Reference number of working document: **ISO/TC 184/SC1 N 000**

Version description: **V14 – October 2003**

Date: 2003-10-24

Reference number of document: **ISO/DIS 14649-12**

Committee identification: ISO/TC 184/SC 1/WG 7

Secretariat: DIN

Industrial automation systems and integration

Physical device control

ISO 14649 Data model for Computerized Numerical Controllers

Part 12: PROCESS DATA FOR TURNING

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: Draft International Standard
Document subtype: if applicable
Document stage: (20) Preparation
Document language: E

Information on technical or structural contents are available at the following addresses:

Convener:	Fredrich Glantschnig
	AMT Consulting
Address:	Hoehenweg 33a
	CH-5417 Untersiggenthal
Telephone:	+41 (056) 288-2039
FAX:	+41 (056) 288-3942
E-mail:	fglantschnig@swissonline.ch

Owners of this part of documents:

	Prof. Suk-Hwan Suh	Stefan Heusinger
	NRL-SNT, POSTECH	ISW, University of Stuttgart
Address:	San 31, Hyoja-Dong, Nam-Gu	Seidenstrasse 36
	Pohang, Kyungbuk, 790-784, Korea	D-70174 Stuttgart, Germany
Telephone:	+82 (54) 279-2196	+49 (0711) 121-2342
FAX:	+82 (54) 279-5998	+49 (0711) 121-2413
E-mail:	shs@postech.ac.kr	stefan.heusinger@isw.uni-stuttgart.de

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Attention is drawn to the possibility that some of the elements of this part of ISO 14649 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

International Standard ISO 14649-12 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 1, *Physical device control*.

ISO 14649 consists of the following parts, under the general title Industrial automation systems and integration — Physical device control — Data model for computerized numerical controllers:

NOTE Phase numbers below refer to the planned release phases of ISO 14649 which are described in Annex D of ISO 14649-1:2002.

- Part 1: Overview and fundamental principles (Phase 1)
- Part 10: General Process Data (Phase 1)
- Part 11: Process Data for Milling (Phase 1)
- Part 12: Process Data for Turning (Phase 2)
- Part 13: Process Data for wire-EDM (Phase 2)
- Part 14: Process Data for sink-EDM (Phase 2)
- Part 111: Tools for Milling Machines (Phase 1)
- Part 121: Tools for Turning Machines (Phase 2)

Gaps in the numbering were left to allow further additions. ISO 14649-10 is the ISO 10303 Application Reference Model (ARM) for process-independent data. ISO 10303 ARMs for specific technologies are added after part 10.

ISO 14649 is harmonized with ISO 10303 in the common field of Product Data over the whole life cycle. Figure 1 of ISO 14649-1 shows the different fields of standardization between ISO 14649, ISO 10303 and CNC manufacturers with respect to implementation and software development.

Introduction

Modern manufacturing enterprises are built from facilities spread around the globe, which contain equipment from hundreds of different manufacturers. Immense volumes of product information must be transferred between the various facilities and machines. Today's digital communications standards have solved the problem of reliably transferring information across global networks. For mechanical parts, the description of product data has been standardized by ISO 10303. This leads to the possibility of using standard data throughout the entire process chain in the manufacturing enterprise. Impediments to realizing this principle are the data formats used at the machine level. Most computer numerical control (CNC) machines are programmed in the ISO 6983 “G and M code” language. Programs are typically generated by computer-aided manufacturing (CAM) systems that use computer-aided design (CAD) information. However, ISO 6983 limits program portability for three reasons. First, the language focuses on programming the tool center path with respect to machine axes, rather than the machining process with respect to the part. Second, the standard defines the syntax of program statements, but in most cases leaves the semantics ambiguous. Third, vendors usually supplement the language with extensions that are not covered in the limited scope of ISO 6983.

ISO 14649 is a new model of data transfer between CAD/CAM systems and CNC machines, which replaces ISO 6983. It remedies the shortcomings of ISO 6983 by specifying machining processes rather than machine tool motion, using the object-oriented concept of Workingsteps. Workingsteps correspond to high-level machining features and associated process parameters. CNCs are responsible for translating Workingsteps to axis motion and tool operation. A major benefit of ISO 14649 is its use of existing data models from ISO 10303. As ISO 14649 provides a comprehensive model of the manufacturing process, it can also be used as the basis for a bi- and multi-directional data exchange between all other information technology systems.

ISO 14649 represents an object oriented, information and context preserving approach for NC-programming, that supersedes data reduction to simple switching instructions or linear and circular movements. As it is object- and feature oriented and describes the machining operations executed on the workpiece, and not machine dependent axis motions, it will be running on different machine tools or controllers. This compatibility will spare all data adaptations by postprocessors, if the new data model is correctly implemented on the NC controllers. If old NC programs in ISO 6983 are to be used on such controllers, the corresponding interpreters shall be able to process the different NC program types in parallel.

ISO TC 184/SC 1/WG 7 envisions a gradual evolution from ISO 6983 programming to portable feature-based programming. Early adopters of ISO 14649 will certainly support data input of legacy “G and M codes” manually or through programs, just as modern controllers support both command-line interfaces and graphical user interfaces. This will likely be made easier as open-architecture controllers become more prevalent. Therefore, ISO 14649 does not include legacy program statements, which would otherwise dilute the effectiveness of the standard.

Industrial automation systems and integration — Physical device control — Data model for Computerized Numerical Controllers — Part 12: Process data for turning

1 Scope

This part ISO 14649-12 specifies the technology specific data elements needed as process data for turning. Together with the general process data described in ISO 14649-10 of this standard, it describes the interface between a computerized numerical controller and the programming system (i.e. CAM system or shop floor programming system) for turning. It can be used for turning operations on all types of machines including turning machine or lathe, or turning centers. In this version, feature and operation data models for conventional turning, involving only x and z movements, are covered. Features and operations for the composite machining including c-axis operation will be covered in the later version of this document or in a separate document. Also, the scope of this part does not include any other technologies, like milling, grinding, contour cutting, or EDM. These technologies will be described in other parts of this standard.

The subject of the turning schema, which is described in this document, is the definition of technology specific data types representing machining features and processes for turning operation on lathes. Not included in this schema are representations, executable objects, and base classes which are common for all technologies. They are referenced from ISO 10303's generic resources and ISO 14649-10 of this standard. The description of process data is done using the EXPRESS language as defined in ISO 10303-11. The encoding of the data is done using ISO 10303-21.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 14649. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 14649 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 10303-11: Industrial automation systems and integration: Product data and exchange – Description methods: the EXPRESS Language Reference Manual (IS).

ISO 10303-21: Industrial automation systems and integration: Product data and exchange - Implementation methods: Clear text encoding of the exchange structure (IS).

ISO 10303 AP 203: Application protocol: Configuration controlled design (IS).

ISO 10303 AP 214: Application protocol: Core data for automotive mechanical design process (IS).

ISO 10303 AP 224: Application protocol: Mechanical product definition for process planning using machining features (IS).

ISO 14649-1: Data model for computerized numerical controllers – Overview and fundamental principles (FDIS).

ISO 14649-10: Data model for computerized numerical controllers – General process data (FDIS).

ISO 14649-11: Data model for computerized numerical controllers – Process data for milling (FDIS).

3 Terms and definitions

For the purposes of this part of ISO 14649, the terms and definitions given in ISO 14946-10 of this standard and the following apply.

3.1 Roughing

A machining operation used to cut a part. While the aim of roughing is to remove large quantities of material in a short time, the surface quality is usually not important. The roughing operation is usually followed by a finishing operation, cf. finishing.

