MISC;IOFILS;BOXL;BUFFER;HOWTPR.

SETLOS: Order the list of points to get anti_clockwise 'sence' of the loop that will refer their edges.

Subroutines: None.
Commons : MISC.

TERINT: Write a prompting text in unit LUTOUT; Read a whole line of text from unit LUTINP; Clear the text from leading and trailing blanks; Echo the augmented text on unit LUST; Put leading and trailing single quotation marks as well as a trailing comma,and output them to CARD buffer accordingt to the value of pretty.

Subroutines: ADDINT;TYPOUT;RMCRLF;RMVBLN.
Commons : MISC;IOFILS;HOWTPR.

TERREA: Write a prompting text in unit LUTOUT; Read a whole line of text from unit LUTINP; Clear the text from leading and trailing blanks; Echo the augmented text on unit LUST; Put leading and trailing single quotation marks as well as a trailing comma,and output them to CARD buffer accorting to the value of pretty.

Subroutines: ADDREA;TYPOUT;RMCRLF;RMVBLN.
Commons : MISC;IOFILS;HOWTPR.

TERTEX: Write a prompting text in unit LUTOUT; Read a whole line of text from unit LUTINP; Clear the text from leading and trailing blanks; Echo the augmented text on unit LUST; Put leading and trailing single quotation marks as well as a trailing comma,and output them to CARD buffer accorting to the value of pretty.

Subroutines: ADDREA;TYPOUT;RMCRLF;RMVBLN.
Commons : MISC;IOFILS;HOWTPR.

WDIREC: Write dir_x and dir_xz as the attributes onto neutral file.

Subroutines:ADDSTR;READIM.
Commons :MISC;BUFFER;HOWTPR;STATIS;BOX.

WINDEN: Write a blank string at the begin of the buffer.

Subroutines: ADDSTR.
Commons : MISC;BUFFER.

WINDEX: Write an index_entry_property list onto the neutral file.

subroutines: ADDSTR;RMVBLN;REFANY.
Commons : MISC;BUFFER;INDEX;HOWTPR.

WLBOOL: Write a BOOL entity or a list of booleans onto the neutral file.

Subroutines: ADDSTR;NAMINT;REFANY.
Commons : MISC;BUFFER;BOOL;HOWTPR.

WLCELE: Write a contour element or a list of contours to the neutral file.

Subroutines: ADDSTR;NAMINT;REFANY;WINDEN.
Commons : MISC;HOWTPR.

WLCIRC: Write a circle entity or a list of circles to the neutral file.

Subroutines: ADDSTR;NAMINT;WDIREC;WINDEN;WLPOIN.
Commons : MISC;HOWTPR.

WLCIR2: Write a circle entity or a list of circles to the neutral file.

Subroutines: ADDSTR;NAMINT;WDIREC;WINDEN;WLPOIN;ADDREA.
Commons : MISC;HOWTPR.

WLCOMP: Write a component and geomtry association onto the neutral file.

Subroutines: ADDSTR;NAMINT;REFANY.
Commons : MISC;HOWTPR;BUFFER;INDEX.

WLDIRC: ,Write a direction as an attribute onto the neutral file.

Subroutines: ADDSTR;READIM.
Commons : MISC;HOWTPR;BUFFER;BOX;STATIS.

WLDIRE: Write a direction or a list of directions onto the neutral file.

Subroutines: ADDSTR;READIM;NAMINT.
Commons : MISC;HOWTPR.

WLLINE: Write a line entity or a list of lines onto the neutral file.

Subroutines: ADDSTR;REFANY;NAMINT.
Commons : MISC;HOWTPR.

WLPOIN: Write a point or a list of points onto the neutral file.

Subroutines: ADDSTR;READIM;NAMINT.
Commons : MISC;HOWTPR.

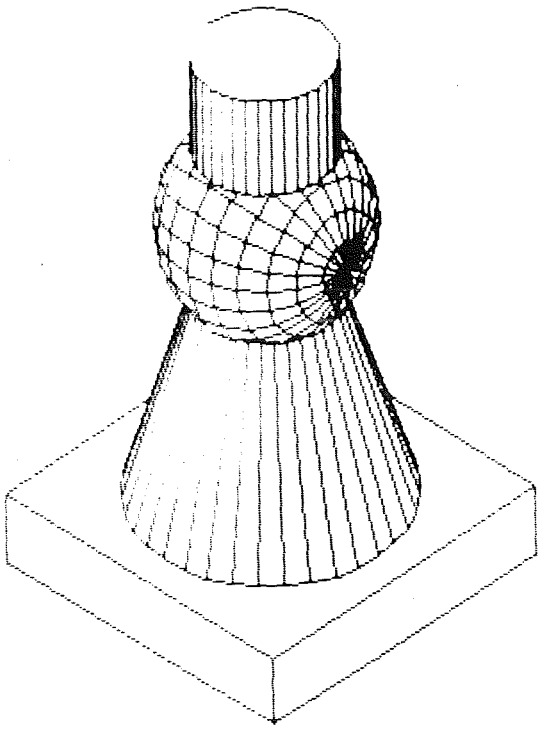
WMATER: Write a material_property list onto the neutral file.

Subroutines: ADDSTR;ADDINT;REFANY.
Commons : MISC;HOWTPR;BUFFER;MATER;INDEX.

WPOINT: Write a point constant as an attribute onto the neutral file.

