Contents

Page

1 Introduction 1
1.1 STEP and PDES 1
1.2 MRSEVs 1

2 Background 3
2.1 Conceptual Model of Manufacturing 3
2.2 Process Plans 4
2.3 Machining Process Plan 5
2.4 NC Process Information Model 6
2.5 Form Features Information Model 6
2.6 Tolerance Information Model 7
2.7 MRSEV Technical Issues 7

3 A Library of MRSEVs for 3-Axis Machining 9
3.1 Nature of MRSEVs in Library 9
3.2 Description Methods 9
3.3 Profiles 10
3.3.1 General_Profile 10
3.3.2 Other Types of Profile 10
3.4 Hierarchy Figures and Conventions 11
3.5 The Main Hierarchy 11
3.6 Islands 14
3.7 Groups and Copy Methods 15
3.7.1 Patterns 15
3.7.2 Replication 17
3.8 Details of Mrsev_Volumes 19
3.8.1 Linear_Sweep Pockets 19
3.8.1.1 Rectangular_Pocket_No_Islands 19
3.8.1.2 Other_Pocket_With_Islands 21
3.8.2 Hole 21
3.8.3 Grooves 24
3.8.3.1 Standard_Groove Cross Sections 24
3.8.3.2 Profile 25
3.8.3.3 Forming the Groove 25
3.8.3.4 Standard_Groove Offsets 25
3.8.3.5 Other_Grooves 26
3.8.3.6 Groove Examples 26
3.8.4 Thread 29
3.8.5 Edge_Cut 31
3.8.6 Rotation_Pocket 37
3.8.6.1 Vertical_rotation_pocket 37
3.8.6.2 Horizontal_Rotation_Pocket 37

Page

3.8.7 Islands 39
3.8.7.1 Anti_Groove_Island 40
3.8.7.2 Horizontal_Rotation_Island 41
3.8.7.3 Vertical_Rotation_Island 41
3.8.8 Ramp 45
3.8.8.1 Straight_Ramp 45
3.8.8.2 Circular_Ramp 45

Appendix A: References 48

Appendix B: Prototype EXPRESS Schema for MRSEVs.....50

Appendix C: An Example of Using MRSEVs 57

Figures

Page

Figure 1. Manufacturing Model for Machining 3
Figure 2. Main Hierarchy of MRSEVs in Library 13
Figure 3. Island Hierarchy 14
Figure 4. MRSEV Copy Methods 15
Figure 5. Groups, Patterns, and Replications 18
Figure 6. Rectangular_Pocket_No_Islands 20
Figure 7. Other_Pocket_With_Islands 22
Figure 8. Holes 23
Figure 9. Standard Groove Cross Sections 24
Figure 10. Rectangular_Groove 27
Figure 11. Other_Groove 28
Figure 12. Thread 30
Figure 13. Ends of Edge_Flat Differ 32
Figure 14. Block with Edge_Flat 34
Figure 15. Pocket with Edge_Round 35
Figure 16. Edge_Round with Cutter 36
Figure 17. Vertical_Rotation_Pocket 38
Figure 18. Horizontal_Rotation_Pocket 39
Figure 19. Anti_Groove_Island 42
Figure 20. Horizontal_Rotation_Island 43
Figure 21. Vertical_Rotation_Island 44
Figure 22. Straight_Ramp 46
Figure 23. Circular_Ramp 47
Figure B1. Hierarchy of MRSEVs in Prototype 52
Figure B2. Hierarchy of FFIM Entities from Figure B1 53
Figure C1. Design of the "Clevis2" Part 58
Figure C2. Process Plan for Second Cut on Clevis2 59
Figure C3. Portion of MRSEV File for Clevis2 60
Figure C4. An Other Pocket MRSEV 61
Figure C5. Two More MRSEVs 61

A LIBRARY OF MATERIAL REMOVAL SHAPE ELEMENT VOLUMES (MRSEVs)

Thomas R. Kramer

Guest Researcher, NIST &

Research Associate, Catholic University

NISTIR 4809

March 26, 1992

Funding for the work described in this paper was provided to Catholic University by the National Institute of Standards and Technology under cooperative agreement Number 70NANB9H0923.

1 Introduction

1.1 STEP and PDES

STEP is the emerging international Standard for the Exchange of Product Model Data being developed under the auspices of the International Organization for Standardization (ISO).

PDES is the acronym for Product Data Exchange using STEP. It is the U. S. activity in support of STEP. PDES is being furthered in the United States by several organizations. The IGES/PDES Organization (IPO) is a voluntary standards organization which participates in developing the STEP standards. Two of the many IPO/ISO committees helping develop STEP are the Form Features Committee and the Manufacturing Technology Committee. PDES, Inc. is a consortium of major companies which has the goal of accelerating the implementation of PDES. The National Institute of Standards and Technology (NIST) has established a National PDES Testbed. The Testbed is funded by the Computer-aided Acquisition and Logistic Support (CALS) program of the Office of the Secretary of Defense.

Data descriptions are handled in STEP by defining an information model in the EXPRESS data modeling language [Schenck90, Spiby91] for each type of data that is needed. Once an information model is in place, data representing a specific product may be put into a computer or on paper according to the rules in STEP Part 21 for mapping EXPRESS to a STEP physical file [Altemueller88a, Altemueller88b, VanMaanen91]. The data section of a STEP physical file normally consists wholly of instances of data entities of the sort made available by the EXPRESS model.

1.2 MRSEVs

In machining a metal part, it is sometimes feasible to use features on the design of the part to determine what material to machine away in order to make the part from a piece of raw stock or a partially machined workpiece. Very often, however, it will be desirable to machine the workpiece so that it has temporary features which do not appear on the design, or which do not correspond exactly to design features. To handle this situation, it is useful to have a method for describing the shape of material to be removed by machining operations. Shapes of volumes to be removed are generated during process planning and used for NC-programming. The need for these material removal volume shapes was cited in a document, *NC Process Information Model* [Cain89], prepared for consideration in the IPO Manufacturing Technology Committee. The term “Material Removal Shape Element Volumes” (MRSEVs) was defined in that document to refer to such shapes. That term, MRSEV, will be used in this paper.

The IPO Form Features Committee has built a Form Features Information Model (FFIM) that provides for expressing volumes in a wide variety of shapes. The entities defined in the FFIM may be used as resources in defining a library of MRSEVs.

Consideration has been given to MRSEVs (by other names, before the term was coined) in the NIST PDES Testbed and its predecessors. Issues regarding MRSEVs are listed in section 2 and discussed in a separate paper [Kramer92a]. A library of proposed MRSEVs was prepared in 1988 by the author and Mr. Mark Unger, a former NIST researcher. A current version of the library is described in section 3. A subset of the library was formalized in an EXPRESS schema in 1989. The EXPRESS schema for the subset draws heavily on the features defined in the FFIM by means of the EXPRESS "map" directive, which has since been deleted from EXPRESS. The schema is defined and described in Appendix B. It is obsolete because of changes in EXPRESS and the FFIM but could be updated if the opportunity and need arise.

The subset of the library given by the MRSEV schema is being used in the NC-program generation system called the Off-Line Programming System (OLPS) [Kramer91a, Kramer92b]. The MRSEVs in the schema are used for passing shape information to OLPS in conjunction with process plans for machining. How this is done is described in Appendix C, which also gives sample data.

Process planning issues, while important, are not the focus of this paper, and will be discussed here only as they relate to MRSEVs.

2 Background

2.1 Conceptual Model of Manufacturing

The library of MRSEVs described here should be understood in the context of a particular conceptual view of manufacturing by machining. An activity and data flow model of the view taken here (excluding shop floor activities which would occur after the NC-program is written) is shown in Figure 1. The model is specific to cutting piece parts, in that it includes a workpiece, a fixture, and a setup. The model is not specific to any particular type of cutting, and could apply equally well to machining, turning, and most other types of cutting found in machine shops.

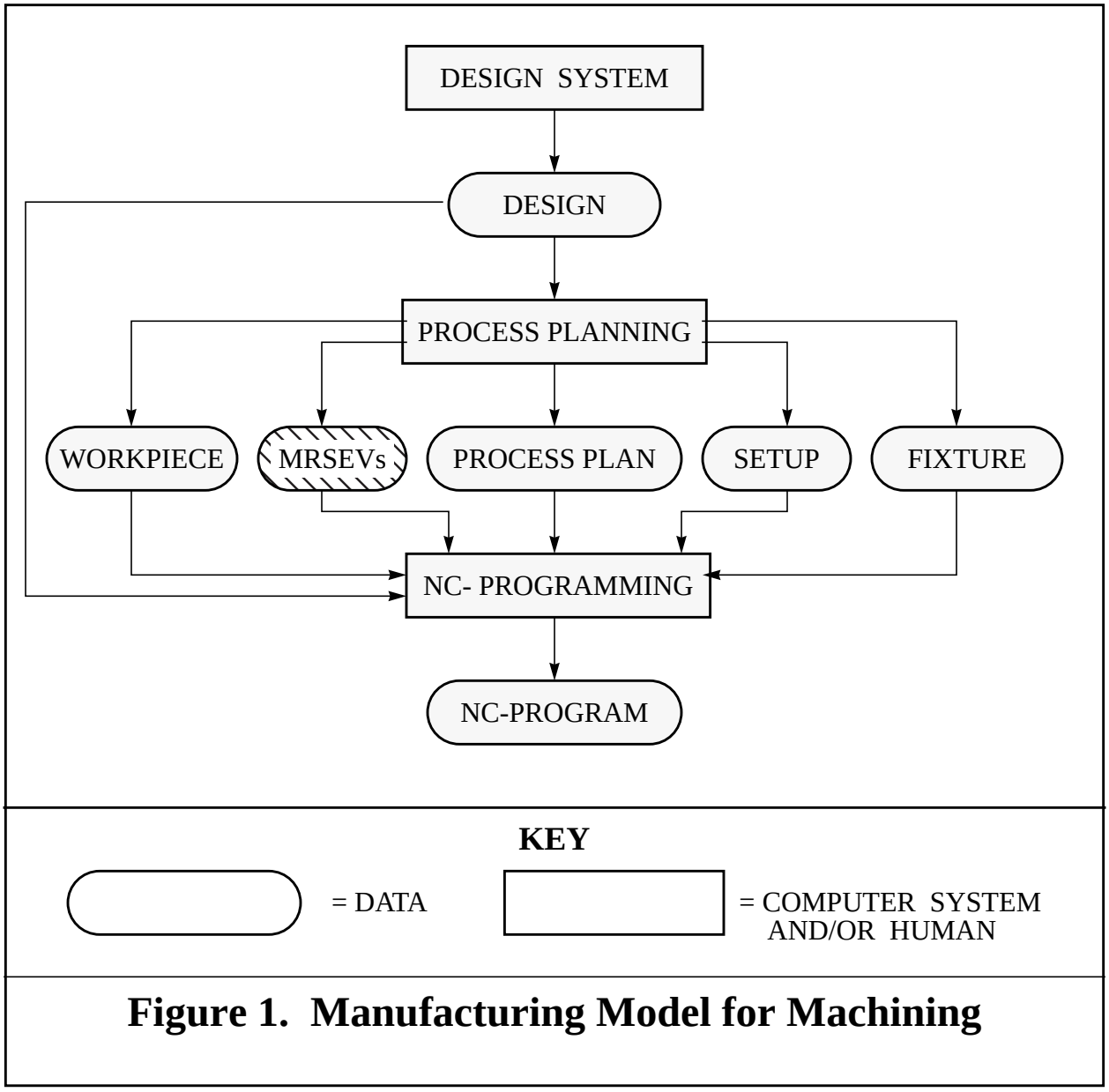


Figure 1. Manufacturing Model for Machining

In the model, a design system is used to produce a design. A design is taken to include specification of nominal shape as well as tolerances and surface finishes. The nominal shape may be expressed using a surfaced wire frame, a boundary representation, constructive solid geometry, form features, or even some other representation method. It may be useful to express the design in form features as the first step in process planning, regardless of how it comes out of the design system.

The process planning system takes in the design and puts out a machining process plan, a workpiece description, a set of MRSEVs, a setup description, and a fixture description. The setup description specifies where the workpiece, design, fixture and MRSEVs are with respect to a global coordinate system, which might be the coordinate system used to express the NC-program.

The NC-programming system takes in all of the output of process planning as well as the original design and produces an NC-program.

If more than one machine or more than one fixturing is required to make a part, then process planning will produce more than one set of output data, and NC-programming will have to make an NC-program for each set. In this case, it is expected that there would also be a process plan for a higher level of control which would coordinate the execution of the several NC-programs. This model is not intended to describe current practice in industry, although the model is very similar to current practice. Rather, the model describes what it believed to be a good approach to machining, excluding feedback loops, which are desirable in the model, but have not been included on Figure 1. The OLPS system follows this model.

It is possible to build an automatic system that hides one or more of the functions shown in Figure 1 and does not allow intervention between stages. Indeed, the NIST Vertical Workstation has been demonstrated in the mode where a STEP file giving a boundary representation is a variable input and the finished part is the output [Kramer89]. Experience has shown, however, that the range of parts which can be produced this way is very much smaller than the range of parts that can be produced if the process is split up with the interface between subsystems consisting of data in a standard format. This permits subsystems to be developed or acquired independently. It also provides an entree for human intervention between stages.

In the near term, process planning and NC-programming should be done on systems designed to run both interactively and automatically. In the automatic mode, a system should handle what it can, but the system should revert to interactive mode smoothly if it runs into a situation it cannot handle. As the system becomes more capable in the automatic mode, more of the work can be done automatically.

2.2 Process Plans

Process planning may be defined as “a procedure for determining the operations or actions necessary to transform material from one state to another” [ANSI]. The core of a process plan is a set of procedures (which we will call “steps”), some or all of which must be carried out in order to achieve the objectives of the plan. The plan must also provide guidance on the alternatives for which steps must be carried out, and in what

order. In simple plans this information may be implicit by the understanding that all steps will be carried out in the order listed. In addition, a plan will probably include (i) administrative information, such as the name of the plan and the version number, (ii) a list of physical resources required, and (iii) a list of parameters or other data resources used in the plan.

At NIST, Dr. Steven Ray has devised a process planning language called ALPS (A Language for Process Specification) [Ray91]. An EXPRESS schema for ALPS was written by the author. ALPS was the language of the process plans used in the Vertical Workstation of the Automated Manufacturing Research Facility during an October 1990 Open House. The plans were prepared both as STEP physical files and as constructs in a relational database. ALPS is a domain-independent language and contains nothing specific to machining.

The IPO Manufacturing Technology Committee has also been developing a process planning language [Paul91], but it is not yet ready for use. This language is very similar to ALPS and is also domain-independent.

2.3 Machining Process Plan

A machining process plan is a plan for doing the machining required in a single fixturing of a part. It consists primarily of the specifications for a set of machining operations. The specification for a single cutting operation includes at least two types of data:

1. description of the operation - the type of operation, the specific type of tool to use, feed rate, spindle speed, horizontal stepover, vertical pass depth, and, possibly other information specific to the operation.
2. specification of the shape of the material to be removed - a pointer to a MRSEV.

In the initial implementation of OLPS, which took place before the ALPS language was developed, process plans were written in the LISP-readable format used in the “VWS2” software for the Vertical Workstation [Kramer87]. In that format, each machining step carried information specifying which other steps must be carried out first. In ALPS, this sort of information is carried partially in the individual machining steps, and partially in non-machining steps.

In order to write a process plan for machining, a library of machining operations which may be included in process plans is required. A library of 20 operations has been prepared in an EXPRESS schema [Kramer91b]. A subset of 8 of them has been implemented in OLPS. The library of operations has been plugged into the ALPS language by redefining one ALPS entity, “primitive”, to make it be the topmost entity in the hierarchy of entities in the library. The combined EXPRESS schema is used in OLPS as the basis for preparing machining process plans. An example is given in Appendix C.

2.4 NC Process Information Model

The IPO Manufacturing Technology Committee has been considering process planning and machining, as well as other issues, and is working on information models relevant to these topics.

In the document "NC Process Information Model" (NCPIM), Version 1.0, dated February 1, 1989, the IPO Manufacturing Technology Committee created an information model for NC processes. The model was *"limited to the NC processes which are generated for removing material from mechanical piece parts"* [Cain89, p. 6].

The NCPIM defines the term Material Removal Shape Element Volume, as the *"Specified unit of volume of material to be removed in a material removal plan. It may include the removal of material from several regions of the part-in-process which are not adjoining."* [Cain89, p. 58].

The NCPIM also defined "NC Removal Shape Element Volume" as *"A decomposition of a material removal shape element volume into NC removable shape element volumes. Each volume is entirely contained within its associated material removal shape element volume."* [Cain89, p. 59].

The NCPIM does not go into further detail about the two types of volumes. One example of the volumes is given: a Material Removal Shape Element Volume could be a series of holes to be made by a single drill [Cain89, p. 58]. It seems intended that an NC Removal Shape Element Volume must be all one piece, and not broken up into parts.

2.5 Form Features Information Model

The STEP Form Features Information Model (FFIM) [Dunn88, Dunn92] defines "form_feature" as its top-level entity. A form_feature may be a single feature, a pattern of copies of a single feature, or a group of (probably different) features. This paper is based on the 1988 version of the FFIM. Many of the names are changed in more recent versions.

The FFIM includes a large, comprehensive set of shapes defined by "feature_sweeps". There are three types of feature_sweeps: axisymmetric, along, and in_out. An axisymmetric_feature_sweep is basically a solid of rotation. The other two types are, roughly speaking, the solid you get if you sweep some type of planar profile along a path, keeping the plane normal to the path. The along_feature_sweep is meant to be along the surface of an existing solid, and is made with an open profile. The in_out_feature_sweep is intended to dig into or stick out of the surface of an existing solid. The feature sweeps come with a variety of methods of edge and end blending.

The feature_sweeps that describe closed volumes (axisymmetric_feature_sweep and in_out_feature_sweep) allow convenient description of elementary geometric shapes (block, wedge, sphere, cone, cylinder) and a very wide variety of other shapes.

The FFIM requires the user to know the relationship between the swept volume and the shape of the existing part in order to use feature_sweeps to make features. Each feature_sweep must be used to describe a depression, a protrusion, or a passage in order to be part of a feature.

The FFIM includes many other feature types in addition to feature_sweeps.

The FFIM was built with the intention that it be a resource model for use by applications in STEP. The MRSEV schema given in Appendix B uses the FFIM this way.

2.6 Tolerance Information Model

A Tolerance Information Model has also been formulated by STEP. Tolerance entities are used in conjunction with entities from other models to express tolerances.

2.7 MRSEV Technical Issues

MRSEV technical issues are listed but not discussed here. A discussion of these issues is the subject of a separate paper [Kramer92a]. The reader may find it helpful to read that paper in conjunction with reading this one. The resolution of each issue for the purposes of the library described in this paper is given after each issue.