3.2 Finishing

A machining operation used to cut a part. The finishing operation usually follows a roughing operation. The goal of finishing is to reach the surface quality required, cf. roughing.

4 Symbols (and abbreviated terms)

No symbols defined in this part.

5 Process data for turning

5.1 Header and references

The following listing gives the header for the turning schema and the list of types and entities, which are referenced within this schema.

```
SCHEMA turning_schema;

(*
   Version : 14
   Date    : 24.10.2003
   Author  : ISO TC184/SC1/WG7
   Contact : Suk-Hwan Suh <shs@postech.ac.kr>
             Stefan Heusinger <stefan.heusinger@isw.uni-stuttgart.de>
*)


(* ************************************************************************ *)
(* Types from machining_schema                 ISO 14649-10                 *)
(* ************************************************************************ *)

REFERENCE FROM machining_schema(
   axis2_placement_3d,
   bounded_curve,
   cartesian_point,
   direction,
   general_profile,
   identifier
   in_process_geometry,
   label,
```

```
   length_measure,
   linear_profile,
   machine_functions,
   machining_operation,
   manufacturing_feature,
   material,
   open_profile,
   partial_area_definition,
   partial_circular_profile,
   plane_angle_measure,
   positive_length_measure,
   positive_ratio_measure,
   pressure_measure,
   property_parameter,
   rot_speed_measure,
   round_hole,
   speed_measure,
   taper_select,
   technology,
   thread,
   time_measure,
   toleranced_length_measure,
   two5D_manufacturing_feature,
   vee_profile,
   workingstep);

(* ************************************************************************ *)
(* Types from milling_ schema                ISO 14649-11                   *)
(* ************************************************************************ *)

REFERENCE FROM milling_schema(
   adaptive_control,
   approach_retract_strategy,
   process_model_list);
```

5.2 Manufacturing features for turning

The base class of all features used for turning is the *turning_feature*. The *turning_feature* is a subclass of the *two5D_manufacturing_feature* described in ISO 14649-10. The turning features described in this chapter are fully harmonized with ISO 10303 AP224. Features that can be obtained by turning operation, as well as milling operation, such as *round_hole* and *thread_hole*, are not described in this part of the standard; but users can use these features by referencing ISO 14649-10. Also, *toolpath_feature* which is defined in ISO 14649-10 can be used for the toolpath type features in turning.

5.2.1 Turning feature

The entity *turning_feature* is the abstract base class for all features used for turning. The defined turning features are classified geometric shapes that can be obtained by turning the cylindrical workpiece with 2-axis (x and z) operation or 3-axis (x, z, and c) operation (Figure 1). In this version, features that can be obtained by 3-axis operation are not included as stated in chapter 1.

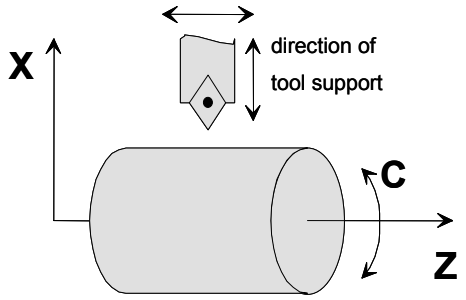


Figure 1: Axis and motion nomenclature of turning operation

Currently, the following features are included: *outer_round* (*outer_diameter*, *outer_diameter_to_shoulder*), *revolved_feature* (*revolved_flat*, *revolved_round*, *groove*, *general_revolution*), *knurl* (*straight_knurl*, *diagonal_knurl*, *diamond_knurl*, *tool_knurl*).

NOTE: Turning specific features; i.e., features which are not defined in AP224, but frequently used in common practice, such as *cut_in, circular_face* (including *end_face* and *step_face*) are described in Annex C of this document.

```
ENTITY turning_feature
   ABSTRACT SUPERTYPE OF (ONEOF(outer_round, revolved_feature, knurl))
   SUBTYPE OF (two5D_manufacturing_feature);
END_ENTITY;
```

5.2.2 Outer round

An *outer_round* is a type of *turning_feature* that is an outline or significant shape that is swept through a complete revolution about an axis. Each *outer_round* is either an *outer_diameter* or an *outer_diameter_to_shoulder*. The axis of revolution shall be the same as the z-axis of the feature.

```
ENTITY outer_round
   ABSTRACT SUPERTYPE OF (ONEOF (outer_diameter, outer_diameter_to_shoulder))
   SUBTYPE OF (turning_feature);
END_ENTITY;
```

5.2.2.1 Outer diameter

The *outer_diameter* is a subtype of *outer_round* that is a sweeping of an outline specified by a line segment one complete revolution about an axis. The line is finite in length and coplanar with the axis. The *outer_diameter* (Figure 2) may have a constant diameter around the axis of rotation (cylinder; left figure), or it may be tapered (cone; right figure). In case of the definition of a cylinder the *diameter_at_placement* and the *feature_length* are sufficient. A cone describes a continual transition from one diameter to another diameter across a certain *feature_length*. For its definition the additional attribute *reduced_size* is used. In other words, cone and cylinder, which are commonly used on the shop floor, can be respectively represented by *outer_diameter* or *outer_diameter* with taper as shown in Figure 2.

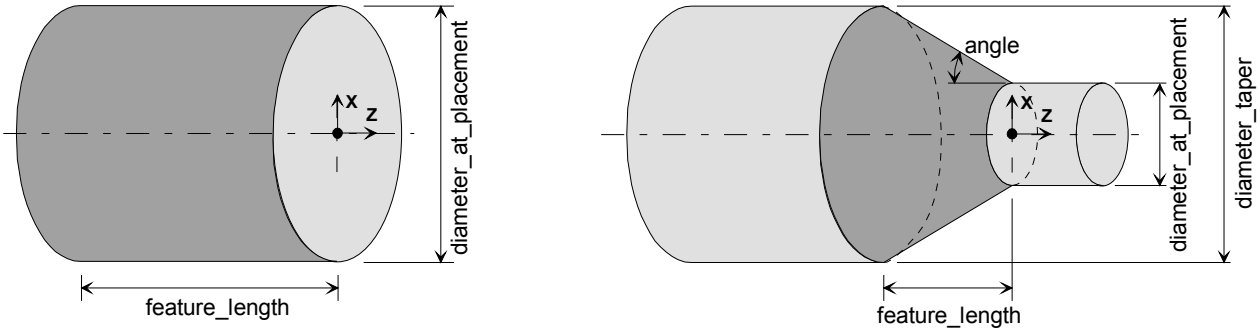


Figure 2: Outer_diameter (left) and outer_diameter with taper (right)

```
ENTITY outer_diameter
   SUBTYPE OF (outer_round);
   diameter_at_placement : toleranced_length_measure;
   feature_length        : toleranced_length_measure;
   reduced_size          : OPTIONAL taper_select;
END_ENTITY;
```

diameter_at_placement : This attribute describes the diameter at the side of the feature, where the origin of the co-ordinate system of the feature's placement is defined.

feature_length : The length of the feature. Its length is measured along its rotation axis from z = 0 of the feature's co-ordinate system to the leftmost point (negative value of z) of the feature.

reduced_size : The optional attribute *reduced_size* makes it possible to distinguish between the definition of cylinder and cone. If omitted, the feature *outer_diameter* describes a cylinder. A cone can be described by one of the methods defined in taper_select, which is defined in ISO 14649-10. For defining a cone, it can be selected between two possibilities. If a *diameter_taper* is used, this is the diameter of the opposite side of the feature's placement co-ordinate system. If an *angle_taper* is chosen for describing the cone, this angle is the angle between the negative z-axis and the line on the positive x-side of the z-axis defined by the intersection of the cone with the xz-plane of the feature, extended to meet the z-axis. An angle greater than 0 degrees and less than 90 degrees indicates a cone with increasing diameter for decreasing z-values, an angle between 0 degrees and –90 degrees indicates a cone with decreasing diameter for decreasing z-values.