Subroutines: ADDSTR;READIM;REFANY;WINDEN.
Commons : MISC;HOWTPR.

APPENDIX B. THE TEST PART



APPENDIX C. THE SYNTHAVISION FILE OF THE TEST PART

```
SHADLINE
COMMANDS
COMP2     1  490
LANG
  1    DATACAME          766  555            2.6110E-01 0.0000E+00
  1    MOVECAME                              2.6082E+03 2.6082E+03 2.6082E+03
  1    TURNCAME                              0.0000E+00 0.0000E+00 0.0000E+00
  1    VERT                                 -4.0825E-01 8.1650E-01-4.0825E-01
  1    DATALIGH                              5.7735E-01 5.7735E-01 5.7735E-01
  2    CUT
FINISHED
GEOMETRY
PART0001
+RCC        2.0000E+01 2.5000E+01 0.0000E+00 0.0000E+00-1.5000E+01 0.0000E+00
            1.0000E+01
+SPH        2.0000E+01 0.0000E+00 0.0000E+00 1.5000E+01
+TRC        2.0000E+01-4.0000E+01 0.0000E+00 0.0000E+00 3.0000E+01 0.0000E+00
            2.0000E+01 1.0000E+01
 END
PART0002
+BOX       -5.0000E+00-5.0100E+01-2.5000E+01 0.0000E+00 0.0000E+00 5.0000E+01
            5.0000E+01 0.0000E+00 0.0000E+00 0.0000E+00 1.0000E+01 0.0000E+00
 END
FINISHED
GROUPS
ASSE0001
 PART0001
 PART0002
 END
FINISHED
```

APPENDIX D. THE CAD*I NEUTRAL FILE OF THE TEST PART

```
AD*I_FORMAT_BEGIN_19851011   METAFILE
AD*I_FORMAT_BEGIN_19860611   CAD*I TP1
    HEADER(
        (*  AUTHOR  :  *)    'F.KATZ',
        (*  ADDRESS :  *)    'IRE/KFK',
        (*  HARDWARE:  *)    'IBM3090',
        (*  OP. SYS.:  *)    'IBM3090-MVS/XA',
        (*  PRE_PROC:  *)    'SYNTHAVISION 1.1',
        (*  DATE    :  *)    'YYYYMMDD',
        (*  DISCLAIM:  *)    ,
        (*  LEV_GEOM:  *)    '3A',
        (*  LEV_ASSE:  *)    '3',
        (*  LEV_PARA:  *)    '1A',
        (*  LEV_REF :  *)    '1',
        (*  MAX_INT :  *)    3,
        (*  MANTISSE:  *)    4,
        (*  EXPONENT:  *)    2);
    WORLD( #1:OPEN);
        WORLD_HEADER(
        (*  LENG.UN.:  *)    +1.0000E-03,
        (*  ANGL.UN :  *)    +1.0000E+00,
        (*  WORLD.S.:  *)    +1.0000E+05,
        (*  MINDIST :  *)    +1.0000E-04,
        (*  AN.TS.V.:  *)    +1.0000E-04,
        (*  CU.TS.V.:  *)    +1.0000E-04);
        SCOPE;
    CONSTRUCT( #2:OPEN);
        SCOPE;
        SOLID_CYLINDER( #3:
                POINT(+2.0000E+01,+2.5000E+01,        +0.0),
                DIRECTION(        +0.0,-1.5000E+01,        +0.0),
                +1.0000E+01,+1.5000E+01);
        SOLID_SPHERE( #4:
                POINT(+2.0000E+01,        +0.0,        +0.0),+1.5000E+01);
        TRUNCATED_CONE( #5:

                POINT(+2.0000E+01,-4.0000E+01,        +0.0),

                DIRECTION(        +0.0,+3.0000E+01,        +0.0),
                +2.0000E+01,+1.0000E+01,+3.0000E+01);
        BOOLEAN( #6:UNION, #3, #4);
        BOOLEAN( #7:UNION, #6, #5);
        END_SCOPE;
        CONSTRUCT_RESULT( #7);
    CONSTRUCT( #2,CLOSE);
    CONSTRUCT( #8:OPEN);
        SCOPE;
        BOX( #9:
```

```
                POINT(-5.0000E+00,-5.0100E+01,-2.5000E+01),
                DIRECTION(        +0.0,        +0.0,+1.0000E+00),
                DIRECTION(        +0.0,+1.0000E+00,        +0.0),
                +5.0000E+01,+5.0000E+01,+1.0000E+01);
        END_SCOPE;
        CONSTRUCT_RESULT( #9);
    CONSTRUCT( #8,CLOSE);
    COMPONENT(#10:);
    GEOMETRY_ASSOCIATION(#10, #2,);
    COMPONENT(#11:);
    GEOMETRY_ASSOCIATION(#11, #8,);
    ASSEMBLY(#12:OPEN);

        SCOPE;
        END_SCOPE;
        ASSEMBLY_RESULT((#10,#11));
    ASSEMBLY(#12,CLOSE);
    INDEX_ENTRY("PART0001",#10);
    INDEX_ENTRY("PART0002",#11);
    INDEX_ENTRY("ASSE0001",#12);
    MATERIAL(  2,());
    MATERIAL(  4,(#10));
    MATERIAL(  7,(#11));
    END_SCOPE;
    WORLD( #1,CLOSE);
AD*I_FORMAT___END_19860611   CAD*I TP1
AD*I_FORMAT___END_19851011   METAFILE
```