1. How does a MRSEV relate to a machining operation? - Each metal cutting machining operation in a process plan will refer to a MRSEV. The volume described by a MRSEV should have no material in it when the machining operation is complete, and the operation should remove no material outside the volume.
2. How should patterns, groups and replications be handled? - See the description of the library.
3. How do MRSEVs relate to workpiece shape? - MRSEVs are defined independently of workpiece shape.
4. Should MRSEVs carry machining information?- No.
5. Should the MRSEVs be from a fixed library of canonical types or should they be defined as needed? - Use a fixed library.
6. Is air allowed in a MRSEV? - Yes.
7. How can MRSEVs be linked to design data? - This issue has not been addressed in building the library.
8. Should MRSEVs be defined as closed volumes or as open ones, or allow both? - The library uses closed volumes.
9. Should MRSEVs be defined entirely unambiguously, or is some ambiguity desirable?- In general, ambiguity is undesirable. The library leaves ambiguity only in the case of thread timing and, optionally, the end radius of edge_cuts.
10. What machining operations should the MRSEVs support? - The library supports 3-axis machining.

11. How can the definition of MRSEVs be made compatible with an approach to machining that insures that parts will be made within the tolerances specified in the design? - The library provides for tolerance information only in the case of MRSEVs which cannot generally be made nominally exactly (i.e., exact in principle).
12. How can MRSEVs be used for operations on surfaces? - The library does not provide MRSEVs tailored for surface operations.
13. What range of parts should be makable using MRSEVs? - The library is intended to be adequate to make parts which can be described in a STEP boundary representation in which the surfaces are elementary surfaces (plane, sphere, cone, cylinder, torus) and which can be machined on a 3-axis machining center using cutting tools whose spun volume (the swept volume of the tool when it is spinning but not feeding) has elementary surfaces where it contacts the workpiece. The library is not usable for dealing with sculptured surfaces.
14. How do MRSEVs relate to NC Removal Shape Element Volumes (NCRSEVs)? - The library provides for the relationship between MRSEVs and NCRSEVs suggested in the NC Process Information Model [Cain89 - page 59].
15. To what extent should specialized common shapes be included in a MRSEV library? - The library defined here provides relatively few MRSEVs of specialized types.
16. To what extent should MRSEVs carry information about their accessibility for machining? - The MRSEVs in the library carry no information about accessibility.

3 A Library of MRSEVs for 3-Axis Machining

This section defines a library of MRSEVs for use with process plans for a 3-axis machining center. This could be either a vertical-spindle machine or a horizontal-spindle machine. Both types typically perform the same operations.

3.1 Nature of MRSEVs in Library

At the elemental move level, the more capable 3-axis machining centers can drive the tip of a cutting tool in only three distinct trajectories: (i) a straight line between two points in 3-dimensions, (ii) an arc of a circle (or a complete circle) whose axis is parallel to one of the three principal axes of the machine, or (iii) an arc of a helix whose axis is parallel to one of the three principal axes of the machine. The library includes MRSEV types (holes, ramps, and grooves) for representing the swept volumes created when a cutter whose spinning shape is a cylinder with either a flat end or a rounded end is driven along any of those three trajectories - with the top of the swept volume cut off flat, and with the limitation that the axis of a circular or helical arc must be parallel to the axis of the cutter.

At a higher level, certain more complex shapes are commonly found on machined parts. These include complex grooves, pockets, steps, and exterior profiles. The library provides shapes for all of these. It is intended that the “pocket” MRSEVs defined here be used for producing not only what is usually called a pocket, but also for producing what are usually called steps and exterior profiles and for removing what are usually called slabs.

All the MRSEVs are to be construed as closed volumes. The tops of all MRSEVs are to be construed as being planar. Each type of MRSEV has a native coordinate system attached to it. When an instance of a MRSEV is built, the native coordinate system is located with respect to some parent coordinate system by a locator.

3.2 Description Methods

The library of MRSEVs is described here as a hierarchy of entity types. Each entity type has a number of attributes. The attributes have values which must be of some specific data type (single instances or sets of: a real number, a string, a choice of one of a set of tokens, or an entity type). Some attributes are optional. Describing an optional attribute as “omitted” in this section should be interpreted as meaning the same thing as having a value for the attribute which is a null value. An entity may have subtypes. The subtypes inherit all the attributes of the parent types and may have additional attributes of their own. Only entity types which do not have subtypes can be used as MRSEVs referenced by process plans. This description method is conceptually identical to a subset of EXPRESS, but we are not using EXPRESS syntax in this section, and any other language that includes the elements just described could be used.

Two types of figures are used in this section: (i) MRSEV hierarchies and (ii) descriptions of individual MRSEV types. In many cases, restrictions on the values of attributes are required. These restrictions do not appear in the hierarchies but are included in the individual descriptions.

Appendix B defines a subset of the MRSEV library in EXPRESS, using the STEP Geometry and Topology [Goult91], and Form Features [Dunn 88] models as resources. It should be feasible to use similar methods to define in EXPRESS the entire library given here.

3.3 Profiles

Many of the shapes in the library are defined with the aid of a “profile”. Profiles are used in the library both as two dimensional shapes which may be swept or rotated through space to make a solid, and as paths through space along which other profiles or cross sections can be swept.

The STEP Form Features Information Model (FFIM) and the Geometry Model both provide resources for defining profiles. This section assumes that some method using those resources can be implemented, but does not describe any specific implementation. An implementation using the FFIM is given in Appendix B.

3.3.1 General_Profile

A “general_profile” is defined to be an ordered collection of arcs of circles and straight line segments lying in a plane. The elements (arcs of circles and straight line segments) of a profile are numbered with consecutive integers, starting with 1. There must be at least one element. The elements are joined end-to-end to form a continuous curve. Adjacent straight line segments may not be colinear unless they go in opposite directions. Adjacent arcs of circles may not have the same center unless they go in opposite directions.

If there are three or more elements, or if there are two elements but only one pair of ends is connected, this establishes a positive direction for the profile - going from element one towards element 2. If there is only one element, or if there are two elements and both pairs of ends are connected, a positive direction is assigned some other way (such as by using the definition of the underlying curve of the first element). The “first point” of the profile is the point at the negative direction end of the first element, and the “last point” of the profile is the point at the positive direction end of the last element. These conventions lend themselves to straightforward implementations.

3.3.2 Other Types of Profile

Other types of profile are subtypes of general_profile. If the last point of a profile is in the same place as the first point, the profile is called a “closed_profile”. Otherwise, it is called an “open_profile”. If a closed_profile does not intersect itself, it is called a “nice_closed_profile”. If an open_profile does not intersect itself and has the following character, it is called a nice_open_profile (see Figure 17): Let a straight line L be drawn between the first point P and the last point Q of the profile. The profile does not intersect L except at P and Q. If a line M is drawn perpendicular to L at any point, then: (1) if M intersects L outside the segment between P and Q, it does not intersect the profile, and (2) if M intersects L inside the segment or at P or Q, it intersects the profile at exactly one point or it overlaps exactly one straight line segment that is part of the profile.

3.4 Hierarchy Figures and Conventions

Figure 2 shows the main hierarchy of MRSEVs in the library.

The layout of Figure 2 is intended to be easily understood. The following conventions are used:

1. Each entity type is defined in a white rectangular box.
2. The names of entity types are given in **boldface**.
3. The names of attributes of entities types are given in *italic*.
4. Select data types are given in helvetica.
5. The data type of each attribute is given after the attribute name. If the data type is not real, string, or select, it must refer to the name of an entity type defined elsewhere. Only one entity type referenced in this way is defined on Figure 2; that is mrsev_volume, which is the type of one of the elementary_volumes of a MRSEV. That connection is shown near the top of the figure by a heavy broken line.
6. Parent entity types are located above their subtypes and are connected to their subtypes by solid lines, which may branch horizontally. A complete set of all attributes for each instantiable (leaf node) entity type in the library can be extracted from Figure 2 by tracing these lines upwards through all the higher ancestors of the entity type.

Figures 3 and 4 use the same graphic conventions. Figure 3 shows the hierarchy of islands which may occur in pockets. Figure 4 shows the hierarchy of copy methods which may be used with MRSEVs.

In the remainder of this paper, the specific entity (mrsev) defined in the hierarchy will be spelled using lower case letters, while the generic MRSEV will continue to be spelled in capitals.

3.5 The Main Hierarchy

As shown in Figure 2, the topmost entity of the main hierarchy is mrsev. A mrsev has three attributes: (i) *elementary_volumes*, the value of which is a set of one or more mrsev_volume, (ii) *copy_maker*, the value of which is an optional mrsev_copy_method, and (iii) *mrsev_name*, the value of which is a string. Within a physical file defining a set of mrsevs, no two mrsevs should have the same *mrsev_name*.

Mrsev_copy_methods are discussed in subsection 3.7.

The rest of Figure 2 shows the subtypes of mrsev_volume. The two main subtypes are primary_mrsev_volume (which has a location of its own) and secondary_mrsev_volume (which has location only with respect to a primary_mrsev_volume). The only subtype of secondary_mrsev_volume in the library is thread, which is a right-handed thread on the interior of a hole (for holding a bolt).

Every primary_mrsev_volume is located in the overall coordinate system for mrsevs (mentioned earlier) by the *location* attribute, the value of which must be an "axis_placement". The axis_placement defined in the STEP geometry schema [Goult91] could be used here (and is used in the schema of Appendix B). We do not define axis_placement any further in this paper.

The subtypes of primary_mrsev_volume and their approximate shapes are as follows. In most cases there are further restrictions on the shapes (which are discussed later) so that the shapes can be produced by machining. Each shape is bounded on one side by a plane which forms the upper surface of the mrsev_volume and lies perpendicular to the axis of the cutting tool when the shape is in position to be machined. Details of the location of the upper bounding plane are not given in the descriptions which follow, but are given later.

1. groove - the shape resulting from sweeping a cross section (which may be a parametrically described standard shape or a nice_closed_profile) along a general_profile path while keeping the cross section perpendicular to the path. Typically, the cross section will be the silhouette of a cutter and the path will lie on a flat surface of a workpiece, so that removing this shape from a workpiece results in what is commonly called a groove.
2. ramp - the shape resulting from sweeping a cutting tool either (i) in a straight line which does not lie in the plane perpendicular to the axis of the tool or (ii) in an arc of a helix whose axis is parallel to the axis of the tool.
3. linear_sweep - the shape resulting from sweeping a nice_closed_profile along a straight line perpendicular to the plane of the profile. Typically, the line will be perpendicular to a flat surface of a workpiece. The two subtypes of linear_sweep are holes and pockets. Islands of material may be left in some types of pockets. Islands are discussed in section 3.6.
4. rotation_pocket - the shape resulting from swinging a nice_open_profile all the way around an axis that passes through the ends of the profile. For a horizontal_rotation_pocket, part of this shape is cut off with a plane that is parallel to the axis.
5. edge_cut - the shape resulting from sweeping a right triangle along a general_profile path while keeping the triangle normal to the path. The hypotenuse of the triangle may be replaced by an arc of a circle. The use of a mrsev of this type is in flattening or rounding an edge of a workpiece.

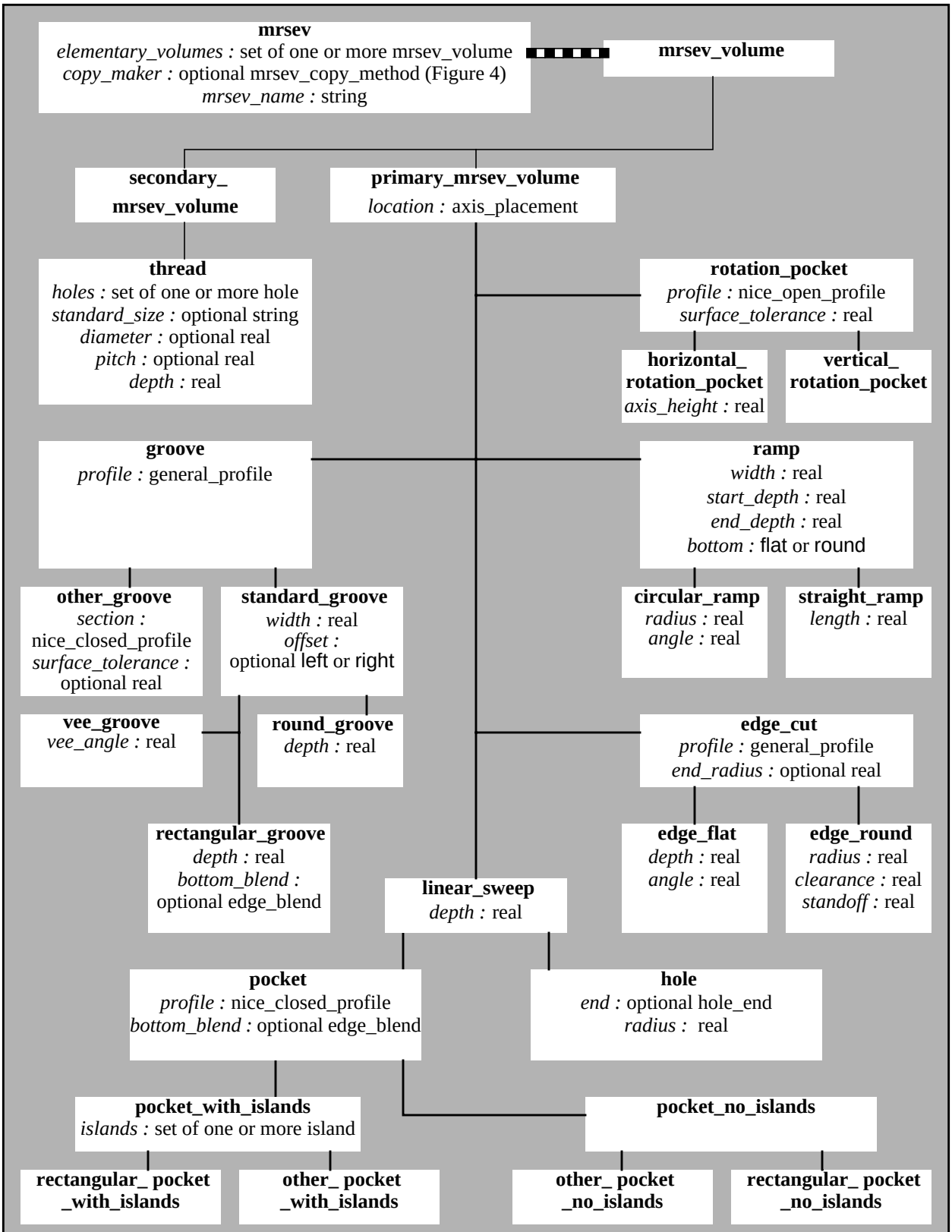


Figure 2. Main Hierarchy of MRSEVs in Library

3.6 Islands

An island is a closed volume which is to be boolean subtracted from a pocket mrsev. Only the (flat-bottomed) pocket_with_islands subtype of pocket may contain an island. Because an island is boolean subtracted from its parent pocket, if an island extends through the top surface of the pocket, that portion of it which is outside the pocket has no effect; a tall island does not add material to a workpiece.

As shown in Figure 3, three subtypes of island are defined. Each may best be thought of as the opposite of one of the mrsev types. The island is what you would get if you filled its opposite with material (ignoring edge conditions), as if casting in a mold, and then inverted the mold, extracting the cast shape.

1. mesa_island - the opposite of a pocket without islands. A mesa_island looks like a mesa (see Figure 7).
2. anti_groove_island - the opposite of a groove (see Figure 19).
3. rotation_island - the opposite of a rotation_pocket (see Figures 20 and 21).

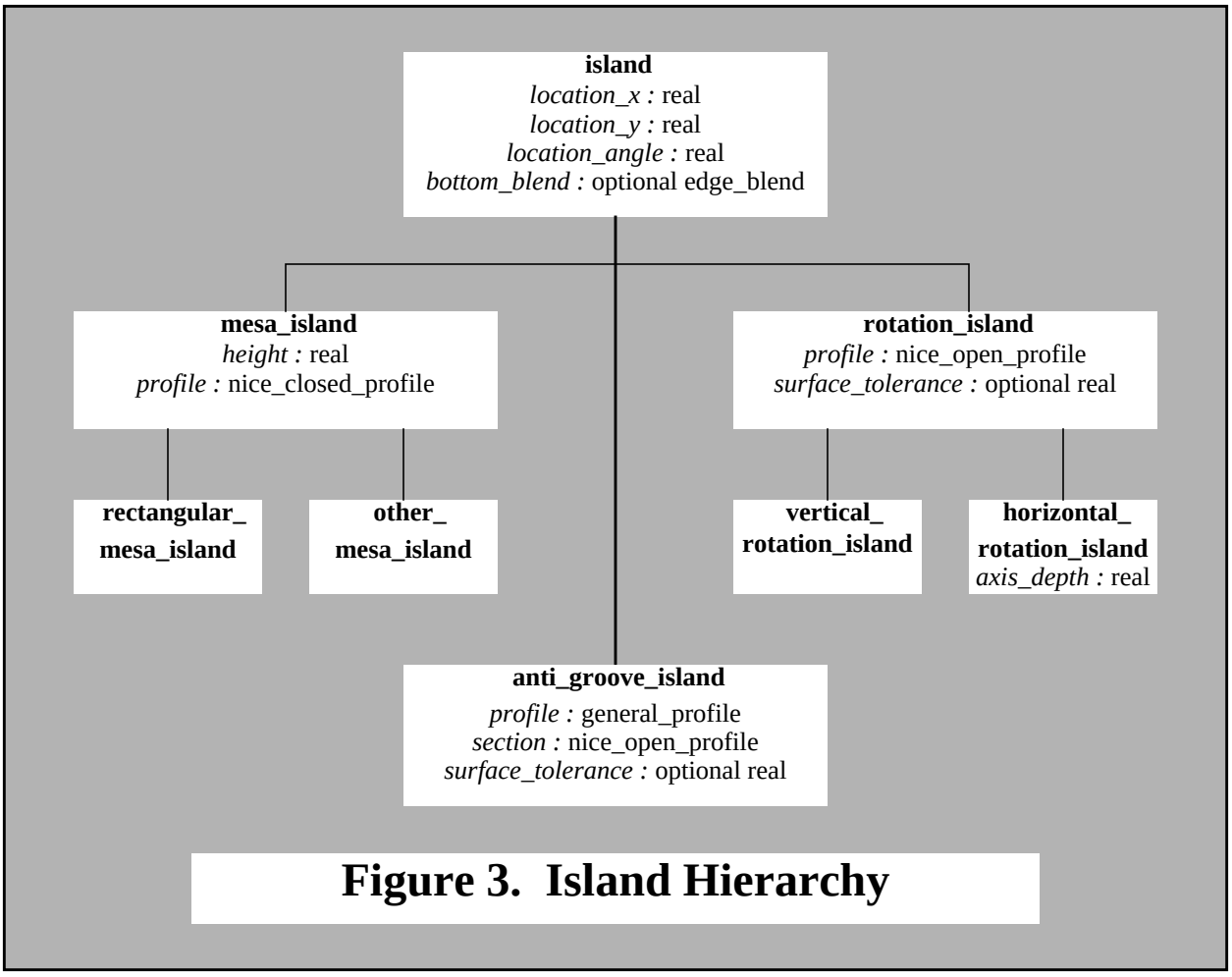


Figure 3. Island Hierarchy

3.7 Groups and Copy Methods

Groups are provided in the MRSEV library by having the value of the *elementary_ volumes* attribute of a mrsev be a set of mrsev_volume's rather than a single mrsev_volume. This is shown in the top block of Figure 2. Of course the set may (and typically will) have only one element. The mrsev_volume's in a set do not need to be of the same type. For example, pockets and grooves might both be included. Because groups are provided this way, no entity named "group" is required and none is defined.