5.2.2.2 Outer diameter to shoulder

An *outer_diameter_to_shoulder* is a type of *outer_round* that is a sweeping of a shape one complete revolution about an axis. The shape shall be specified by two lines that connect at a point and extend infinitely. The enclosed angle shall be smaller than 180°.

NOTE: A turning specific feature which is frequently used on the shop floor, the *step_face* can be represented by *outer_diameter_to_shoulder*. Details for *step_face* are described in Annex C.1.

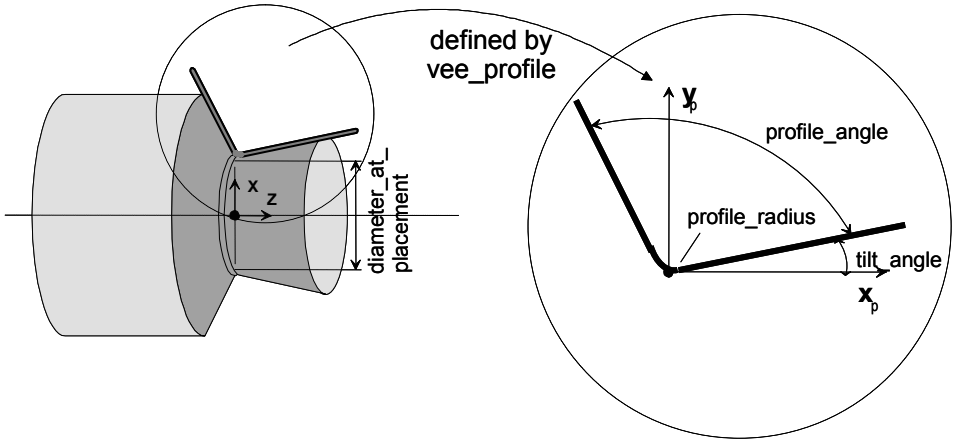


Figure 3: Outer_diameter_to_shoulder

```
ENTITY outer_diameter_to_shoulder
   SUBTYPE OF (outer_round);
   diameter_at_placement : toleranced_length_measure;
   v_shape_boundary      : vee_profile;
END_ENTITY;
```

diameter_at_placement : This attribute describes the diameter at the feature's placement (co-ordinate system). The z co-ordinate is the position where the two sides of the *vee_profile* come together.

v_shape_boundary: An outline or shape that shall be revolved about an axis. The *vee_profile* specifies the revolved shape required by an *outer_diameter_to_shoulder*. The placement of the profile shall be along the x-axis of the *outer_diameter_to_shoulder* at a specified distance away from the origin. The orientation of the y-axis of the *vee_profile* shall be the same as the x-axis of the *outer_diameter_to_shoulder* and the x-axis of the *vee_profile* shall be the same as the z-axis of the *outer_diameter_to_shoulder*.

5.2.3 Revolved feature

A *revolved_feature* is a subtype of *turning_feature* that is a sweeping of a planar profile on complete revolution about the z-axis. The planar profile shall be finite in length, coplanar with the axis of revolution, and shall not intersect the axis of revolution. The *revolved_feature* may be either an outer shape of a part or a volume removal, depending on the material side. Each *revolved_feature* is one of the following: *general_revolution*, *groove*, *revolved_flat*, or *revolved_round*.

```
ENTITY revolved_feature
   ABSTRACT SUPERTYPE OF (ONEOF (revolved_round, revolved_flat, groove,
  general_revolution))
   SUBTYPE OF (turning_feature);
   material_side : OPTIONAL direction;
   radius        : length_measure;
END_ENTITY;
```


material_side: The *material_side* specifies the material removal direction; i.e., the direction towards which the material moves as it is removed from the part. The *material_side* direction is defined in the feature co-ordinate system.

radius: The distance from the axis of rotation to define placement of the profile that will be swept about the axis. The value of this *length_measure* shall be greater or equal to 0.

5.2.3.1 Revolved flat

The *revolved_flat* is a type of *revolved_feature* that is the sweeping of a straight line about an axis.

NOTE: A turning specific feature which is frequently used on the shop floor, the *circular_face* can be represented by *revolved_flat*. Details for this feature are described in Annex C.1.

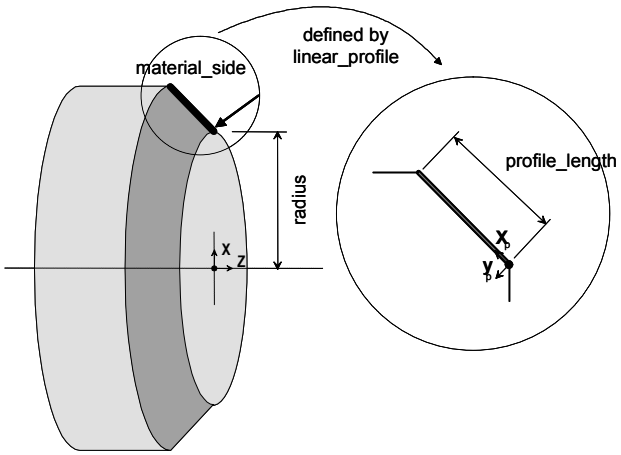


Figure 4: Revolved flat

```
ENTITY revolved_flat
   SUBTYPE OF (revolved_feature);
   flat_edge_shape : linear_profile;
END_ENTITY;
```

flat_edge_shape: A linear profile that when revolved about an axis defines the shape of an area of the part. The placement of the profile shall be along the x-axis of the *revolved_flat* at a distance specified by the inherited attribute *radius* away from the origin. The z-axis orientation of the *linear_profile* shall be the same as the y-axis of the *revolved_flat*, the x-axis and z-axis are independent of the orientation of the *revolved_flat* feature.

5.2.3.2 Revolved round

The *revolved_round* is a subtype of *revolved_feature* that is the sweeping of an arc about an axis.

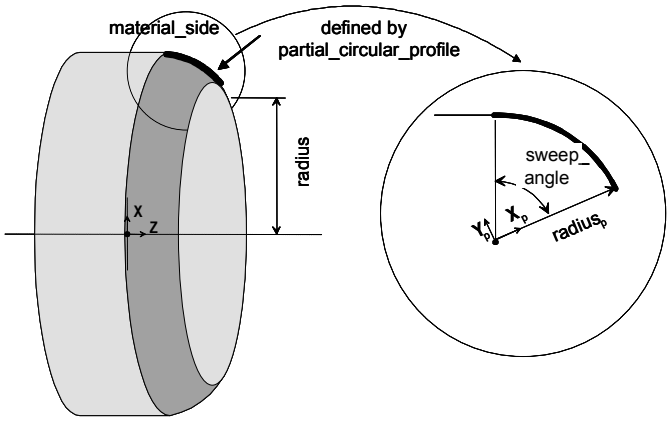


Figure 5: Revolved round

```
ENTITY revolved_round
   SUBTYPE OF (revolved_feature);
   rounded_edge_shape : partial_circular_profile;
END_ENTITY;
```

rounded_edge_shape: This attribute specifies the arc that when revolved about an axis defines the shape of an area of the part. The placement of the profile shall be along the x-axis of the *revolved_round* at a distance specified by the inherited attribute *radius* away from the origin. The z-axis orientation of the *partial_circular_profile* shall be the same as the y-axis of the *revolved_round*, the x-axis and y-axis are independent of the orientation of the *revolved_round* feature.