Copy methods include patterns and replications. Figure 4 shows the hierarchy of copy methods. Figure 5 shows illustrations of groups, patterns and replications.

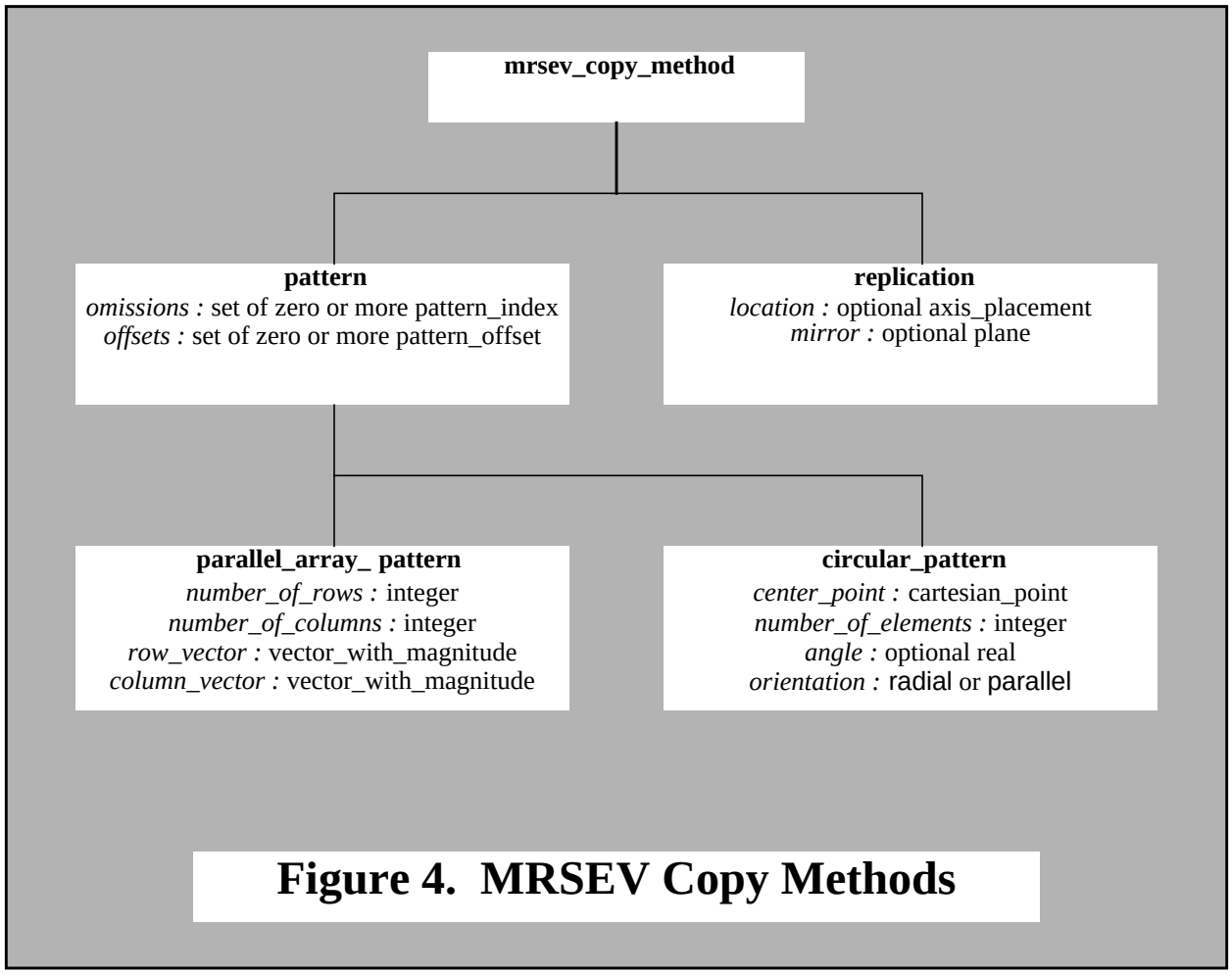


Figure 4. MRSEV Copy Methods

3.7.1 Patterns

A pattern is a rule for making additional copies of some base shape or shapes. All patterns in this library are two-dimensional and are to be interpreted as applying in the xy-plane of the MRSEV file. The pattern does not include any base shape. The native coordinate system of the pattern is the coordinate system of the MRSEV file. The parallel_equal_spacing_array_pattern and implicit_circular_form_feature_pattern

described in the Form Features Information Model (FFIM) have been adapted for use in this MRSEV library, using the shorter names “parallel_array_pattern” and “circular_pattern”.

These patterns include options for omissions and offsets of specified elements of a pattern. An omission from a pattern is represented by giving the index (for a circular_pattern) or two indices (for a parallel_array_pattern) of the element to be omitted. The number of elements in a circular_pattern is the number before any omissions are made. A pattern_offset is represented by the identification of the element to be offset and a vector giving the direction and magnitude of the offset, which is to be applied after the element is located by the pattern rules. Omissions and offsets are not discussed further in this paper.

The first location specified by a pattern in a mrsev is the location of the first mrsev_volume in the *elementary_volumes* of the mrsev. The attributes of the pattern tell how to identify additional locations relative to this starting point. Another copy of the first mrsev_volume in the *elementary_volumes* of the mrsev is to be made at each such location. If there is more than one mrsev_volume in the *elementary_volumes* of the mrsev, then, at each location, copies of the other mrsev_volumes in the set will be made in the position relative to the copy of the first mrsev_volume as determined by the definitions of the mrsev_volumes. In this way it is possible to make patterns of groups of mrsevs. We have not provided for recursive patterns. In other words, we cannot make patterns of patterns. It may be desirable to add this capability.

Note that since a pattern does not include a base shape, the same pattern can be used in several mrsevs. If a 2 by 3 array of holes were to be center_drilled, drilled, and counterbored, for example, all 18 hunks of material to be removed could be specified by defining three mrsevs, each of which used the same pattern but a different base shape.

The attributes of the two patterns should be interpreted as implied by the names of the attributes. A few additional explanations are required.

For a parallel_array_pattern:

1. The *row_vector* gives the direction and distance between adjacent elements of the same row
2. The *column_vector* gives the direction and distance between adjacent elements of the same column.
3. The orientation of the copies is the same as the orientation of the original mrsev_volume.

For a circular_pattern:

1. The axis of the circle is parallel to the z-axis of the location of the first mrsev_volume in the *elementary_volumes* and inherits the positive sense of that z-axis.

2. Copies will be equally spaced. In other words each angle between a line from the center of the circle to the z-axis of copy n and a line from the center of the circle to the z-axis of copy n+1 will be equal to every other such angle.
3. If the *angle* attribute is omitted, the copies will be arranged equally spaced in a full circle.
4. If the *angle* attribute is used, it will be given in degrees, it will be less than 360, greater than -360 and not zero. It represents the angle swept by a line from the center of the circle as it rotates from pointing to the first location to pointing to the last. The positive direction of rotation is counterclockwise as viewed from the positive end of the axis of the circle.
5. The *number_of_elements* is the total number, including the original being copied.
6. The *orientation* attribute indicates whether the x-axes of the copies should be parallel to each other (in which case the value of *orientation* should be "parallel"), or whether the x-axis should rotate so that a constant angle is maintained between the x-axis of each copy and a radius of the circle (in which case the value of *orientation* should be "radial"). Figure 5 shows the radial case.

3.7.2 Replication

A replication is a reproduction of a set of mrsev_volumes in another location. This may be specified by either an axis_placement, which should be interpreted as a replacement for the axis placement specified in the first mrsev_volume in the set, or by a mirror plane, or both.

If a *location* is specified and there is more than one mrsev_volume in the set, then a copy of each other mrsev_volume in the set will be made in the position relative to the copy of the first mrsev_volume as determined by the definitions of the mrsev_volumes.

Mirroring through a plane will be done in the conventional manner of mapping each point P of the mrsev_volume to a new point P' by drawing a line from P perpendicular to the plane and extending the line an equal distance on the other side of the plane. The end of the line is P'.

Mirroring thread mrsevs, however, will be understood not to reverse the sense of the thread; all threads are right-handed. Mirroring maps each mrsev into a mrsev of the same type, as a consequence of how this library is constructed.

If a replication includes both a *location* and a *mirror*, the *location* should be applied before the *mirror*.

3.8 Details of Mrsev_Volumes

This section presents details of mrsev_volumes which can be instantiated. A figure which includes a list of the attributes required is given in most cases. The figures generally show the top and front orthographic views of an example of a mrsev_volume of the given type. The native coordinate system is indicated on each figure using heavy black arrows labelled “x-axis”, “y-axis”, and “z-axis”.

The figures in this section show mrsev_volumes as though they were voids in a block of material, with the top of the mrsev_volume at the top of the block. It should be borne in mind, however, that mrsev_volumes are defined as closed volumes to be boolean subtracted, so the appearance of the remaining material would differ if the mrsev_volume were placed differently.

3.8.1 Linear_Sweep Pockets

Only two (rectangular_pocket_no_islands and other_pocket_with_islands) of the four types of pockets which are subtypes of linear_sweep are discussed here. The other two (rectangular_pocket_with_islands and other_pocket_no_islands) are exactly as would be expected.

For all linear_sweep pockets:

1. The origin of the native coordinate system of the pocket lies on the plane of the bottom of the pocket.
2. The z-axis of the native coordinate system of the pocket is perpendicular to the plane of the bottom of the pocket.
3. The *depth* of the pocket (a positive real number) is measured parallel to the z-axis.
4. If the bottom of a pocket is blended with the walls of the pocket (implying the bottom must be blended with the walls of any islands, as well), the blends must have room to reach all the way to the bottom of the pocket.

3.8.1.1 Rectangular_Pocket_No_Islands

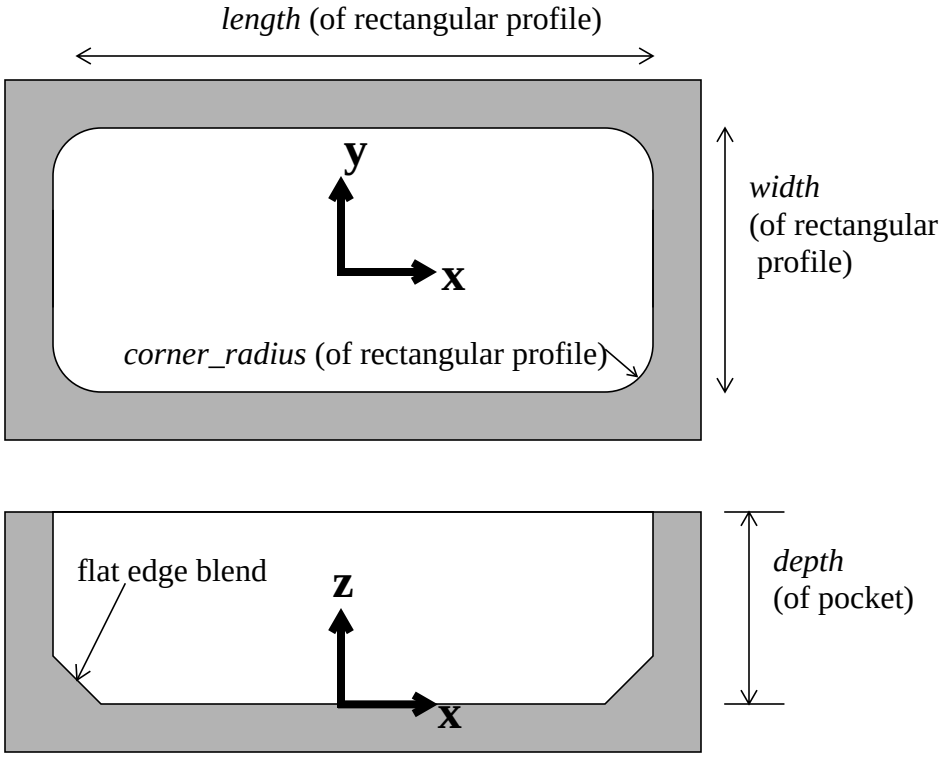
An example of a rectangular_pocket_no_islands is shown in Figure 6.

The *profile* of a rectangular pocket will be a rectangle with corners which may be rounded. The rectangle is oriented with its sides (before rounding) parallel to the x- and y-axes of the native coordinate system. The data to describe the profile will include *length* (a positive real number) for the length of the side parallel to the x-axis, *width* (a positive real number) for the length of the side parallel to the y-axis, and *corner_radius* (a non-negative real number) for the radius of corner rounding. If *corner_radius* is zero, the corners are not rounded. *Length* must be greater than or equal to *width*.

Corner_radius may not be greater than half of *width*, but it may equal that number. This means that two or all four sides of the rectangle may be rounded away entirely, yielding a “racetrack” shape in the first case, and a circle in the second.

The local coordinate system of the pocket will be at the center of the bottom of the pocket.

This example is shown with a flat edge blend between the walls of the pocket and the bottom. Alternatively, there could be a round blend (see Figure 7) or no blend. The lines where the blend meets the bottom of the pocket are not included in the top view of this figure.



attribute of pocket	**type**
location	axis_placement
depth	positive real number
profile	rectangular_profile
bottom_blend	edge_blend

attribute of rectangular_profile	**type**
length	positive real number
width	positive real number <= length
corner_radius	non-negative real number

Figure 6. Rectangular_Pocket_No_Islands

3.8.1.2 Other_Pocket_With_Islands

An example of an other_pocket_with_islands is shown in Figure 7.

The profile of an other_pocket_with_islands (which must be a nice_closed_profile) has a two-dimensional native coordinate system of its own. The x- and y-axes of the native coordinate system of the other_pocket_with_islands are inherited from the profile, and, thus, may be either inside or outside the pocket. The z-axis of the native coordinate system is generated from the x- and y-axes of the pocket to form a right-handed system. The origin is placed at the bottom of the pocket.

3.8.2 Hole

Figure 8 shows five examples of holes.

A hole is a right circular cylinder with an optional end condition at the bottom. The native coordinate system of a hole has its origin at the center of the bottom, with the z-axis on the axis of the cylinder, pointing toward the top.

End conditions on holes include conical end, round end, flat blend, and round blend. The end condition of a hole may use up the entire length of the cylinder, but may not require more than that. Thus, a hole may be purely hemispherical or conical. If the end condition is a blend, it may not blend away the entire bottom of the hole.

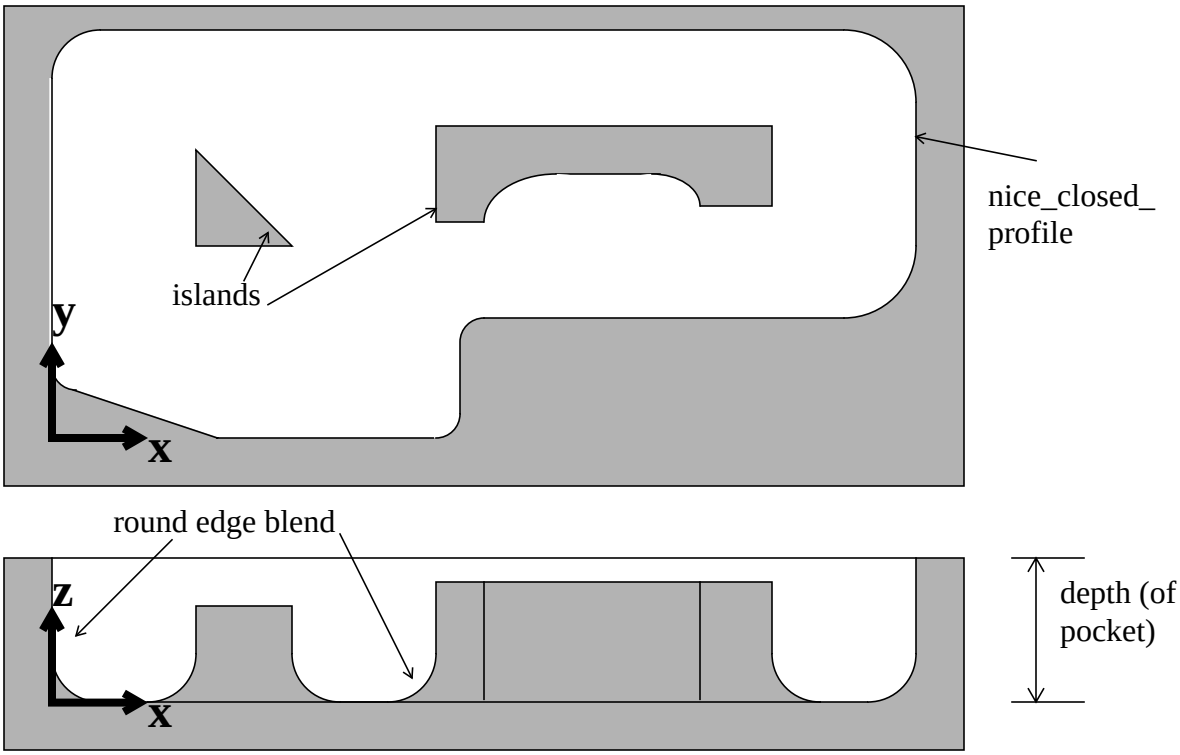
The *depth* of the hole is the length of the cylinder.

Holes are intended to be used for center_drilling, twist_drilling, counterboring, boring, countersinking, and any other operation involving an essentially cylindrical or conical MRSEV.

Earlier versions of this library have included a tip condition which may be applied to an end condition. For example, a hole could have a conical end, and then the tip of the cone could be rounded. Earlier versions also included allowing a blend between the walls of the cylinder and the end, as well. For simplicity, no tip conditions are included in this version, and blends are included as a type of end condition, not as something that may be applied in addition to an end condition. It may be desirable to add these later to allow the expression of a wider variety of shapes.

The library does not include stepped holes, and never has included them. A stepped hole is a hole with an enlargement of the radius of the cylinder at the top, or with several successive enlargements, possibly including blends between enlargements. It may be desirable to add stepped holes to this library, since cutting tools to make them in one operation are in common use.

This example is shown with a round edge blend between the walls of the pocket and the bottom. Alternatively, there could be a flat blend (see Figure 6) or no blend. The lines where the blend meets the bottom of the pocket are not included in the top view of this figure. The islands here are both mesa_islands.

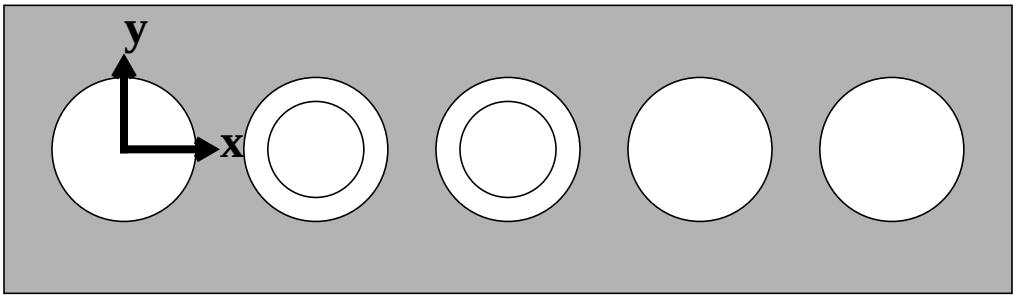


attribute of pocket	**type**
location	axis_placement
depth	positive real number
profile	nice_closed_profile
bottom_blend	edge_blend
*island*s	set of one or more island

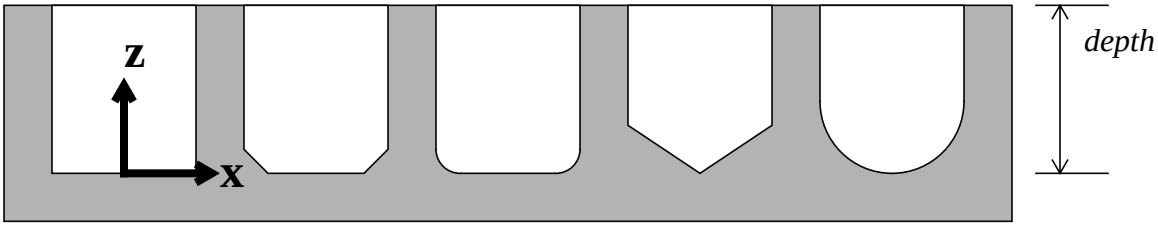
attribute of mesa_island	**type**
location_x	real number
location_y	real number
location_angle	real number
bottom_blend	optional edge_blend
height	positive real number
profile	nice_closed_profile

Figure 7. Other_Pocket_With_Islands

This figure shows five holes. A coordinate system is located in the one on the left. The co-ordinate systems of the others would be similarly located. The hole on the left has no end condition. The other four holes have end conditions as labelled.



flat blend round blend conical end round end



attribute of hole	**type**
location	axis_placement
depth	positive real number
end	hole_end
radius	positive real number

Figure 8. Holes

3.8.3 Grooves

As stated earlier, a groove is the shape resulting from sweeping a cross section (which may be a parametrically defined standard shape or a nice_closed_profile) along a general_profile path. For any of the three standard_grooves, the cross section will be the silhouette of a cutter and the path will lie on a flat surface of a workpiece, so that removing this shape from a workpiece results in what is commonly called a groove.

3.8.3.1 Standard_Groove Cross Sections

The three standard_grooves are rectangular_groove (made by a flat-ended end-mill which might have a radius or flat between the end and side), round_groove (made by a ball-nosed end-mill), and vee_groove (made by an engraver or other conical tipped cutter). The cross sections of the standard grooves are shown in Figure 9. For the standard grooves, instead of using a nice_closed_profile to describe the cross section of the groove, two or three attributes of the groove itself describe the cross section. This could be done equally well (but with more data required) by creating subtypes of nice_closed_profile for use as cross sections of grooves.

If a rectangular groove has a bottom_blend, the blend may not require more than the depth of the rectangle and may not use up the entire bottom of the groove.

The bottom of a round_groove is a circular arc. The diameter of the arc is not given as an attribute since it may be calculated from the width and the depth. If the depth is greater than or equal to half the width, the diameter equals the width. If the depth is less than half the width, the diameter equals ($d + (w^2/4d)$) where w is width and d is depth.

The cross section for a groove has a native 2-dimensional coordinate system. In the case of standard_grooves, the origin of this coordinate system is at the center of the bottom of the cross section, and the x-axis is parallel to the width of the cross section. For relating this 2-dimensional coordinate system to the groove, a z-axis is added, making a right-handed coordinate system.

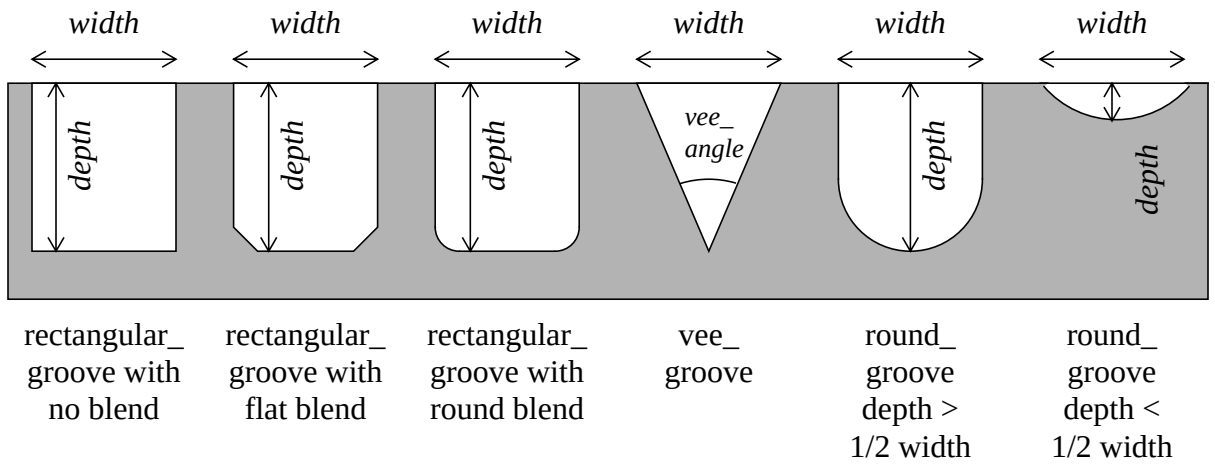


Figure 9. Standard Groove Cross Sections

3.8.3.2 Profile

The *profile* of a groove (the path along which the cross section travels) is described in terms of its own two-dimensional coordinate system. The coordinate system of the groove is located with respect to the groove by adding a z-axis to the coordinate system of the *profile*, making a right-handed coordinate system. The *profile* does not have to be closed, and it may intersect itself.

We have not put any standard shapes (such as circles or rectangles) for groove *profiles* in the library. The standard_grooves defined here have standard cross sections, not standard profiles. It may prove desirable to add standard profiles.

3.8.3.3 Forming the Groove

In order to form a groove, the origin of the coordinate system for the cross section is dragged along the *profile* from the first point to the last, sweeping the cross section through space. The swept volume is the groove. During the dragging, the y-axis of the cross section is oriented parallel to the z-axis of the coordinate system of the groove and the z-axis of the cross section is kept pointing in the direction of travel. That direction is defined by the positive direction of the *profile* and the tangent to the *profile* at the point where the cross section is attached to the *profile*. If there are any sharp corners in the *profile*, so that the tangent is not defined at the corner point, the attachment point will stop moving for a moment while the cross section rotates through the angle less than 180 degrees necessary to reorient the z-axis of the cross section from being parallel to the semi-tangent at the end of one element of the *profile* to being parallel to the semi-tangent at the beginning of the next element.

If the *profile* is open, the additional volumes swept in rotating the cross section through 180 degrees about its y-axis at the ends of the *profile* are added to the groove.

For a standard_groove, the swept volume just described corresponds to the shape of the cut a machining center will make by driving the center point of the tip of a cutting tool around a *profile*.

3.8.3.4 Standard_Groove Offsets

If the groove is a standard groove with an *offset*, a new profile is generated and used as just described to form the groove.

If *offset* is “left”, the new profile is the path that would be taken by a point on the far plus-x side of the cross section if the groove were made without an *offset*. If the original *profile* has (i) sharp left turns or (ii) arcs which bend to the left and have a radius of less than half the width of the cross section, this path becomes complex. In order to avoid the complexities, we will not allow the use of “left” *offset* in this situation.

Similarly, if *offset* is “right”, the new profile is the path that would be taken by a point on the far minus-x side of the cross section if the groove were made without an *offset*. Also similarly, we will not allow the use of “right” *offset* if the original *profile* has (i) sharp right turns or (ii) arcs which bend to the right and have a radius of less than half the width of the cross section.

3.8.3.5 Other_Grooves

For an other_groove we make a similar restriction (for the same reason) as follows: (i) the *profile* may not have any sharp corners, (ii) arcs which bend to the left may not have a radius of less than the maximum value of x reached on the cross section, and (iii) arcs which bend to the right may not have a radius of less than the absolute value of the minimum value of x reached on the cross section.

In addition, if an other_groove is open, the cross section must be symmetric about the y-axis. This is so that the shape at the ends of the groove will be well-defined.

The cross section of an other_groove is, of course, given by the nice_closed_profile that is the value of the *section* attribute of the groove. We will require that the minimum value of y reached on the cross section be zero. In most cases, the upper boundary of the cross section will be a line parallel to the x-axis of the cross section.

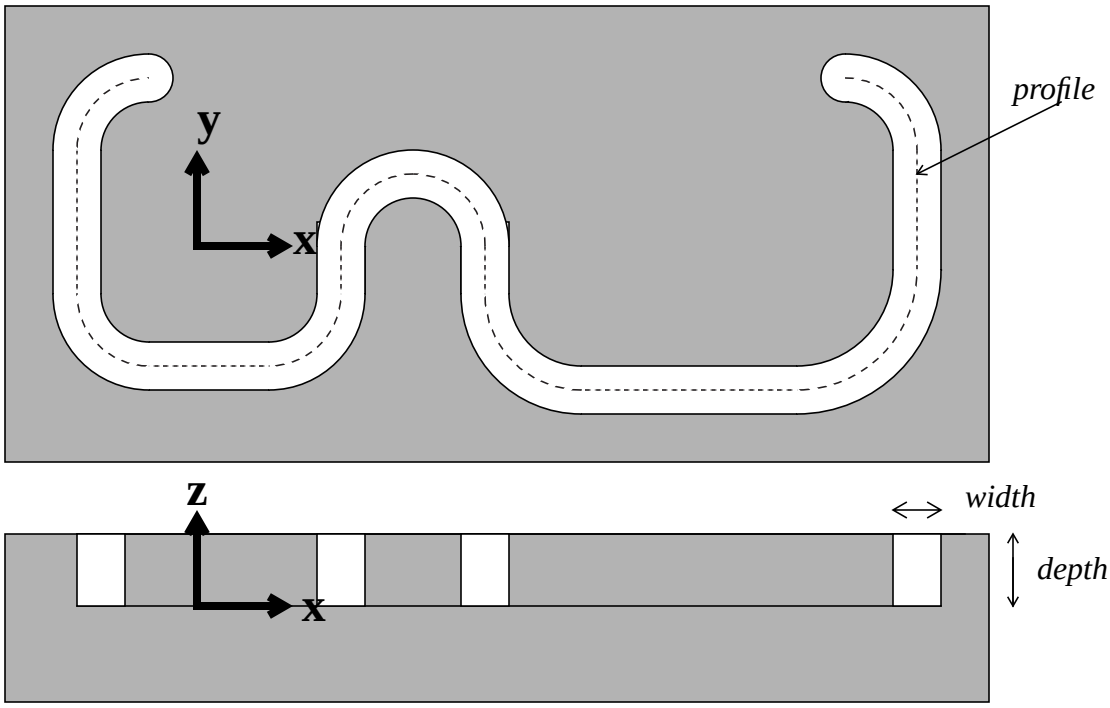
Since it may not be possible to machine an other_groove nominally exactly, this type of groove has an optional *surface_tolerance* attribute, which indicates the maximum distance allowed between the nominal surface of the groove as defined and the nominal surface of the groove as machined.

3.8.3.6 Groove Examples

Figure 10 shows an example of a rectangular groove. It uses an open *profile*, which happens not to intersect itself. The *offset* and *bottom_blend* attributes are omitted for this particular groove.

Figure 11 shows an example of an other_groove. It uses a closed *profile*, which also does not intersect itself.

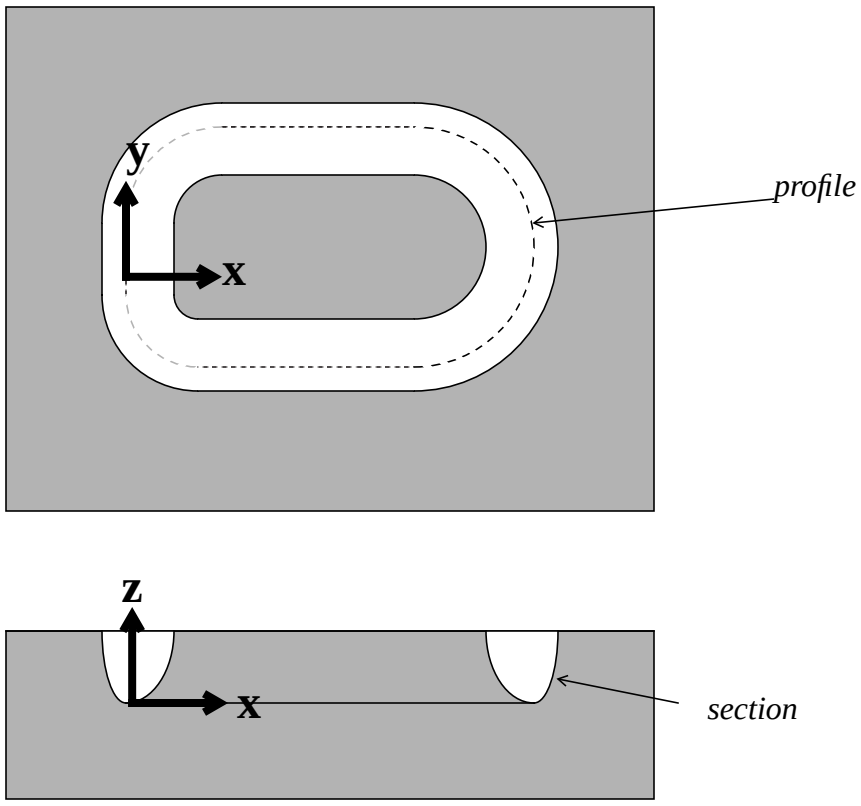
This figure shows a rectangular_groove. The *profile* of the groove is shown with a dotted line on the top view of the groove. The *profile* is open. The rectangle is not blended at the bottom, and the rectangle is not offset with respect to the *profile*.



attribute of rectangular_groove	**type**
location	axis_placement
profile	general_profile
width	positive real number
offset	optional left or right
depth	positive real number
bottom_blend	optional edge_blend

Figure 10. Rectangular_Groove

This figure shows an other_groove. The *profile* of the groove is shown with a dotted line on the top view of the groove. The *profile* is closed. The cross section of the groove is shown on the front view. This particular groove could not be machined nominally exactly.



attribute of other_groove	**type**
location	axis_placement
profile	general_profile
section	nice_closed_profile
surface_tolerance	optional positive real number

Figure 11. Other_Groove

3.8.4 Thread

An example of a thread is shown in Figure 12.

The thread mrsev_volume is for use with a tapping operation that will put a thread on the inside of an existing hole. A thread mrsev_volume is the volume that would be taken up by the portion of a bolt that would screw into the hole to the end of the thread.

The thread is the only subtype of secondary_mrsev_volume in the library. It is a secondary volume because it does not have a location of its own. The location of a thread is determined by the location of its parent hole. The thread does have a depth of its own, however. The thread starts at the top of the hole (a distance from the origin of the hole along the z-axis of the hole equal to the depth of the hole) and extends in the minus-z direction a distance equal to the *depth* of the thread.

A thread is not described by adding optional attributes for a thread to the definition of a hole because at least two machining operations are generally required to make a threaded hole: a hole-making operation such as twist_drill and a thread-making operation, such as tapping. An appropriate mrsev is needed for each operation.

Actually, a thread may be many different volumes, because a thread mrsev_volume applies to a set of holes. Thread is defined this way to keep the amount of data needed for threading holes in a workpiece to a minimum.

The *depth* of a thread must be less than the *depth* of its parent hole, less the vertical length of any hole end condition the hole may have.

The *diameter* of a thread (if given) must be greater than twice the *radius* of the parent hole.

To describe the thread fully, either the *standard_size* string must be given or the *diameter* and *pitch* must be given, but not both. All three are optional in the definition.

A mrsev which has a thread in its *elementary_volumes* may have a *copy_maker* only if holes have been defined at the locations specified by the *copy_maker*. In most cases, the same *copy_maker* would be applied to the parent hole of the thread and to the thread. It is possible to thread only some of the holes in a pattern of holes by having the *copy_maker* for the thread be a duplicate of the pattern for the holes, except that pattern omissions are specified where holes are not to be threaded.

3.8.5 Edge_Cut

An edge_cut is the shape resulting from sweeping a right triangle along a general_profile path while keeping the triangle perpendicular to the path. One leg of the triangle is kept parallel to the plane of the profile, and the other leg is kept perpendicular to the plane. The hypotenuse of the triangle may be replaced by an arc of a circle which curves toward the right angle.

There are two subtypes of edge_cut, “edge_flat”, in which the hypotenuse of the triangle is kept as a straight line, and “edge_round”, in which the hypotenuse is replaced by an arc of a circle.

An edge_cut is used for flattening or rounding an edge of a workpiece. Typically this is done with a rounder or chamfer tool in a rounding or chamfering operation. The edge_cut could also be used with a countersinking operation, but using a hole with a conical end is preferable.

Edge-cuts and edge-blends are both treated as blends in most solid modelers, but they are quite different in machining, where the names “chamfer” and “fillet” are more commonly used. An edge_blend leaves extra material, while an edge_cut indicates material to be removed. An edge_blend is often made as its parent MRSEV is being made, so edge_blends are represented as attributes of the parent feature. Usually, machining the parent MRSEV will leave an edge to which an edge_cut is applied later in a separate operation. Edge_cuts can also be applied to outer edges of a workpiece, where there may be no parent MRSEV.

If the entire edge of a pocket is to have an edge_cut applied to it, the same profile used to define the pocket can be used to define the edge_cut, so that little extra data is required to define the edge_cut, once the pocket is defined.

The swept volume of an edge_cut and the method of describing the sweeping process are the same for an edge_cut as for a groove, except at the end of an edge_cut using an open profile, which is discussed below. The native coordinate system of the triangle has its origin at the corner of the triangle which has the right angle. One side of the triangle lies on the minus-x axis and the other side lies on the minus-y axis. The coordinate system of the edge_cut itself is the coordinate system of its profile with a z-axis added.

The prohibitions concerning right and left turns which apply to a groove will not apply to an edge_cut. This is because during the machining of a groove, the cutter is usually surrounded by material, but during machining an edge_cut, the cutter is usually mostly surrounded by air and can machine the edge_cut without gouging the part. Also, the volume swept by the cutter in making a groove is expected to be about the same as the volume of the groove, whereas the edge_cut volume is expected to be significantly less than half the swept volume of the cutter that makes it.

If the profile of an edge_cut is open, there will be an addition to the swept volume of the triangle which differs from the additional volumes at the end of an open groove. The additional volume at the end of the edge cut will always be a piece of a cone for an edge_flat and a piece of cone minus a piece of torus for an edge_round, but the exact

shape will depend upon the cutting tool being used and how it is used. Thus, an optional attribute called “end_radius” is provided so the user may define the end shape by providing a value or leave the end shape ambiguous by not using the attribute.

Figure 13 shows how the shape of the end of an open edge_flat can vary according to how the tool to make it is used.

This figure shows three standard orthographic views of a block which is having an edge_flat made on two sides by two identically shaped chamfer tools. The surface that has been cut by the tools is shown on the top view with the lightest shading. It is easy to see that the right ends of the two surfaces (indicated by arrows) are different. The reason for this is that the two tools are being used differently. The crosshatched tool has its tip farther away from the side of the block but closer to the bottom of the block than the shaded tool. Thus, the end_ radius made by the shaded tool (shown with a heavy black line) is smaller.