5.2.3.3 Groove

The *groove* is a type of *revolved_feature* that is a narrow channel or depression that is swept through one complete revolution about an axis. The face shape that has the groove applied to it is determined by the profile orientation as shown in Figure 6.

NOTE: A commonly used feature in shop floor, such as *cut_in*, the shape that is obtained by applying a cutting tool, can be represented by the *groove* feature. Refer to Annex C.2 for the details. Standardized undercuts for shoulders (represented by the turning feature *outer_diameter_to_shoulder*) and threads (represented by *thread*) may be represented by restrictions on a *square_u_profile* defining the *groove*'s shape.

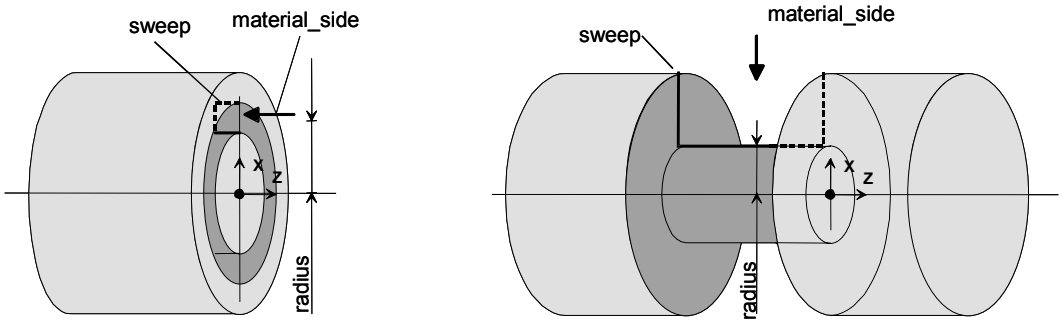


Figure 6: Groove

```
ENTITY groove
   SUBTYPE OF (revolved_feature);
   sweep : open_profile;
END_ENTITY;
```

sweep: An outline or shape that shall be revolved about an axis. The *open_profile* specifies the sweep shape required by a groove. The groove bottom condition can be described by using the subtypes of *open_profile* defined in ISO 14649-10, such as *square_u_profile*, *rounded_u_profile*, *partial_circular_profile*, and *general_profile*. The placement of the profile shall be along the x-axis of the *groove* at a distance specified by the inherited attribute *radius* away from the origin. The z-axis orientation of the *open_profile* shall be the same as the y-axis orientation of the *groove* feature. The x- and y-axis orientations of the *open_profile* are independent of the orientation of the *groove* feature. The *groove* feature may be defined on different faces of a part depending on the orientation of the profile.

5.2.3.4 General revolution

The *general_revolution* is a subtype of *revolved_feature* that is an arbitrary planar shape swept one complete revolution about a z-axis. The arbitrary planar shape shall be finite in length, coplanar with the axis of revolution, and shall not intersect the axis of revolution. The *general_revolution* may be either an outer shape of a part or a volume removal, depending on the material direction.

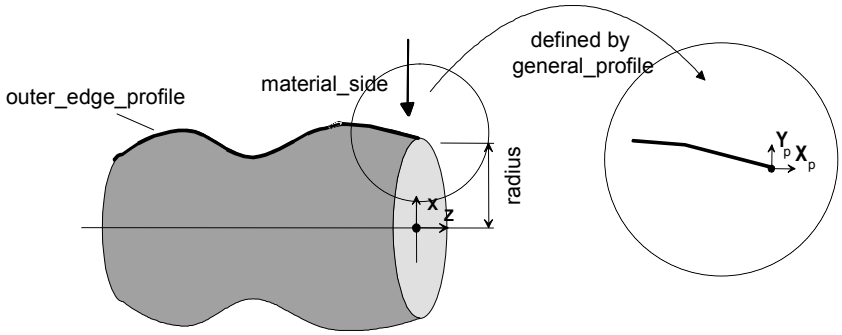


Figure 7: General revolution

```
ENTITY general_revolution
   SUBTYPE OF (revolved_feature);
   outer_edge_profile : general_profile;
END_ENTITY;
```

outer_edge_profile: This attribute describes the shape of a *general_revolution* using a *general_profile*. The placement of the profile shall be along the x-axis of the *general_revolution* at a specified distance away from the origin. This distance is expressed with the attribute *radius* inherited from the base class.

5.2.4 Knurl

A *knurl* is a type of *turning_feature* which is a scoring pattern consisting of a series of shallow cuts on a cylindrical surface. In general special tools are used for applying knurls. Each *knurl* is assigned on the cylindrical surface of the base feature. The entity *knurl* is the abstract base class for the following kind of knurls: *straight_knurl*, *diagonal_knurl*, *diamond_knurl,* and *tool_knurl*. The knurl's placement (shown in the figures as co-ordinate system with index "k"), which is inherited from the entity *turning_feature*, is relative to the placement of the base feature.

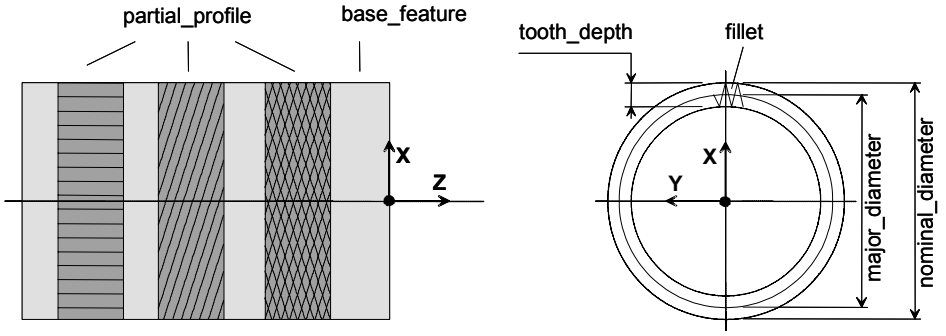


Figure 8: Knurl with basic parameters

```
ENTITY knurl
   ABSTRACT SUPERTYPE OF (ONEOF(straight_knurl, diagonal_knurl, diamond_knurl,
  tool_knurl))
   SUBTYPE OF (turning_feature);
   base_feature     : turning_feature;
   partial_profile  : OPTIONAL partial_area_definition;
   tooth_depth      : OPTIONAL toleranced_length_measure;
   diametral_pitch  : OPTIONAL toleranced_length_measure;
   root_fillet      : OPTIONAL toleranced_length_measure;
   number_of_teeth  : OPTIONAL INTEGER;
   major_diameter   : OPTIONAL toleranced_length_measure;
   nominal_diameter : OPTIONAL toleranced_length_measure;
END_ENTITY;
```

base_feature : A reference to a feature on which the *knurl* is applied.

partial_profile : This optional attribute can be used for limiting the area, the *knurl* is applied on. If omitted, the length of the *base_feature* is used.

tooth_depth : The depth from the crest of a tooth to the point where two teeth intersect.

diametral_pitch : The ratio of the *nominal_diameter* to the number of teeth in the circumference.

root_fillet : The attribute *root_fillet* specifies the dimension of a radius between teeth on a knurling tool.

number_of_teeth : The number of teeth in the circumference produced on the part surface. The *number_of_teeth* need not be specified for a particular *knurl*. If *nominal_diameter* and *diametral_pitch* are also given, this attribute has to hold the quotient of *norminal_diameter* to *diametral_pitch*.

major_diameter : The size of the part before a *knurl* is applied to it.

nominal_diameter : The size of the part after a *knurl* has been applied.

5.2.4.1 Straight knurl

A *straight_knurl* is a type of *knurl* in which the knurl scoring is parallel to the axis of the scored surface (z-axis). In Figure 9, x, z and x_k, z_k indicate the placement of the *base_feature* and *knurl*, respectively.