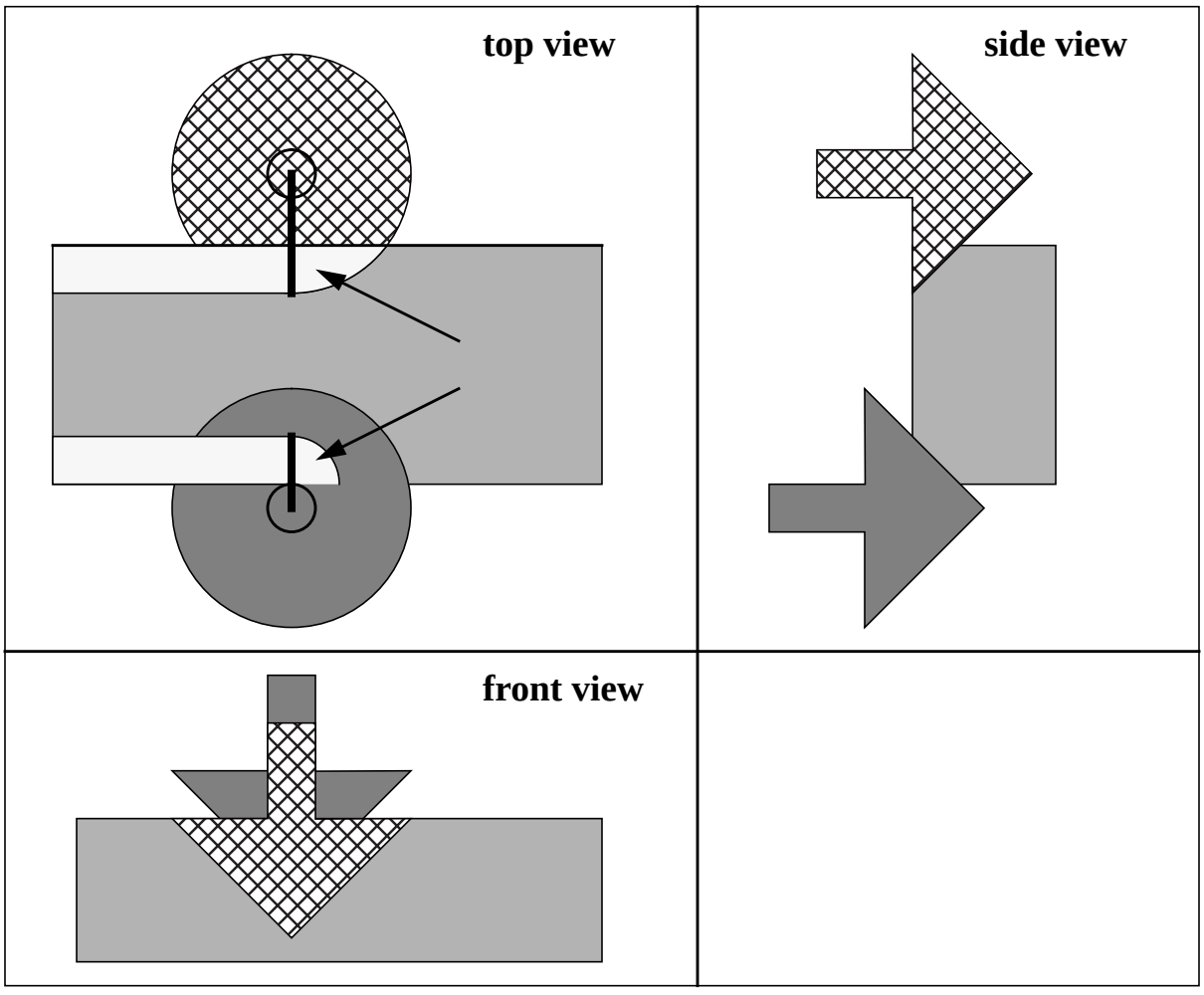


Figure 13. Ends of Edge_Flat Differ

Figure 14 shows a partial edge_flat on the outer edge of a block-shaped workpiece. The *depth* attribute is the length of the side of the triangle that is parallel to the y-axis of the cross section. The *angle* attribute is the angle between that side and the hypotenuse of the triangle.

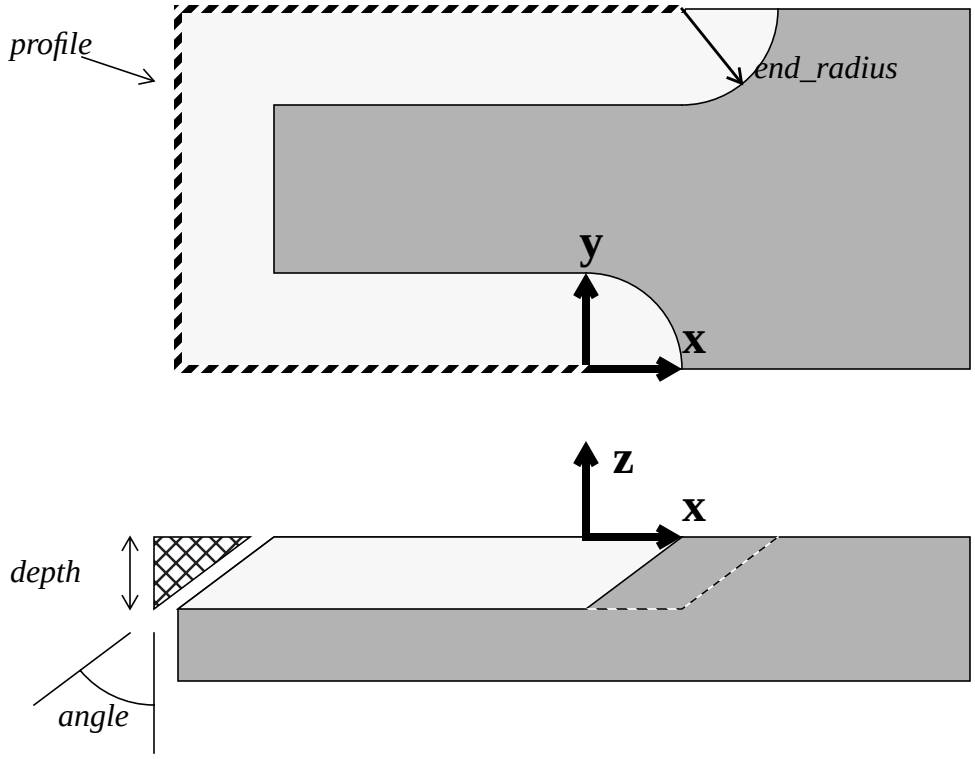
Figure 15 shows an edge_round on the inside of a pocket.

The attributes *clearance* and *standoff* are (i) the maximum value of y on the circle whose arc has replaced the hypotenuse of the triangle (in the native coordinate system of the triangle), and (ii) the maximum value of x on the circle. The value of both attributes is zero if a smooth rounding of a horizontal plane with a vertical wall is being cut. The value of both of those attributes must be a non-negative real number less than the *radius*. Figure 16 shows an enlarged front view of a typical rounding tool making an edge_round on an edge formed between a horizontal surface and a vertical one. As may be seen from the figure, *standoff* indicates how far the tool stands off from the vertical surface, and *clearance* indicates the clearance of the outer edge of the tool above the horizontal surface.

In the case of an edge_flat, the exact added volume is a slice of a cone taken perpendicular to the axis of the cone. The *angle* of the edge_flat is half of the tip angle of the cone, the *end_radius* of the edge_flat is the radius of the larger circular surface of the slice, and the *depth* of the edge_flat is the thickness of the slice. The location of the slice may be deduced from Figure 13 and will not be described in detail here.

In the case of an edge_round, the exact added volume is the same slice, with a torus subtracted from it. The torus is not described exactly here, but can be. The minor radius of the torus is the *radius* of the edge_round. The center of the torus is at the point marked A on Figure 16, and its axis would lie on the axis of the tool in that figure.

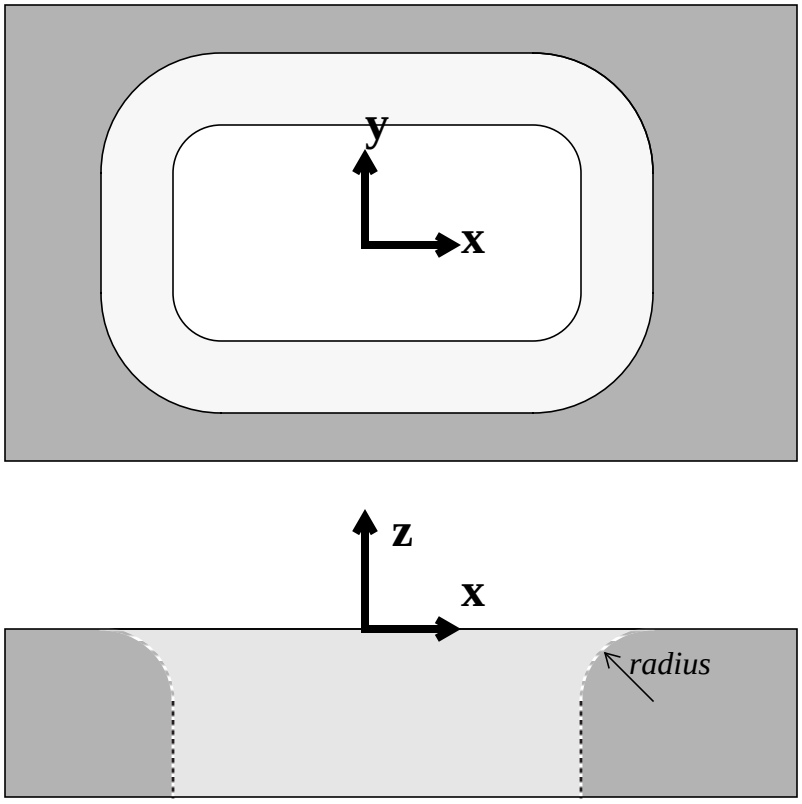
This figure shows an edge flat on part of the outer edge of a block. The coordinate system of the edge_flat is shown. The triangle that would be swept to produce the edge_flat volume is shown cross-hatched on the front view of the block. Since the triangle always points to the right of the profile, the triangle would travel away from the viewer while it was sweeping. The origin of the profile has been placed (arbitrarily) at the first point of the profile and aligned with the sides of the block. The profile is shown with a dotted line on the top view.



attribute of edge_flat	type
location	axis_placement
profile	general_profile
end_radius	optional non-negative real number
depth	positive real number
angle	positive real number

Figure 14. Block with Edge_Flat

This figure shows an edge round on the inside edge of a rectangular_pocket which goes all the way through a block. The coordinate system of the edge_round is shown. The round is a full quarter round, so that *clearance* and *standoff* are both zero. The *radius* of the edge_round is shown on the front view. The edge_round uses the profile of the pocket, which has the shape of the inner "racetrack" on the top view.



attribute of edge_round	**type**
location	axis_placement
profile	general_profile
end_radius	optional non-negative real number
radius	positive real number
clearance	non-negative real number
standoff	non-negative real number

Figure 15. Pocket with Edge_Round

This figure shows a side view of a rounding tool, crosshatched, cutting a round on the edge of a workpiece. *Clearance* and *standoff* are both positive. The coordinate system of the "triangle" which sweeps out the removed volume is shown. The "triangle" is shown in light shading. The tool is moving out of the plane of the picture toward the viewer.

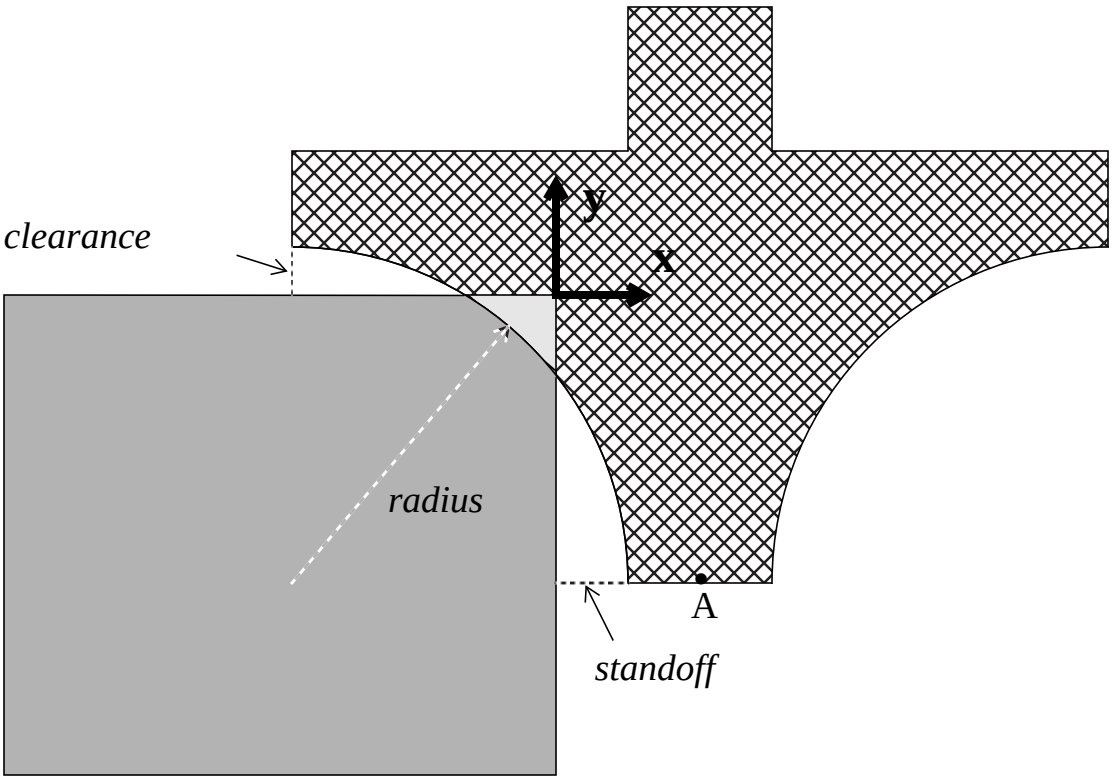


Figure 16. Edge_Round with Cutter

3.8.6 Rotation_Pocket

A rotation_pocket is a solid of revolution, where the rotation is through a full circle. There are two subtypes, vertical_rotation_pocket and horizontal_rotation_pocket, according to whether the axis of rotation is parallel to the z-axis of the native coordinate system of the pocket or perpendicular to it. Both types use a nice_open_profile, defined in section 3.3.2, to generate the solid of revolution. A vertical_rotation_pocket includes the entire solid of revolution. A horizontal_rotation_pocket consists of half or less of the solid; part of it is cut off by a plane parallel to the axis, as described below.

If a rotation_pocket can be machined nominally exactly, the value of the surface_tolerance attribute may be zero. Otherwise, that value is positive and gives the maximum allowable distance between the nominal surface of the pocket as designed and the nominal surface of the part as machined.

For a rotation_pocket, the line L, described in section 3.3.2 with regard to a nice_open_profile, is the x-axis of the native coordinate system of the profile, the origin of that system is the point P, and the point Q lies on the positive side of the origin. The profile lies below the x-axis in that system.

3.8.6.1 Vertical_rotation_pocket

An example of a vertical_rotation_pocket is shown in Figure 17.

The x-axis of the native coordinate system of the profile is coincident with the x-axis of the pocket, and the y-axis of the profile is coincident with the z-axis of the pocket. The axis of rotation is the z-axis.

The first segment of the profile may be coincident with part of the z-axis of the pocket. In this case, rotating the profile will cause that segment to lie inside the resulting solid. This has no importance.

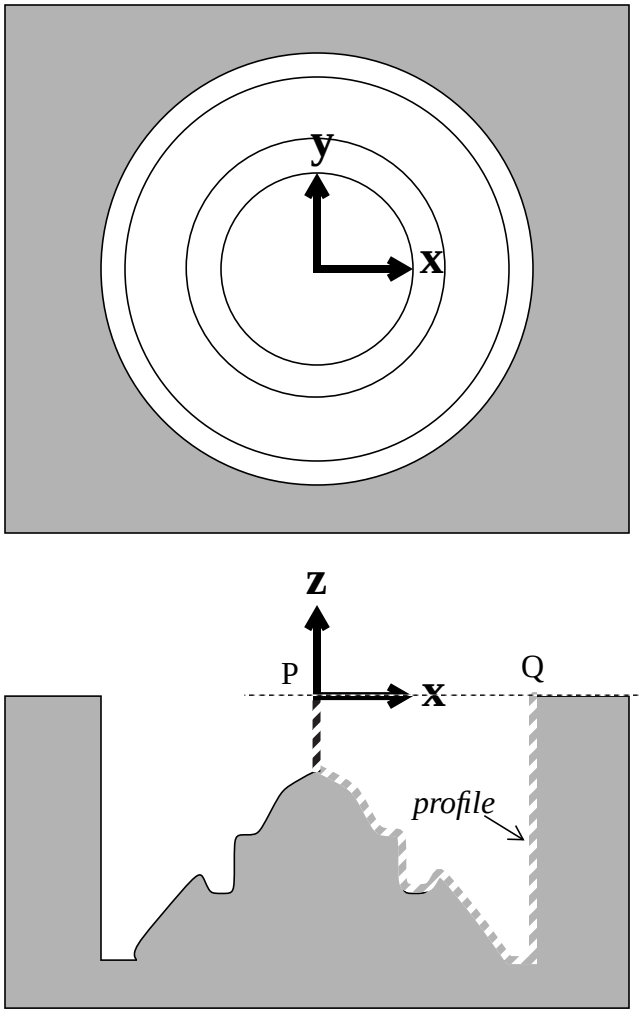
A vertical_rotation_pocket is bounded by the xy-plane of the pocket. This is equivalent to closing the nice_open_profile with a line from Q to P before rotating it to generate the solid.

3.8.6.2 Horizontal_Rotation_Pocket

An example of a horizontal_rotation_pocket is shown in Figure 18.

The y-axis of the native coordinate system of the profile lies on top of the z-axis of the pocket, but the origin of the coordinate system of the profile is a distance equal to the value of the *axis_height* attribute up that z-axis. The x-axis of the profile is parallel to the x-axis of the pocket. The axis of rotation is the line on which the x-axis of the profile lies. Only that portion of the solid of rotation which lies below the xy plane of the coordinate system of the pocket is the pocket.

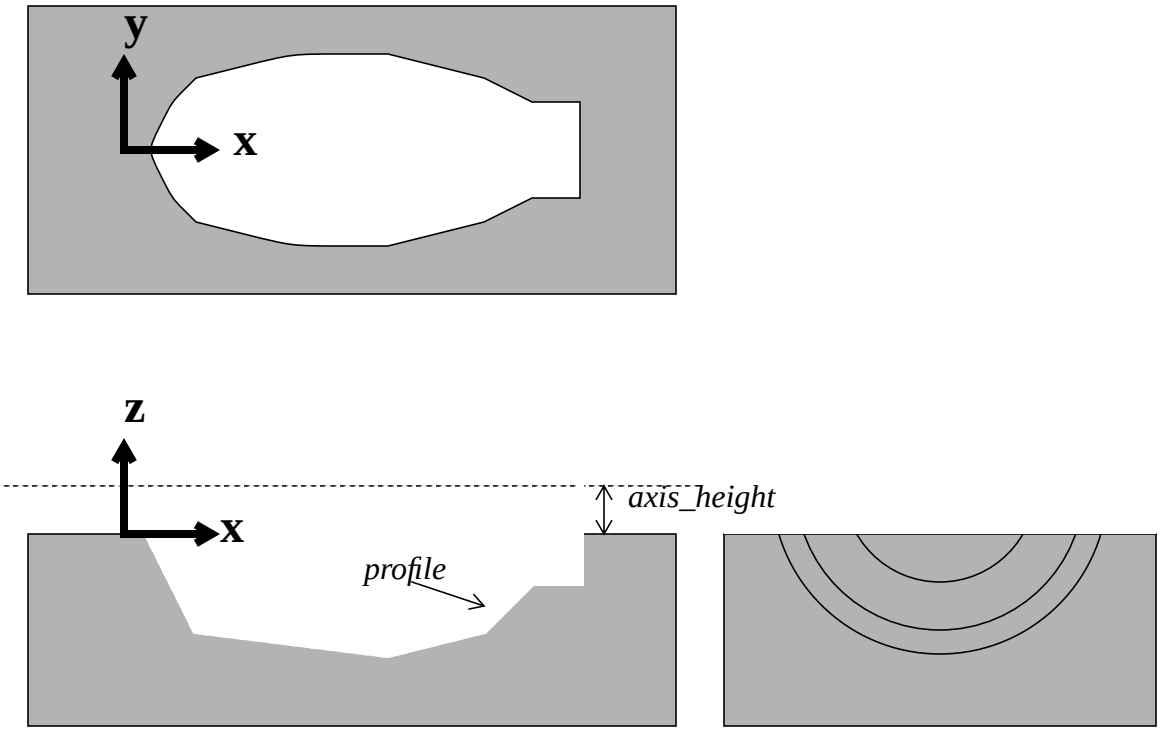
This figure shows a vertical_rotation_pocket. The profile used to generate the pocket is shown with a heavy dotted line. The line L and points P and Q described in the definition of a nice_open_profile, are shown on the front view of the pocket.



attribute of pocket	**type**
location	axis_placement
profile	nice_open_profile
surface_tolerance	non-negative real number

Figure 17. Vertical_Rotation_Pocket

This figure shows three orthographic views of a horizontal_rotation_pocket. Only that portion of the solid of rotation that is below the xy-plane is the pocket.



attribute of pocket	**type**
location	axis_placement
profile	nice_open_profile
surface_tolerance	non-negative real number
axis_height	non-negative real number

Figure 18. Horizontal_Rotation_Pocket

3.8.7 Islands

Islands were discussed in section 3.6. As mentioned earlier, each type of island is very similar to what you would get by using some type of mrsev_volume as a mold, casting the island, and turning it upside down. Anti_groove_island corresponds to groove, vertical_rotation_island corresponds to vertical_rotation_pocket, and horizontal_rotation_island corresponds to horizontal_rotation_pocket.

All types of island are required to lie wholly within the parent pocket, except that the top of the island may extend through the top of the pocket. As mentioned earlier, any part of the island extending through the top of the pocket has no effect, since the island is boolean subtracted from the pocket. Any extension of an island caused by an edge blend is considered to be part of the island. The islands in a pocket may not intersect each other. The bottom of each type of island is flat, and the native coordinate system of each type of island has its xy-plane on the bottom of the island.

An island is located by putting it flat on the bottom of its parent pocket. The origin of the native coordinate system of the island is placed at the point specified by *location_x* and *location_y*, which are the x- and y-coordinates of the point in terms of the native coordinate system of the pocket. The x-axis of the coordinate system of the island is rotated counterclockwise from the x-axis of the pocket by the *location_angle* of the island, which is specified in degrees.

This location system forces the z-axis of the native coordinate system of an island to be parallel to the z-axis of the native coordinate system of its parent pocket.

In order that MRSEVs can be used effectively to carry shape information in machining process plan steps, it must be possible to machine any island with the same tool used to machine its parent pocket. This implies that blends on islands or their parent pockets will have to be assigned carefully to achieve machinability. Because a blend may be implicit in the definition of an island, it is not always necessary to have an explicit blend for an island if the parent pocket has an explicit blend.

The next three figures show subtypes of islands. Mesa_islands are shown in Figure 7 and are not repeated here. Each figure shows an island sitting on a flat surface. The parent pockets of the islands are not shown. The islands are shown with lighter shading than the surfaces on which they sit. The coordinate systems shown are the coordinate systems of the islands.

3.8.7.1 Anti_Groove_Island

Figure 19 shows an anti_groove_island.

The coordinate system of an anti_groove_island is the native coordinate system of the profile of the island with a z-axis added to form a right-handed system. The profile must meet the constraints listed below.

The nice_open_profile which is the value of the *section* attribute of the island and gives the cross section of the island, is located with points P and Q (described in section 3.3.2) on the x-axis of the native coordinate system of the profile. P and Q must lie on opposite sides of the origin. We will let Q be the one lying on the plus-x side. The line PQ serves to close the profile. The profile lies above the x-axis.

In order to form the island, the origin of the coordinate system for the cross section is dragged along the profile from the first point to the last, sweeping the cross section through space. The swept volume is the island. During the dragging, the y-axis of the cross section is oriented parallel to the z-axis of the coordinate system of the island and

the z-axis of the cross section is kept pointing in the direction of travel. That direction is defined by the positive direction of the profile and the tangent to the profile at the point where the cross section is attached to the profile.

The profile of the island must meet the following restrictions (so the island will be machinable): (i) the profile may not have any sharp corners, (ii) arcs which bend to the left may not have a radius of less than the x-value of Q plus the extent of any bottom_blend, and (iii) arcs which bend to the right may not have a radius of less than the absolute value of the x-value of P plus the extent of any bottom_blend.

The swept volume must not intersect itself.

The optional *bottom_blend* attribute may be used to specify a blend between the island and the bottom of the pocket.

If the profile is open, the ends of the island are defined to be flat. If a *bottom_blend* is specified, it will apply to the edges between the ends of the island and the bottom of the pocket, as well as to the other edges where the island meets the bottom.

3.8.7.2 Horizontal_Rotation_Island

An example of a horizontal_rotation_island is shown in Figure 20.

The y-axis of the native coordinate system of the profile lies on top of the z-axis of the island, but the origin of the coordinate system of the profile is a distance equal to the value of the *axis_depth* attribute down that z-axis. The x-axis of the profile is parallel to the x-axis of the island. The axis of rotation is the line on which the x-axis of the profile lies. Only that portion of the solid of rotation which lies above the xy plane of the coordinate system of the island is the island.

3.8.7.3 Vertical_Rotation_Island

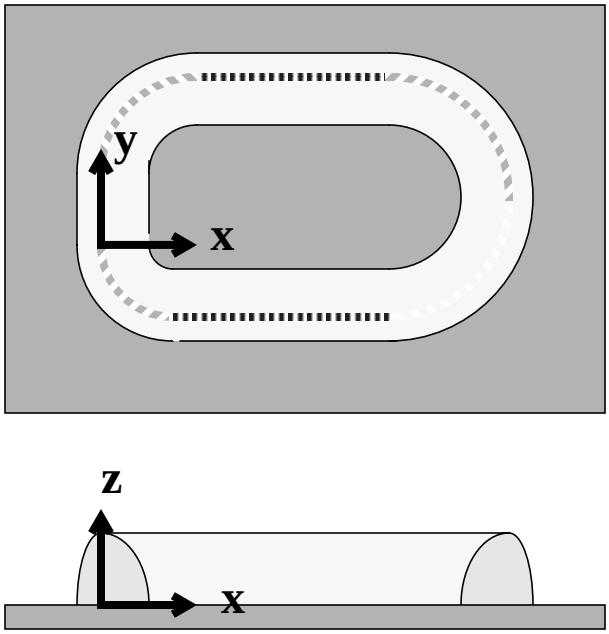
An example of a vertical_rotation_island is shown in Figure 21.

The x-axis of the native coordinate system of the profile is coincident with the x-axis of the island, and the y-axis of the profile is coincident with the z-axis of the island. The axis of rotation is the z-axis of the island. The profile lies above the x-axis in its native coordinate system.

The first segment of the profile may be coincident with part of the z-axis of the island. In this case, rotating the profile will cause that segment to lie inside the resulting solid. This has no importance.

A vertical_rotation_island is bounded by the xy-plane of the island. This is equivalent to closing the nice open profile with a line from Q to P before rotating it to generate the solid.

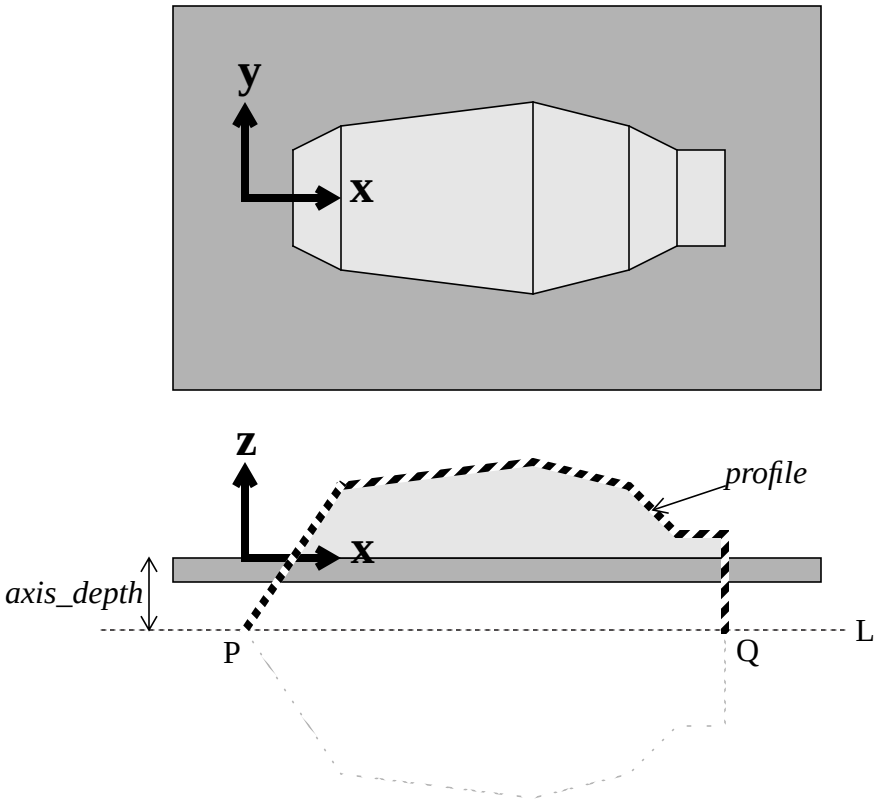
This figure shows an anti_groove_island made with a closed profile. A cross section of the island is shown in the front view so that the shape of the nice_open_profile which is the value of the island's section attribute can be seen.



attribute of anti_groove_island	**type**
location_x	real number
location_y	real number
location_angle	real number
bottom_blend	optional edge_blend
profile	general_profile
section	nice_open_profile
surface_tolerance	optional non-negative real number

Figure 19. Anti_Groove_Island

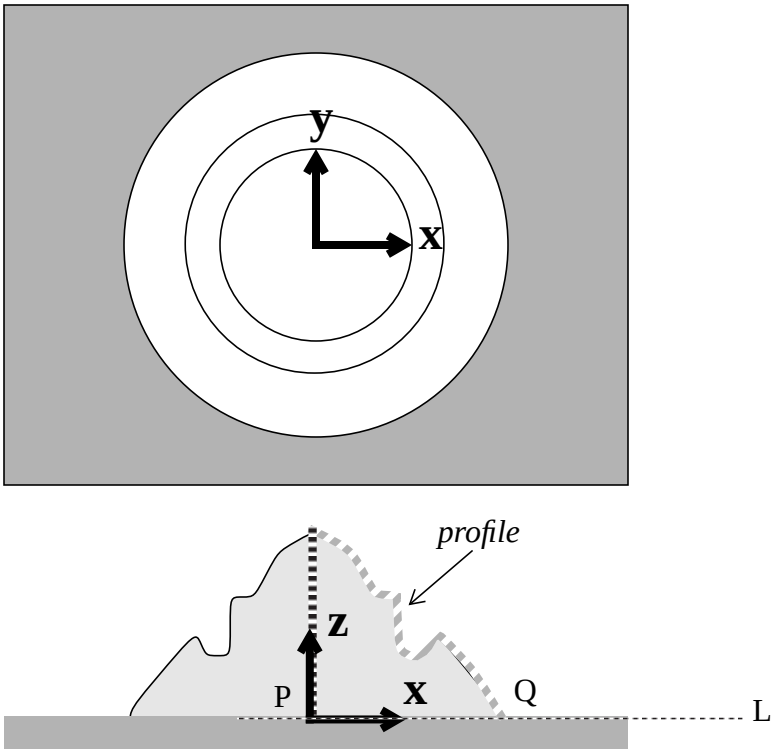
This figure shows a horizontal_rotation_island. The full solid of revolution is shown with a faint outline on the front view, and the axis of rotation is shown with a dotted line on the same view.



attribute of island	type
location_x	real number
location_y	real number
location_angle	real number
bottom_blend	optional edge_blend
profile	nice_open_profile
surface_tolerance	optional non-negative real number
axis_depth	non-negative real number

Figure 20. Horizontal_Rotation_Island

This figure shows a vertical_rotation_island.



attribute of island	**type**
location_x	real number
location_y	real number
location_angle	real number
bottom_blend	optional edge_blend
profile	nice_open_profile
surface_tolerance	optional non-negative real number

Figure 21. Vertical_Rotation_Island

3.8.8 Ramp

A ramp is the shape resulting from sweeping an end mill either (i) in a straight line which does not lie in the plane perpendicular to the axis of the tool or (ii) in an arc of a helix whose axis is parallel to the axis of the tool. The end-mill may have a flat end or a ball-nosed end.

The surfaces of all mrsev_volumes and islands except ramps are planes or surfaces of elementary solids (sphere, cylinder, cone, torus). Some of the surfaces of some ramps, however, are not.

Ramps are included in the MRSEV library because it is easy to make them with a 3-axis machining center. Machining centers can also make cuts which are arcs of circles or helixes for which the axis lies in a plane perpendicular to the axis of the cutting tool. Shapes resulting from such cuts are not included in this library.

The attributes of a ramp always include the *width* of the ramp (the diameter of the end-mill), the *start_depth* and *end_depth* of the ramp, and the *bottom* type of the ramp, which may be either “flat” or “round”, corresponding to a flat ended end-mill or a ball-nosed end-mill. *End_depth* must be greater than *start_depth*.

If the tool is ball-nosed, *start_depth* must be at least as large as half the *width*. It might be useful to delete this restriction, allowing ramps which vary in width.

3.8.8.1 Straight_Ramp

An example of a straight_ramp is shown in Figure 22.

A straight_ramp starts with the axis of the end-mill coincident with the z-axis of the native coordinate system of the ramp, with the tip of the tool a distance *start_depth* below the x-axis. The ramp ends with the tip of the tool on the xz-plane a distance *end_depth* below the point on the x-axis where x equals the *length* of the ramp. The tip of the tool moves in a straight line between those two points, and the axis of the tool is kept parallel to the z-axis of the ramp. The xy-plane forms the top of the ramp.

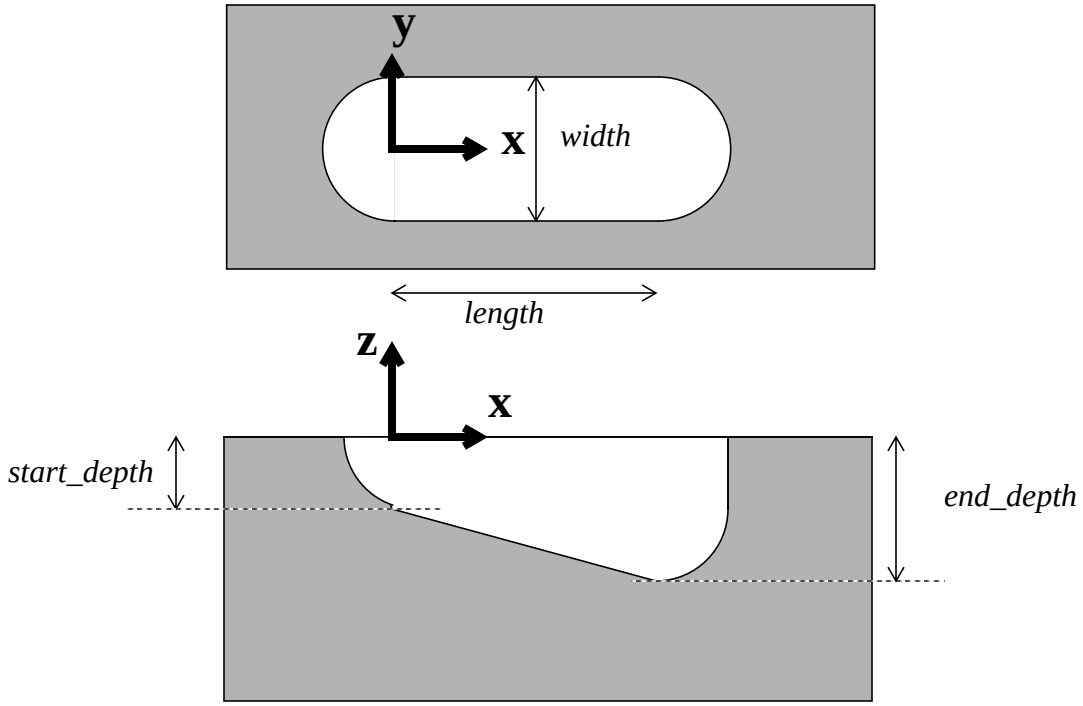
If the tool is ball-nosed, all the surfaces of the ramp are elementary. If the tool has a flat end, a portion of the bottom surface of the ramp is that of an elliptical cylinder.

3.8.8.2 Circular_Ramp

An example of a circular_ramp is shown in Figure 23.

The axis of the helical arc (which is the path of the tip of the cutter) lies on top of the z-axis of the native coordinate system of the ramp. The tip of the cutter starts a distance *start_depth* below the point on the x-axis where x equals *radius*. The tip of the cutter then travels along the helical arc through an angle specified in degrees by the value of the *angle* attribute. The angle is positive measured counterclockwise. The *radius* of the ramp must be greater than half the *width* of the ramp (this requirement might be relaxed, but different topology results if it is violated). The axis of the tool is kept parallel to the z-axis of the ramp. The xy-plane forms the top of the ramp.