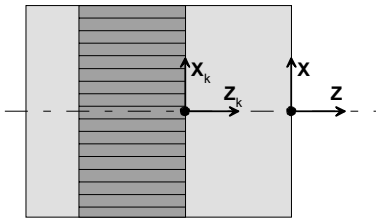


Figure 9: Straight knurl

```
ENTITY straight_knurl
   SUBTYPE OF (knurl);
END_ENTITY;
```

5.2.4.2 Diagonal knurl

A *diagonal_knurl* is a type of *knurl* which makes it possible to define helical cuts at an angle about the axis of a surface.

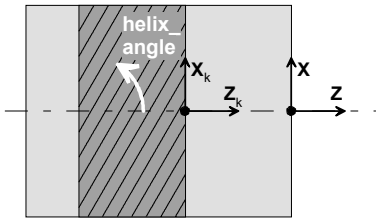


Figure 10: Diagonal knurl

```
ENTITY diagonal_knurl
   SUBTYPE OF (knurl);
   helix_angle : plane_angle_measure;
END_ENTITY;
```

helix_angle:	The *helix_angle* specifies the angle the knurl pattern makes with the orientation axis of an applied to surface.

5.2.4.3 Diamond knurl

A *diamond_knurl* is a special knurl with two helical cuts on its surface. The angles of the two helixes are basically independent from each other.

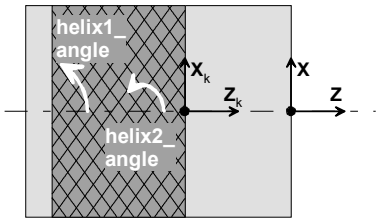


Figure 11: Diamond knurl

```
ENTITY diamond_knurl
   SUBTYPE OF (knurl);
   helix1_angle : plane_angle_measure;
   helix2_angle : OPTIONAL plane_angle_measure;
END_ENTITY;
```

helix1_angle: This attribute specifies the angle of the first scratch against the z-axis.

helix2_angle: The angle of the second helical cut. If helix2_angle is not given, the angle of the second scratch shall be calculated by the controller as: 180° - *helix1_angle*.

5.2.4.4 Tool knurl

A *tool_knurl* is the general knurl pattern that cannot be represented by any of above. A *tool_knurl* is only done with special tools, the *user_defined_turning_tool* for knurls defined in ISO 14649-121.

```
ENTITY tool_knurl
   SUBTYPE OF (knurl);
END_ENTITY;
```

5.3 Machining workingstep for turning

5.3.1 Turning workingstep

A single turning feature may be made using a *machining_workingstep* from ISO 14649-10. On a turning workpiece, however, it is often the case that several connected turning features can be made simultaneously by a single turning operation, and, for efficiency, should be made that way. A *machining_workingstep* is not able to perform an operation over a list of features. Therefore, the *turning_workingstep* has been defined that allows a single operation to be applied to a list of connected features. The connected features have to be ordered by decreasing z-values of the feature placement. The inherited attribute *its_secplane* refers to the first feature of the feature list. The operation might be a *contouring* operation using the *contour_turning* strategy defined in section 5.4.

```
ENTITY turning_workingstep
   SUBTYPE OF (workingstep);
   its_features  : LIST [2:?] OF manufacturing_feature;
   its_operation : turning_machining_operation;
   its_effect    : OPTIONAL in_process_geometry;
END_ENTITY;
```

its_features : This attribute describes a list of at least two *manufacturing_feature*s upon which the *turning_workingstep* operates. It is used to connect several features to perform a contouring operation.

its_operation : The operation which will be performed upon the list of *manufacturing_feature*s. The operation must be a member of the *manufacturing_feature*'s list of operations.

its_effect : The change to the geometry of the workpiece effected by the operation. A CAM system can use this attribute to describe the predicted effect of this operation on the geometry of the workpiece. If given, the controller can compare the geometry change described by this attribute with the geometry change predicted by its internal algorithm. ISO 14649 does not describe how closely the two geometries must match in order for the controller to be considered to be in conformance.

5.4 Machining operations for turning

In this section all machining operations and technology specific data are introduced which will be needed for turning.

5.4.1 Turning technology

This entity defines the technological parameters of the turning operation. It is a subtype of entity *technology* defined in ISO 14649-10 of this standard. The feed rate for the x- and z-axis is defined through the *feedrate* property or alternatively by the *feed_per_revolution*, which is inherited from the entity's upper class. If its_adaptive_control is invoked, some or all of the values from *turning_technology* may be ignored.

```
ENTITY turning_technology
   SUBTYPE OF (technology);
   spindle_speed             : speed_select;
   feed_per_revolution       : OPTIONAL  feed_select;
   sync_spindle_and_z_feed   : BOOLEAN;
   inhibit_feedrate_override : BOOLEAN;
   inhibit_spindle_override  : BOOLEAN;
   its_adaptive_control      : OPTIONAL adaptive_control;
END_ENTITY;
```

spindle_speed : The attribute spindle_speed makes it possible to select between constant spindle speed and constant speed at the surface.

feed_per_revolution : This optional attribute makes it possible to define the feed rate alternatively to the inherited *feedrate* from the entity *technology* in millimetre per revolution.

sync_spindle_and_z_feed : If true, feed rate in z and spindle speed shall be synchronised. Therefore, the pitch of tap can be kept constant at the beginning or the end of a thread when feed rate is being accelerated or decelerated. If false, feed rate in z and spindle speed are controlled independently.

inhibit_feedrate_override : If true, the feed rate override through the operating panel or by adaptive control systems shall not be allowed. If false, the feed rate override through the operating panel or by adaptive control systems shall be allowed.

inhibit_spindle_override : If true, the spindle speed override through the operating panel or by adaptive control systems shall not be allowed. If false, the spindle speed override through the operating panel or by adaptive control systems shall be allowed.

its_adaptive_control : Any kind of vendor specific adaptive control strategy.

5.4.1.1 Speed select

This type is used for selecting the mode of speed control for the lathe spindle. In general two possibilities have to be considered. In the first case a constant spindle speed is given; this speed is measured in revolutions per second. The second case is used for keeping the speed at the surface of the workpiece constant; the unit of this speed is meter per second.

```
TYPE speed_select = SELECT (const_spindle_speed, const_cutting_speed);
END_TYPE;
```

5.4.1.2 Const spindle speed

With this entity, a constant rotational speed of the spindle can be given. The speed shall be kept constant at each position of the diameter.

```
ENTITY const_spindle_speed;
   rot_speed : rot_speed_measure;
END_ENTITY;
```

rot_speed : Constant speed of the spindle. As defined for the property *rot_speed_measure*, positive values indicate counterclockwise rotation as viewed from the positive z-axis.

5.4.1.3 Const cutting speed

If this entity for defining the speed is chosen, the spindle speed shall be adapted in the way that the speed at the surface is constant. This means that the smaller the diameter of the machined surface is the higher is the spindle speed and vice versa.

```
ENTITY const_cutting_speed;
   speed     : speed_measure;
   max_speed : OPTIONAL rot_speed_measure;
END_ENTITY;
```

speed : A constant speed at the surface of the workpiece. The speed means the tangential velocity at the tool tip while removing material. Positive values indicate counterclockwise rotation as viewed from the positive z-axis.

max_speed : With this optional attribute the maximal rotational speed can be limited. While decreasing the diameter of the machined surface, the spindle speed has to be raised.

For very small radii the spindle speed becomes very high and is limited by the capabilities of the lathe. If *max_speed* is given, it should always be positive as the rotational speed will be limited by the absolute value of max_speed.