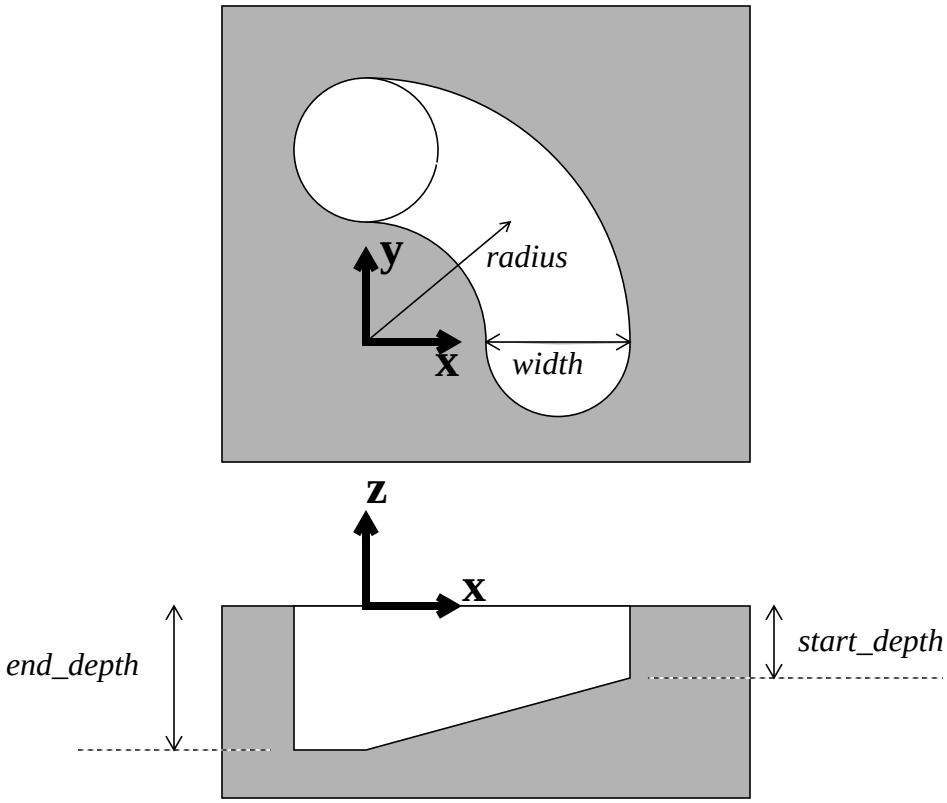
This figure shows a straight_ramp with a round bottom.



attribute of straight_ramp	**type**
location	axis_placement
width	positive real number
start_depth	non-negative real number
end_depth	positive real number > start_depth
bottom	flat or round
length	positive real number

Figure 22. Straight_Ramp

This figure shows a circular ramp with a flat bottom. The *angle* of the ramp is 90 degrees.



attribute of circular_ramp	**type**
location	axis_placement
width	positive real number
start_depth	non-negative real number
end_depth	positive real number > start_depth
bottom	flat or round
radius	positive real number
angle	non-zero real number

Figure 23. Circular_Ramp

A References

[Altemueller88a] Altemueller, J., The STEP File Structure, ISO TC184/SC4 Document N279, September, 1988

[Altemueller88b] Altemueller, J., Mapping from EXPRESS to Physical File Structure, ISO TC184/SC4 Document N280, September, 1988

[ANSI] American National Standards Institute, Industrial Engineering Terminology, Production Planning and Control, ANSI, 1973, p. 16

[Cain89] Cain, W. D., NC Process Information Model Version 1.0, unpublished, February, 1989

[Dunn88] Dunn, M., Form Features Information Model, ISO TC184/SC4 Draft Proposal, October, 1988

[Dunn92] Dunn, M., Industrial Automation Systems - Product Data Representation and Exchange - Part 48: Integrated Generic Resources: Form Features, ISO TC184/SC4 Document N102, January, 1992

[Goult91] Goult, R., Product Data Representation and Exchange - Part 42: Integrated Resources: Geometric and Topological Representation, ISO TC184/SC4 Document N87, June, 1991

[Kramer87] Kramer, T.R., Process Plan Expression, Generation, and Enhancement for the Vertical Workstation Milling Machine, NBSIR 87-3678, National Institute of Standards and Technology, November, 1987

[Kramer89] Kramer, T.R., A Parser that Converts a Boundary Representation into a Features Representation, *International Journal of Computer Integrated Manufacturing*, Vol. 2, No. 3, May - June 1989, pp 154-163

[Kramer91a] Kramer, T.R., The Off-Line Programming System (OLPS): A Prototype STEP-Based NC-Program Generator, proceedings of a seminar *Product Data Exchange for the 1990s*, New Orleans, Louisiana, NCGA, February, 1991, Vol. 2

[Kramer91b] Kramer, T.R., An EXPRESS Schema for Machining Plugged into ALPS4, unpublished, January, 1991

[Kramer92a] Kramer, T.R., Issues Regarding Material Removal Shape Element Volumes, NISTIR 4804, National Institute of Standards and Technology, March, 1992

[Kramer92b] Kramer, T.R., Pocket Milling with Tool Engagement Detection, *Journal of Manufacturing Systems*, scheduled for publication in Vol. 11, No. 2, March, 1992

[Paul91] Paul, G., Process Plan Model, ISO TC184/SC4 Document N95, October, 1991

[Ray91] Ray, S. and Catron, B., ALPS - A Language for Process Specification, *International Journal of Computer Integrated Manufacturing*, Vol. 4, No. 2, pp. 105-113

[Schenck90] Schenck, D., ed., Exchange of Product Model Data - Part 11: The EXPRESS Language, ISO TC184/SC4 Document N64, July, 1990

[Spiby91] Spiby, P., ed., Exchange of Product Model Data - Part 11: The EXPRESS Language, ISO TC184/SC4 Document N14, April, 1991

[Van Maanen91] Van Maanen, J., Product Data Representation and Exchange - Part 21: Clear Text Encoding of the Exchange Structure, ISO TC184/SC4 Document N78, March, 1991

B Prototype EXPRESS Schema for MRSEVs

A prototype EXPRESS schema follows, which defines the subset of the MRSEV library which is currently in use in the Off-Line Programming System, namely the linear_sweep subtypes shown in Figure 2 (holes and pockets). Only the “mesa” subtypes of island shown in Figure 3 and only the “replication” subtype of mrsev_copy_method shown in Figure 4 are included.

This prototype uses the “Tokyo” versions of the EXPRESS language, the Form Features Information Model (FFIM), the Geometry Model, and the Topology Model. In a few cases, the FFIM has been assumed to have been modified slightly to solve technical problems in using it as a resource. In a few other cases, the EXPRESS “map” facility has been used in an improper fashion, also to deal with technical problems. Thus, this prototype is not in complete conformance with the Tokyo versions and should be regarded only as being illustrative of how resource models might be used to build application models in STEP.

Moreover, EXPRESS, the FFIM, the Geometry Model, and the Topology Model have all changed since the Tokyo version. To put this prototype in conformance with the current versions would require extensive revisions.

The general approach taken to building the prototype is to draw heavily on entities defined in the FFIM. EXPRESS provides the “map” function for defining new entities in terms of existing ones, and “map” has been used as the principal link between the FFIM and the prototype.

In some instances, the FFIM uses a general version of entity A in defining entity B, while the prototype requires a specialized subtype S of entity A in defining mrsev_volume type C which corresponds to FFIM entity B. In such cases, a “where” clause is added to the mapping of B to C to identify the subtype S. For example, the FFIM uses feature_sweep_path (A) in defining in_out_feature_sweep (B). The prototype defines pocket (C) as an in_out_feature_sweep where the feature_sweep_path is a linear_feature_sweep_path (S).

Because of the requirement of the FFIM that each form_feature be classified as a depression, protrusion or passage, which is not desirable in the prototype, the FFIM form_feature entity is not used in this prototype. Two hierarchical levels below form_feature, the FFIM defines “feature_volume”. Feature_volumes exist without regard to any parent part. Feature_volume is used in the prototype as the top of the hierarchy of mrsev_volumes.

In the prototype, the “primary_mrsev_volume” entity on Figure 2 is not used as an intermediate subtype. The prototype has “linear_sweep” (under the name “feature_sweep” inherited from the FFIM) as a direct subtype of “mrsev_volume” (under the name “feature_volume”, also inherited from the FFIM).

The prototype has several changes in names of entities in the library. These are listed below. The names of several attributes are also changed because in the prototype many names are inherited from the FFIM. The names used in the library were chosen to be as descriptive as possible, without regard to how the library might be built using STEP resource models.

LIBRARY NAME	**PROTOTYPE SCHEMA NAME**
mrsev	material_removal_volume
mrsev_copy_method	removal_volume_copy_method
mrsev_volume	feature_volume
other_pocket_no_islands	other_pocket
rectangular_pocket_no_islands	rectangular_pocket
replication	removal_volume_replication

Figures B1 and B2 show how the prototype is built from the resource models. The island hierarchy defined in the prototype is not included in the figures.

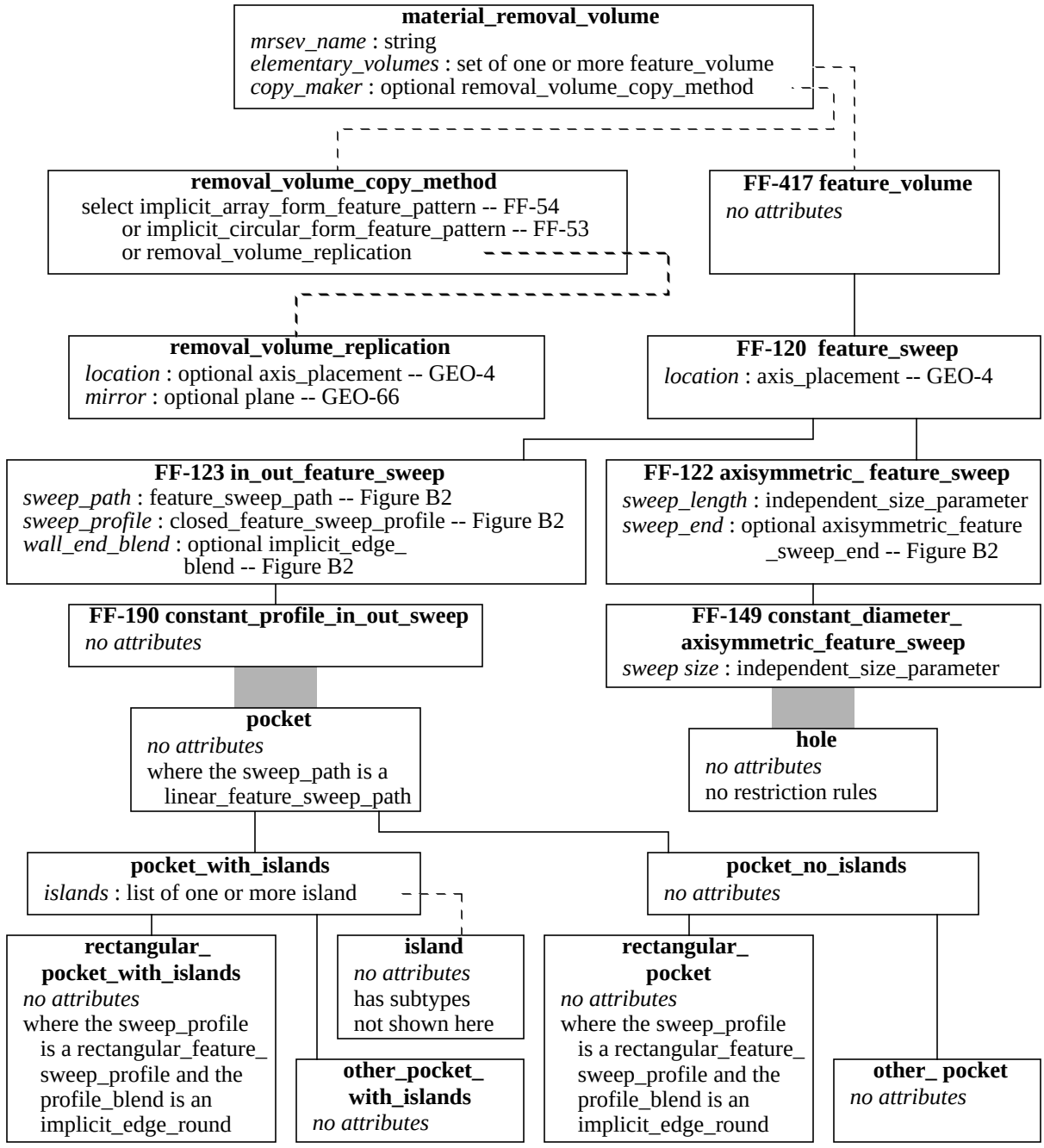


Figure B1. Hierarchy of MRSEVs in Prototype

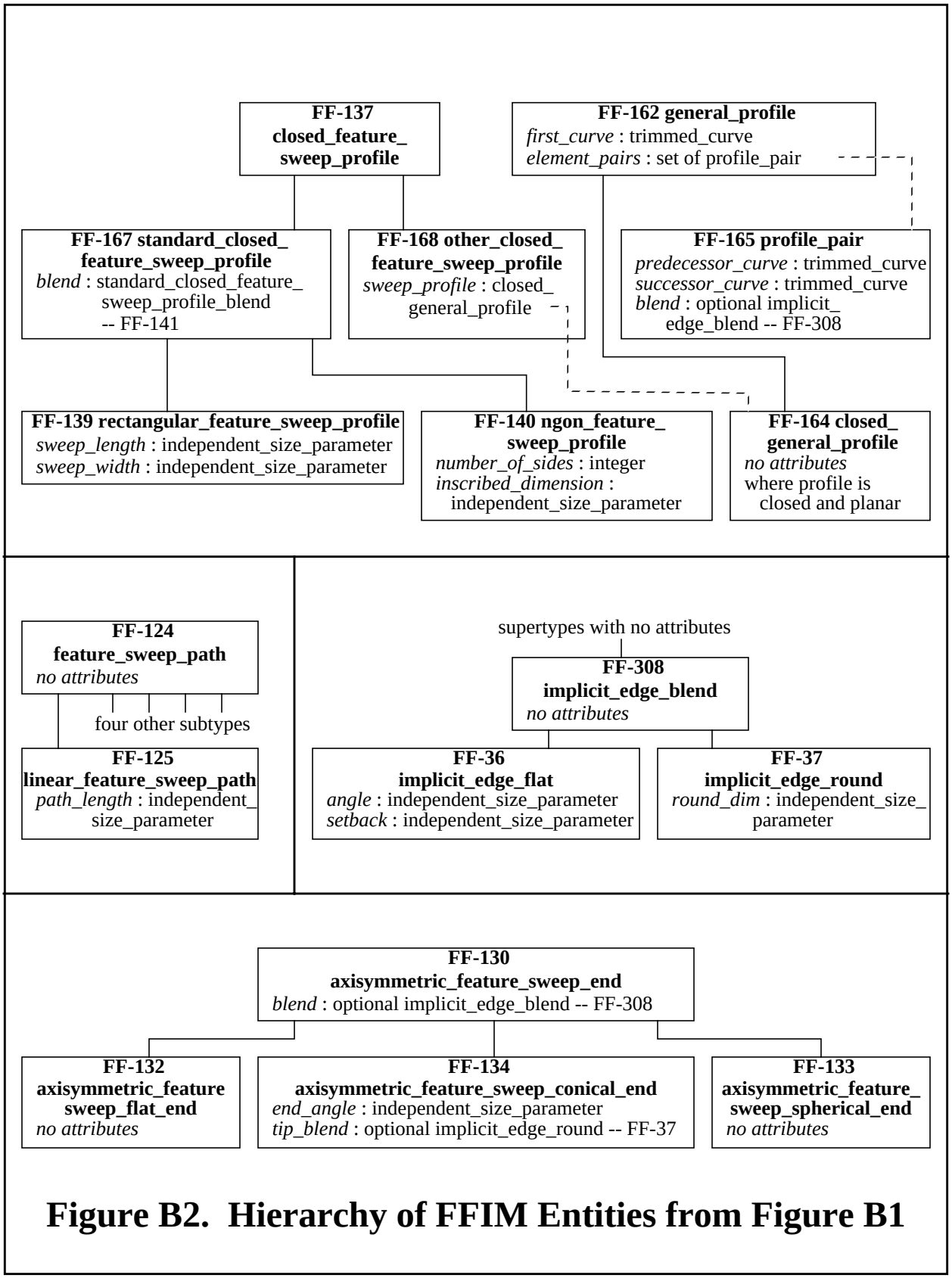


Figure B2. Hierarchy of FFIM Entities from Figure B1

```
SCHEMA Material_Removal_Shape_Element_Volumes
 ASSUME (geometric_models, form_features);
   USES (axis_placement,                                -- GEO-4
         cartesian_three_point,                         -- GEO-11
         constant_diameter_axisymmetric_feature_sweep,  -- FF-149
         constant_profile_in_out_sweep,                 -- FF-190
         feature_sweep,                                 -- FF-120
         feature_volume,                                -- FF-417
         implicit_array_form_feature_pattern,           -- FF-54
         implicit_circular_form_feature_pattern,        -- FF-53
         implicit_edge_round,                           -- FF-37
         line,                                          -- GEO-20
         linear_feature_sweep_path                      -- FF-125
         plane,                                         -- GEO-66
         rectangular_feature_sweep_profile,             -- FF-139
         sweep_path                                     -- attribute name from FF-123
         sweep_profile                                  -- attribute name from FF-123
          );

 MAP hole FROM constant_diameter_axisymmetric_feature_sweep;
 END_MAP;


 ENTITY island
   SUPERTYPE OF (mesa_island);
 END_ENTITY;


(* This is the top-level entity of the MRSEV schema. *)

 ENTITY material_removal_volume;
   mrsev_name           : STRING
   elementary_volumes   : SET [1:#] OF feature_volume;
   copy_maker           : OPTIONAL removal_volume_copy_method;
 END_ENTITY;


 MAP mesa_island FROM constant_profile_in_out_sweep
   SUPERTYPE OF (rectangular_mesa_island XOR
                 other_mesa_island);
   WHERE
    (TYPEOF sweep_path) = linear_feature_sweep_path;
 END_MAP;
```

```
ENTITY other_mesa_island
  SUBTYPE OF (mesa_island);
END_ENTITY;


ENTITY other_pocket
  SUBTYPE OF (pocket_no_islands);
END_ENTITY;


ENTITY other_pocket_with_islands
  SUBTYPE OF (pocket_with_islands);
END_ENTITY;


MAP pocket FROM constant_profile_in_out_sweep
  SUPERTYPE OF (pocket_no_islands XOR
                pocket_with_islands);
  WHERE
    (TYPEOF sweep_path) = linear_feature_sweep_path;
END_MAP;


ENTITY pocket_no_islands
  SUBTYPE OF (pocket)
  SUPERTYPE OF (rectangular_pocket XOR
                other_pocket);
END_ENTITY;


ENTITY pocket_with_islands
  SUBTYPE OF (pocket)
  SUPERTYPE OF (rectangular_pocket_with_islands XOR
                other_pocket_with_islands);
  islands  : LIST [1:#] of island;
END_ENTITY;


ENTITY rectangular_mesa_island
  SUBTYPE OF (mesa_island);
  WHERE
    (TYPEOF sweep_profile) = rectangular_feature_sweep_profile;
    (TYPEOF sweep_profile.blend) = implicit_edge_round;
END_ENTITY;
```

```
ENTITY rectangular_pocket
  SUBTYPE OF (pocket_no_islands);
  WHERE
    (TYPEOF sweep_profile) = rectangular_feature_sweep_profile;
    (TYPEOF sweep_profile.blend) = implicit_edge_round;
END_ENTITY;


ENTITY rectangular_pocket_with_islands
  SUBTYPE OF (pocket_with_islands);
  WHERE
    (TYPEOF sweep_profile) = rectangular_feature_sweep_profile;
    (TYPEOF sweep_profile.blend) = implicit_edge_round;
END_ENTITY;


TYPE removal_volume_copy_method =
  SELECT (implicit_array_form_feature_pattern,
            implicit_circular_form_feature_pattern,
            removal_volume_replication);
END_TYPE;


ENTITY removal_volume_replication;
  location              : OPTIONAL axis_placement;
  mirror                : OPTIONAL plane;
END_ENTITY;


END_SCHEMA;
```

C An Example of Using MRSEVs

The Off-Line Programming System (OLPS) developed at NIST between 1988 and 1990, is a system for generating NC-programs, in the context shown in Figure 1 [Kramer91a]. OLPS uses all the MRSEV types defined in Appendix B.