5.4.1.4 Feed select

This type is used to define the feed measure as velocity or as propagated distance per revolution.

```
TYPE feed_select = SELECT (feed_velocity_type, feed_per_rev_type);
END_TYPE;
```

5.4.1.5 Feed velocity type

This type is used to specify the feed in meters per second.

```
TYPE feed_velocity_type = speed_measure;
END_TYPE;
```

5.4.1.6 Feed per rev type

This type is used to specify the feed in millimeter per revolution.

```
TYPE feed_per_rev_type = REAL;
END_TYPE;
```

5.4.2 Turning machine functions

This entity describes the state of various functions of the machine, like coolant, chip removal, etc. to be applied during the time span of an operation. It is a subtype of entity *machine_functions* defined in ISO 14649-10. Note that some attributes of *turning_machine_functions* are the same as those of *milling_machine_functions* defined in ISO 14649-11.

```
ENTITY turning_machine_functions
   SUBTYPE OF (machine_functions);
   coolant               : BOOLEAN;
   coolant_type          : OPTIONAL coolant_select;
   coolant_pressure      : OPTIONAL pressure_measure;
   axis_clamping         : LIST [0:?] OF identifier;
   chip_removal          : OPTIONAL BOOLEAN;
   oriented_spindle_stop : OPTIONAL direction;
   its_process_model     : OPTIONAL process_model_list;
   other_functions       : SET [0:?] OF property_parameter;
   tail_stock            : OPTIONAL BOOLEAN;
   steady_rest           : OPTIONAL BOOLEAN;
   follow_rest           : OPTIONAL BOOLEAN;
```

```
END_ENTITY;
```

coolant : If true, the coolant shall be activated. If false, the coolant shall not be activated.

coolant_type : Optional specification of the type of the coolant. Only valid if coolant is true.

coolant_pressure : Optional specification of the pressure of the coolant system. Only valid if coolant is true.

axis_clamping : Describes which axes are to be clamped, e.g. x, z, c. Note that this information is machine dependent and should be avoided.

chip_removal : This attribute is used to define if the machine has additional capabilities for breaking or removing chips. If the attribute is set to true, the chip removal system shall be activated. If it is false, the chip removal system shall not be activated.

oriented_spindle_stop : If specified, the spindle will stop in the given direction relative to the machine zero position of the c-axis in case a spindle stop occurs during or at the end of the workingstep. This option is especially used for complete machining, i.e. if a milling operation is applied.

its_process_model : Optional information for process control.

other_functions : Optional list of other functions of generic type.

tail_stock : If true, the tail stock of the lathe shall be used to provide a fixture at the end of the part opposite from the chuck. If false, the tail stock of the lathe shall not be used.

steady_rest : If true, the steady rest of the lathe shall be used to make long and delicate spindle turnings. If false, the follow rest of the lathe shall not be used.

follow_rest : If true, the follow rest of the lathe shall be used to keep slender workpiece from flexing and climbing the tool. If false, the follow rest of the lathe shall not be used.

5.4.2.1 Coolant select

This is to describe the type of coolant. It is one of three types; *flood, mist* and *through_tool.*

```
TYPE coolant_select = ENUMERATION OF (flood, mist, through_tool);
END_TYPE;
```

5.4.3 Turning machining strategy

This entity is used as abstract supertype for the description of the strategy used for creating a turning toolpath. It is a subtype of entity *machining_strategy* defined in ISO 14649-10. All directions defined in subtypes are related to the workpiece co-ordinate system.

```
ENTITY turning_machining_strategy
   ABSTRACT SUPERTYPE OF (ONEOF (unidirectional_turning, bidirectional_turning,
  thread_strategy, contour_turning, grooving_strategy,
  explicit_turning_strategy));
```



```
    overcut_length        : OPTIONAL length_measure;
    allow_multiple_passes : OPTIONAL BOOLEAN;
    cutting_depth         : LIST[0:?] OF length_measure;
    variable_feedrate     : OPTIONAL positive_ratio_measure;
END_ENTITY;
```

overcut_length : The over travel length on the open side(s) of the feature. It is not allowed for manufacturing of features which are bounded by material on all sides, e.g. grooves.

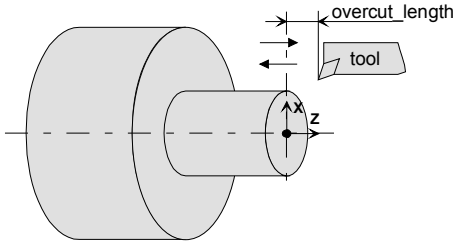


Figure 12: Overcut length of the tool

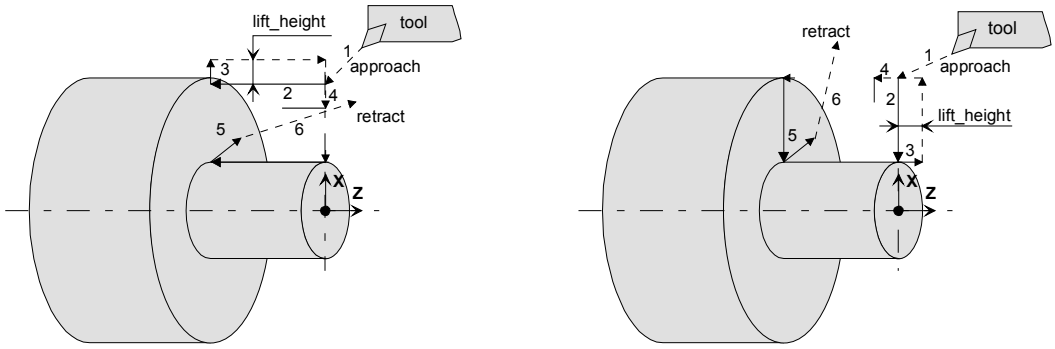
allow_multiple_passes : Optional flag only for roughing workingsteps. If true, this is the standard roughing operation with multiple passes, i.e. several layers of material are removed sequentially, taking into account the list of cutting depth. If false, this is the special roughing operation for pre-cast features with one pass. Default is true.

cutting_depth : If the material to be removed is too thick to be removed in a single pass, it must be removed in several passes. Each entry of the optional *cutting_depth* list stands for the thickness (not the total depth) that shall be removed in the respective pass (until all material to be removed is removed). The number of elements in the list is not required to be equal to the number of passes. If the cutting depth list is empty, the number of passes and the depth of each pass shall be implementation dependent. If there are more passes than list entries, the last entry shall be used for the remaining passes. If there are more list entries than passes, the surplus entries shall be ignored. If *allow_multiple_passes* is false and a *cutting_depth* list is given, only the first entry shall be used.

variable_feedrate : The changed feed rate for the cutting of subsequent layer with respect to the current feed rate. For instance, 0.8 means that the feed rate for the second and third layer cut are respectively 0.8 ∗ f, and 0.8 ∗ 0.8 ∗ f, where f is the feed rate given as *feedrate* or *feed_per_revolution* of *turning_technology*.

5.4.3.1 Unidirectional turning

Unidirectional_turning means turning in a linear fashion, i.e. going from one side to the other, then lifting the tool and going back to the starting point of feed movement. The starting point has always to be outside of the contour. As shown in Figure 13 solid bold lines mean removal of material and the dashed lines are rapid movements. After the approach of the tool (1), the cutting begins in the *feed_direction* (2). By reaching the end position the tool shall be lifted up to a desired height (3); the back path is executed in rapid feed. The starting point for the next cutting is reached by moving the tool in a *stepover_feed* usually perpendicular to the *feed_direction* (4). The length of the component of this movement perpendicular to the *feed_direction* is the lift height plus the cutting depth (inherited from the upper class). After the last cut the tool shall be moved corresponding to the *retract* strategy (5) and then the retract rapid movement will be done (6). Especially for roughing operations, the tool might follow uneven feature outlines whilst feed movement before moving in *back path direction*.



(a) feed direction: –z (b) feed direction: -x

Figure 13: Unidirectional turning

```
ENTITY unidirectional_turning
   SUBTYPE OF (turning_machining_strategy);
   feed_direction      : OPTIONAL direction;
   back_path_direction : OPTIONAL direction;
   lift_direction      : OPTIONAL direction;
   stepover_direction  : OPTIONAL direction;
   lift_height         : OPTIONAL length_measure;
   lift_feed           : OPTIONAL feed_select;
   stepover_feed       : OPTIONAL feed_select;
END_ENTITY;
```

feed_direction : This optional attribute describes the main direction of tool movement for removing material. If this value is not specified, the *feed_direction* shall be parallel to the z-axis and points to smaller values of z.

back_path_direction : The direction of the tool movement to go back where the infeed movement for the next feed movement starts.

lift_direction : The direction of the tool movement from *feed_direction* to the *back_path_direction*. If omitted, the *lift_direction* shall be perpendicular to the *feed_direction*.

stepover_direction : The direction of the tool movement from *back_path_direction* to the *feed_direction*. If omitted, the *stepover_direction* shall be perpendicular to the *back_path_direction*.

lift_height : This property specifies the length of the lift movement. If not given, the stepover movement shall occur to the *retract_plane* of the corresponding operation.

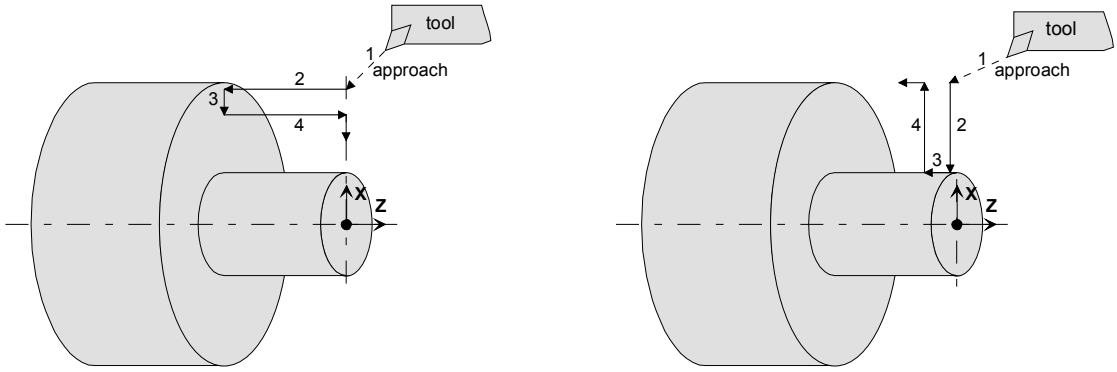
lift_feed : Feed rate for a lift movement. If omitted, the feed of *turning_technology* is used.

stepover_feed : This optional attribute makes it possible to define a feed rate for the *stepover_direction*.

5.4.3.2 Bidirectional turning

Turning in both directions of an axis as shown in Figure 14. *Bidirectional_turning* means that cutting is started from one side, e.g. after the approach (1). On reaching the other side via the toolpath (2) the cutting tool moves

downwards (3) and then back to the first side, while also removing material (4). Note that special tools are needed for bi-directional turning (neutral tools). These tools must be able to cut their way in the stepover direction (3) and backward direction (4) too. For example a tool, used for grooving, which blade is along the front side is not able to cut sideways. For *bidirectional_turning*, direction (4) shall be opposite to direction (2) (turned 180 degrees).



(a) feed direction: –z (b) feed direction: -x

Figure 14: Bidirectional turning

```
ENTITY bidirectional_turning
   SUBTYPE OF (turning_machining_strategy);
   feed_direction     : OPTIONAL direction;
   stepover_direction : OPTIONAL direction;
   stepover_feed      : OPTIONAL feed_select;
END_ENTITY;
```

feed_direction : The direction of the first tool path for the cutting operation. If this value is not specified, it shall be parallel to the z-axis and points to smaller values of z.

stepover_direction : The direction of the tool movement from one tool path to the next tool path. If omitted, it shall be perpendicular to the *feed_direction*.

stepover_feed : This optional attribute specifies the feed rate of *stepover_direction* if different to the feed specified by *turning_technology*.

5.4.3.3 Contour turning

This entity is a subtype of *turning_machining_strategy*. This strategy is to cut along feature's outline or profiles (Figure 15). Mainly, contour turning is used for roughing of pre-casted workpieces and finish cutting of features. The contour turning can be executed over a list of features (see *turning_workinstep* detailed in section 5.3.1).

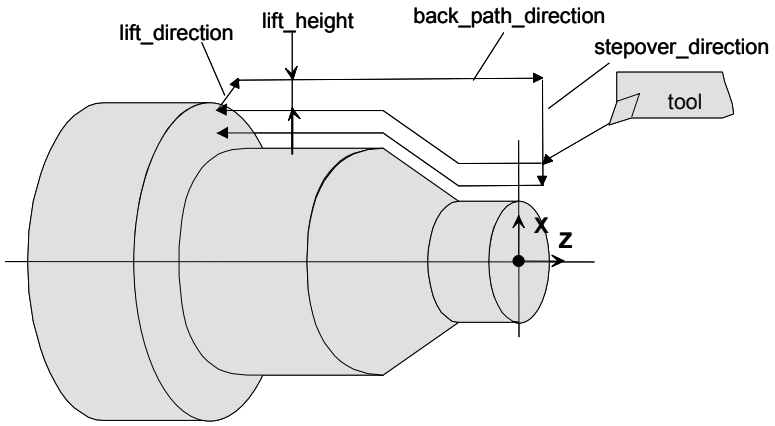


Figure 15: Contour turning

```
ENTITY contour_turning
   SUBTYPE OF (turning_machining_strategy);
   back_path_direction    : OPTIONAL direction;
   lift_direction         : OPTIONAL direction;
   stepover_direction     : OPTIONAL direction;
   lift_height            : OPTIONAL length_measure;
   lift_feed              : OPTIONAL feed_select;
   stepover_feed          : OPTIONAL feed_select;
   variable_stepover_feed : OPTIONAL positive_ratio_measure;
END_ENTITY;
```

back_path_direction : The direction of the tool movement to go back where the next feed movement starts. This attribute implicitly defines the direction of contouring.

lift_direction : The direction of the tool movement from *feed_direction* to the *back_path_direction*. If omitted, the *lift_direction* shall be in direction of the surface normal of the feature end point where the lift motion is initiated.

stepover_direction : The direction of the tool movement from *back_path_direction* to *feed_direction*. If omitted, the *stepover_direction* shall be perpendicular to the *back_path_direction*.

lift_height : This property specifies the length of the stepover movement. If not given, the stepover movement shall occur to the *retract_plane* of the corresponding operation.

lift_feed : Feed rate for a lift movement. Default is the feed of *turning_technology*.

stepover_feed : This optional attribute makes it possible to define a feed rate for the *stepover_direction*.

variable_stepover_feed : The changed stepover feed of subsequent layer with respect to the current *stepover_feed*. For instance, 0.8 means that the stepover feed for the second and third layer cut are respectively $0.8 * f$, and $0.8 * 0.8 * f$, where *f* is the *stepover_feed* given as an attribute of *contour_turning*.

5.4.3.4 Thread strategy

The *thread_strategy* is a subtype of *turning_machining_strategy* for cutting a thread.