Machining operations currently implemented in OLPS are: center-drill, twist-drill, counterbore, rough-mill, finish-mill, initialize-plan, close-plan, and set0-corner. Many other operations are defined but not yet implemented.

Some types of operations can use more than one type of MRSEV. Rough-mill and finish-mill, for example, can refer to any of the four types of pockets defined in Appendix B. On the other side of the coin, some types of MRSEVs can be named by many types of operations. Center-drill, twist-drill, and counterbore, for example, all act on holes. If all three of those operations are required to make a single hole, three different MRSEV holes will normally be used, one for each operation. If the drilled hole were nominally identical to the counterbored hole, however, only one MRSEV hole would be required for the two operations.

As shown in Figure 1, OLPS (which is the “NC-PROGRAMMING” box shown in the figure) requires six kinds of input data, all of which are entered using STEP physical files. The data for the design, the workpiece, and the fixture are in boundary representation format using STEP geometry and topology entities. The process plan is prepared using a schema for the ALPS language [Ray91] with added entities defining machining operations [Kramer91b]. MRSEV data is prepared according to the MRSEV schema in Appendix B, with the slight modification that real numbers are used in place of independent_size_parameters. The setup file currently being used contains 18 specific STEP geometry entities.

OLPS includes its own STEP file reader but does not include an EXPRESS parser. Rather, it requires that information equivalent to the entity definitions in the schemas it uses be entered in a LISP-readable database.

A part, named “clevis2”, that may be cut on a 3-axis machining center using NC-programs written automatically by OLPS is shown in Figure C1. It requires three fixturings. The entire process plan for the second fixturing is shown in Figure C2. A portion of the MRSEV file for the second fixturing is shown in Figure C3. The data for only three of the MRSEVs is shown in Figure C3, namely the MRSEVs whose pictures are in Figures C4 and C5. The MRSEV named “other_pocket31” is shown in Figure C4, where the MRSEV is shown in heavy lines superimposed on a picture of the part (light solid lines and the workpiece (light dashed lines). Figure C5 shows the MRSEVs named “rectangular_pocket23” and “hole30” in a similar view.

The part pictures in the figures are screen images from OLPS. The pictures show wire frame drawings of the edges of the objects. A STEP format boundary representation for each MRSEV is generated automatically by OLPS in order to do the drawing.

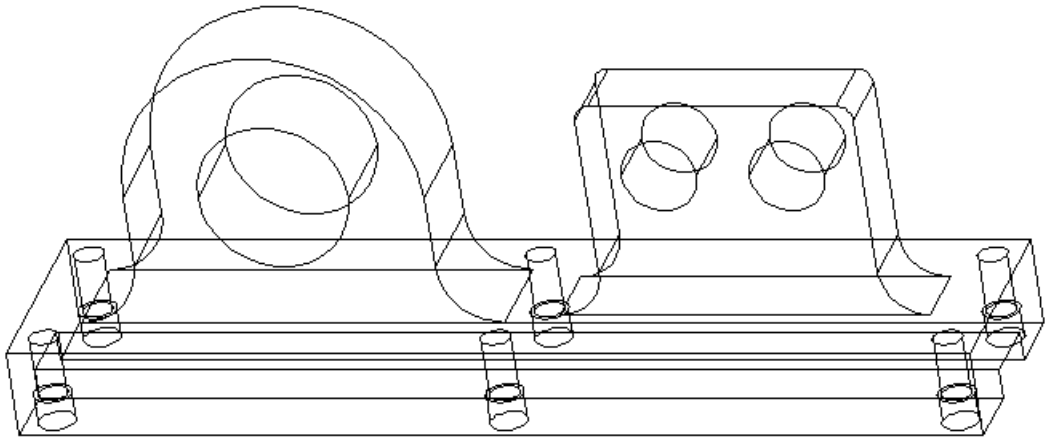


Figure C1. Design of the "Clevis2" Part

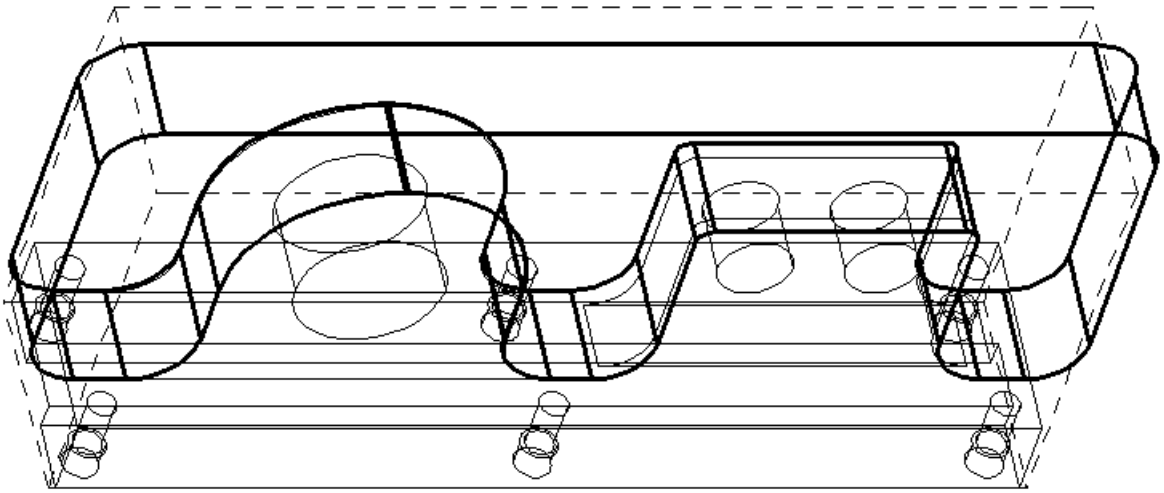
```
STEP;
HEADER;
 FILE_IDENTIFICATION(
   'CLEVIS2_PLAN2',
   '19910115000000',
   ('Tom Kramer',
    'Room A-127, Bldg. 220',
    'NIST',
    'Gaithersburg, MD 20899'),
   ('Factory Automation Systems Division',
    'NIST',
    'Gaithersburg, MD 20899'),
   'STEP Version 1.0',
   'No preprocessor - handwritten',
   'No system');
 FILE_DESCRIPTION('This is a machining process
plan for clevis2 second cut');
 IMP_LEVEL('1.0');
ENDSEC;

DATA;
@1 = start_plan ('clevis2 second cut', '1',
    'start_plan', '1', (), );
@2 = initialize_program ('set up program', '2',
    'initialize_program', '1', (), #1, ,
    'initialize_program', (), '10', , , , , 'cl28nc',
    'clevis2-design', 'clevis2-mrsevs2', 'clevis2-
    setup2', 'block2', 'aluminum', 'vise', 17.1, 7.3, );
@3 = set0_corner ('locate the part', '3', 'set0_corner',
    '1', (), #2, , 'set0_corner', (), '10', , , , , 'probe-
    0.25', 0.0, 0.0, 0.0, 0.0, 1);
@4 = parameterized ('split', '4', 'parameterized', '1',
    (), #3, , 'parallel', 'all', ());
@5 = rough_mill ('rough pocket 21', '5',
    'rough_mill', '1', (), #4, , 'rough_mill', (), '10', , ,
    , , 'ruf-mill-0.625-4', 'rectangular_pocket21',
    3500, 17.0, 0.625, 0.375);
@6 = rough_mill ('rough pocket 23', '6',
    'rough_mill', '1', (), #4, , 'rough_mill', (), '10', , ,
    , , 'ruf-mill-0.625-4', 'rectangular_pocket23',
    3500, 17.0, 0.625, 0.375);
@7 = join ('join', '7', 'join', '1', (), , (#5, #6));
@8 = parameterized ('split', '8', 'parameterized', '1',
    (), #7, , 'parallel', 'all', ());
@9 = finish_mill ('finish pocket 22', '9',
    'finish_mill', '1', (), #8, , 'finish_mill', (), '10', , , ,
    , 'end-mill-0.625-2', 'rectangular_pocket22',
    3500, 17.0, 0.375);
@10 = finish_mill ('finish pocket 24', '10',
    'finish_mill', '1', (), #8, , 'finish_mill', (), '10', , , ,
    , 'end-mill-0.625-2', 'rectangular_pocket24',
    3500, 17.0, 0.375);
@11 = rough_mill ('rough pocket 29', '11',
    'rough_mill', '1', (), #8, , 'rough_mill', (), '10', , ,
    , , 'ruf-mill-0.625-4', 'rectangular_pocket29',
    3500, 17.0, 0.5, 0.375);
@12 = rough_mill ('rough pocket 31', '12',
    'rough_mill', '1', (), #8, , 'rough_mill', (), '10', , ,
    , , 'ruf-mill-0.625-4', 'other_pocket31', 3500, 17.0,
    0.625, 0.375);
@13 = center_drill ('center drill hole 1', '13',
    'center_drill', '1', (), #8, , 'center_drill', (), '10', , ,
    , , 'center-drill-0.1875-2', 'center_drill_hole1',
    5000, 8.0);
@14 = center_drill ('center drill hole 2', '14',
    'center_drill', '1', (), #8, , 'center_drill', (), '10', , ,
    , , 'center-drill-0.1875-2', 'center_drill_hole2',
    5000, 8.0);
@15 = counterbore ('counterbore hole30', '15',
    'counterbore', '1', (), #11, , 'counterbore', (), '10',
    , , , , 'end-mill-1.0-2', 'hole30', 2500, 6.0);
@16 = finish_mill ('finish pocket 32', '16',
    'finish_mill', '1', (), #12, , 'finish_mill', (), '10', , ,
    , , 'end-mill-0.5-3', 'other_pocket32', 3500, 17.0,
    0.25);
@17 = twist_drill ('drill hole1', '17', 'twist_drill', '1',
    (), #13, , 'twist_drill', (), '10', , , , , 'drill-0.5-2',
    'hole1', 2500, 3.5, 0.5);
@18 = twist_drill ('drill hole2', '18', 'twist_drill', '1',
    (), #14, , 'twist_drill', (), '10', , , , , 'drill-0.5-2',
    'hole2', 2500, 3.5, 0.5);
@19 = join ('join', '19', 'join', '1', (), , (#9, #10, #15,
    #16, #17, #18));
@20 = end_program ('end program', '20',
    'end_program', '1', (), #19, , 'end_program', (),
    '10', , , , );
@21 = end_plan ('end second cut', '21', 'end_plan',
    '1', (), #20);
@22 = plan ('equipment', 'olps', 'clevis2_plan2', '1',
    (), (#1, #2, #3, #4, #5, #6, #7, #8, #9, #10, #11, #12,
    #13, #14, #15, #16, #17, #18, #19, #20, #21));
ENDSEC;
ENDSTEP;
```

Note: Letters in data section are actually all upper case. Lower case used here for readability and size.

Figure C2. Process Plan for Second Cut on Clevis2

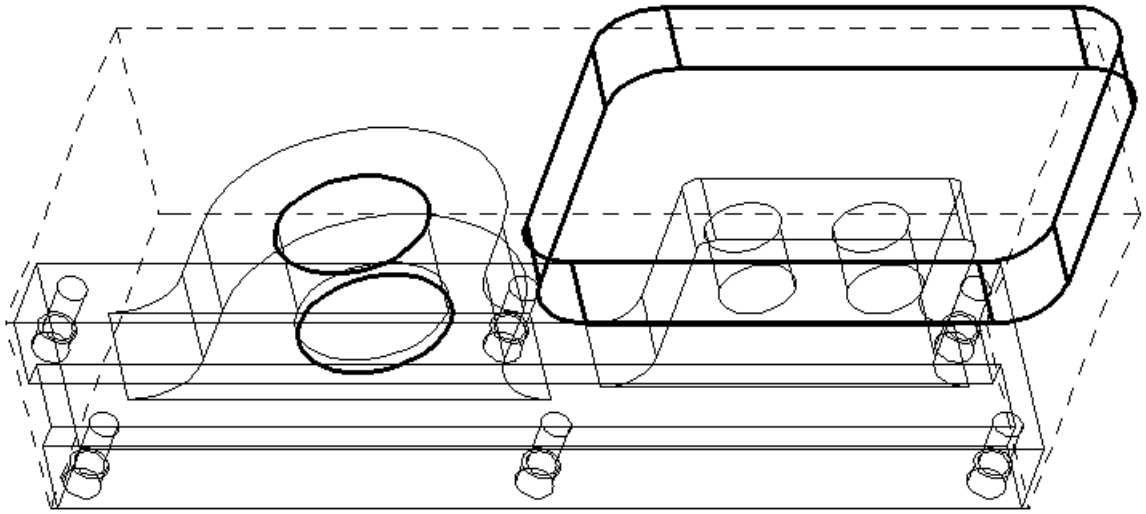
```
STEP;
HEADER;
  (data deleted)
ENDSEC;
DATA;
@2 = DIRECTION ( , 0.0, 0.0, 1.0);
@3 = DIRECTION ( , 1.0, 0.0, 0.0);
@214 = IMPLICIT_EDGE_ROUND (0.4);
@231 = CARTESIAN_POINT ( , 5.0, 1.91, .01);
@232 = LINEAR_FEATURE_SWEEP_PATH
          (0.49);
@233 = AXIS2_PLACEMENT ( , #231, #2, #3);
@235 = RECTANGULAR_FEATURE_
          SWEEP_PROFILE (#214, 3.7, 2.6);
@236 = RECTANGULAR_POCKET (#233,
          #232, #235, );
@237 = MATERIAL_REMOVAL_VOLUME
          ('RECTANGULAR_POCKET23',
          (#236), );
/* hog off right of front */

@301 = CARTESIAN_POINT ( , 1.8, 1.5, 0.3);
@305 = AXIS2_PLACEMENT ( , #301, #2, #3);
@306 = HOLE (#305, 0.8, , 0.5);
@307 = MATERIAL_REMOVAL_VOLUME
          ('HOLE30', (#306), );
/* finish large hole in left boss */

@401 = CARTESIAN_POINT ( , -0.31, 0.61, );
@402 = CARTESIAN_POINT ( , 0.79, 0.61, );
@403 = CARTESIAN_POINT ( , 0.79, 2.51, );
@404 = CARTESIAN_POINT ( , 2.81, 2.51, );
@405 = CARTESIAN_POINT ( , 2.81, 0.61, );
@406 = CARTESIAN_POINT ( , 3.84, 0.61, );
@407 = CARTESIAN_POINT ( , 3.84, 2.11, );
@408 = CARTESIAN_POINT ( , 5.71, 2.11, );
@409 = CARTESIAN_POINT ( , 5.71, 0.61, );
@410 = CARTESIAN_POINT ( , 6.81, 0.61, );
@411 = CARTESIAN_POINT ( , 6.81, 3.11, );
@412 = CARTESIAN_POINT ( , -0.31, 3.11, );
@413 = DIRECTION ( , 1.0, 0.0, );
@414 = DIRECTION ( , 0.0, 1.0, );
@415 = LINE ( , #401, #413);
@416 = LINE ( , #402, #414);
@417 = LINE ( , #403, #413);
@418 = LINE ( , #404, #414);
@419 = LINE ( , #406, #414);
@420 = LINE ( , #407, #413);
@421 = LINE ( , #408, #414);
@422 = LINE ( , #410, #414);
@423 = LINE ( , #411, #413);
@424 = LINE ( , #412, #414);
@425 = TRIMMED_CURVE ( , #415, , , #401, #402, .T.);
@426 = TRIMMED_CURVE ( , #416, , , #402, #403, .T.);
@427 = TRIMMED_CURVE ( , #417, , , #403, #404, .T.);
@428 = TRIMMED_CURVE ( , #418, , , #404, #405, .T.);
@429 = TRIMMED_CURVE ( , #415, , , #405, #406, .T.);
@430 = TRIMMED_CURVE ( , #419, , , #406, #407, .T.);
@431 = TRIMMED_CURVE ( , #420, , , #407, #408, .T.);
@432 = TRIMMED_CURVE ( , #421, , , #408, #409, .T.);
@433 = TRIMMED_CURVE ( , #415, , , #409, #410, .T.);
@434 = TRIMMED_CURVE ( , #422, , , #410, #411, .T.);
@435 = TRIMMED_CURVE ( , #423, , , #411, #412, .T.);
@436 = TRIMMED_CURVE ( , #424, , , #412, #401, .T.);
@437 = IMPLICIT_EDGE_ROUND (0.1);
@438 = IMPLICIT_EDGE_ROUND (0.35);
@439 = IMPLICIT_EDGE_ROUND (0.4);
@440 = IMPLICIT_EDGE_ROUND (1.0);
@441 = PROFILE_PAIR (#425, #426, #439);
@442 = PROFILE_PAIR (#426, #427, #440);
@443 = PROFILE_PAIR (#427, #428, #440);
@444 = PROFILE_PAIR (#428, #429, #439);
@445 = PROFILE_PAIR (#429, #430, #438);
@446 = PROFILE_PAIR (#430, #431, #437);
@447 = PROFILE_PAIR (#431, #432, #437);
@448 = PROFILE_PAIR (#432, #433, #438);
@449 = PROFILE_PAIR (#433, #434, #438);
@450 = PROFILE_PAIR (#434, #435, #438);
@451 = PROFILE_PAIR (#435, #436, #438);
@452 = PROFILE_PAIR (#436, #425, #438);
@453 = CLOSED_GENERAL_PROFILE (#425,
          (#441,#442, #443, #444, #445, #446,
           #447, #448, #449, #450, #451, #452));
@454 = OTHER_CLOSED_FEATURE_SWEEP_
          PROFILE (#453);
@455 = CARTESIAN_POINT ( , 0.0, 0.0, 0.39);
@456 = AXIS2_PLACEMENT ( , #455, #2, #3);
@457 = LINEAR_FEATURE_SWEEP_PATH (0.72);
@458 = OTHER_POCKET (#456, #457, #454, );
@459 = MATERIAL_REMOVAL_VOLUME
          ('OTHER_POCKET31', (#458), );
/* rough contour */

ENDSEC;
ENDSTEP;
```

Figure C3. Portion of MRSEV File for Clevis2



In this figure, the MRSEV "other_pocket31" is shown with a heavy outline, superimposed on a picture of the design (light solid outline) and the workpiece (light dashed lines). The data for this MRSEV is given in Figure C3, on the lines numbered 401 through 459.

Figure C4. An Other Pocket MRSEV



In this figure, the MRSEVs "rectangular_pocket23" and "hole30" are shown with a heavy outline, superimposed on a picture of the design (light solid outline) and the workpiece (light dashed lines). The data for these MRSEVs is given in Figure C3, on the lines numbered 2 through 237 and 301 through 307, respectively.

Figure C5. Two More MRSEVs