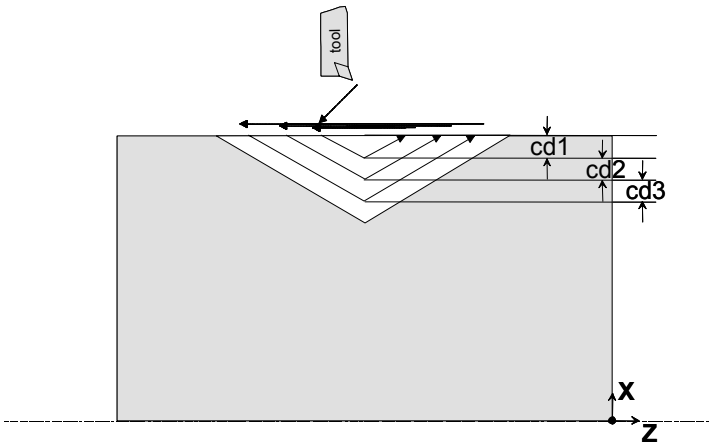


Figure 16: Thread strategy

```
ENTITY thread_strategy
   SUBTYPE OF (turning_machining_strategy);
   cut_in_amount_function : thread_cut_depth_type;
   threading_direction    : threading_direction_type;
   path_return_angle      : OPTIONAL plane_angle_measure;
   lift_height            : OPTIONAL positive_length_measure;
END_ENTITY;
```

cut_in_amount_function : The attribute specifies the amount of cut-in depth.

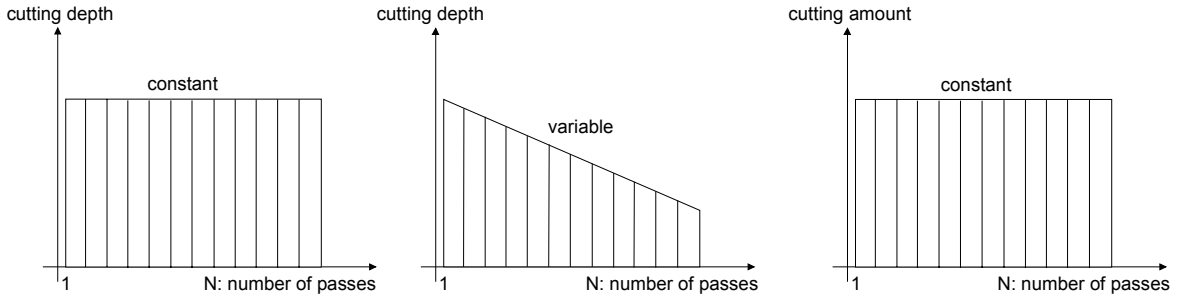
threading_direction : The attribute specifies the pass-pattern of machining a thread.

path_return_angle : The path_return_angle specifies the angle for the lift movement measured against the positive z-axis of the thread.

lift_height : This optional attribute specifies the height of the back path movement over the *major_diameter* of the *thread*.

5.4.3.4.1 Thread cut depth type

This entity describes the cutting depth for each pass. It is one of three types; *constant_depth*, *variable_depth*, and *constant_removal_amount* (Figure 17). If the *constant_depth* is used, the first entry of the *cutting_depth* list will be used. If the *variable_depth* is used, the list of *cutting_depth* is used. If there are more passes than entries, the last entry will be used for remained pass. If there are more entries than passes, other cutting depths are ignored. *constant_removal_amount* is to keep the same material removal volume for each of cut. If *constant_removal_amount* is selected, the actual cutting depth is determined by the CNC. In this case the first entry in the list for *cutting_depth* is applied for the first cut. If the list for *cutting_depth* is empty, the chosen depths are implementation dependent.



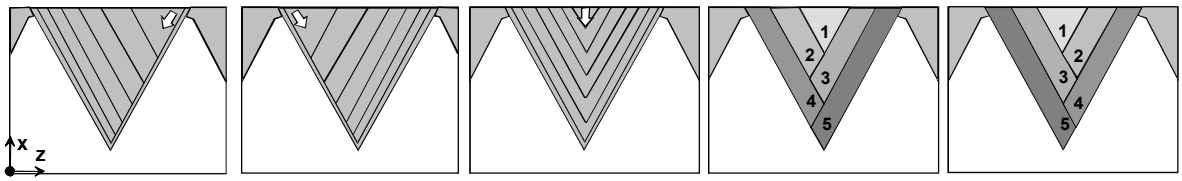
a) constant_depth b) variable_depth c) constant_removal_amount

Figure 17: Thread cut depth type for threading

```
TYPE thread_cut_depth_type = ENUMERATION OF (constant_depth, variable_depth,
  constant_removal_amount);
END_TYPE;
```

5.4.3.4.2 Threading direction type

This entity describes the pass pattern of machining a thread. There are five directions to be distinguished.



(a) left (b) right (c) center (d) left_zigzag (e) right_zigzag

Figure 18: Threading direction type

```
TYPE threading_direction_type = ENUMERATION OF (left, right, center, left_zigzag,
  right_zigzag);
END_TYPE;
```

5.4.3.5 Grooving strategy

The *grooving_strategy* is a subtype of *turning_machining_strategy*. It is used for cutting any kind of grooves which can not be machined with contouring operations.

```
ENTITY grooving_strategy
   SUPERTYPE OF (multistep_grooving_strategy)
   SUBTYPE OF (turning_machining_strategy);
   grooving_direction : OPTIONAL direction;
   travel_distance    : OPTIONAL length_measure;
END_ENTITY;
```

grooving_direction : This optional attribute is used to specify the direction of the plunge movement. If omitted, the *material_side* of the associated groove shall be applied.

travel_distance : The optional attribute *travel_distance* specifies the length of the tool movement to the next path on the operation's *retract_plane*. If omitted, the actual travel distance may be chosen depending on the tool. Refer also to Figure 19.

5.4.3.5.1 Multistep grooving strategy

The *multistep_grooving_strategy* is a subtype of *grooving_strategy*. This strategy is used where the depth of the groove is too deep, so it is impossible to cut the groove with a single cut. The depth of each step is given by *cutting_depth* defined in section 5.4.3.

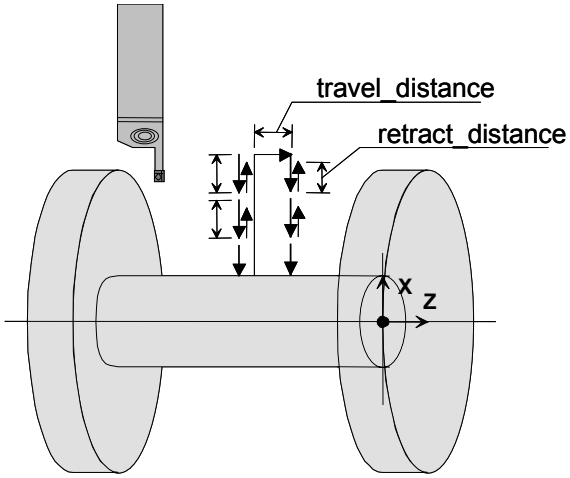


Figure 19: Grooving strategy, including multistep grooving strategy

```
ENTITY  multistep_grooving_strategy
   SUBTYPE OF (grooving_strategy);
   retract_distance : length_measure;
END_ENTITY;
```

retract_distance : If r*etract_distance* is a positive value, the tool shall be retracted this value between steps to enable chip breaking in opposite direction of *grooving_direction*. If *retract_distance* is zero, tool motion shall stop briefly (long enough for the spindle to turn more than a quarter but less than a half revolution) between steps to break the chip. If *retract_distance* is negative, the tool shall be retracted to the retract plane between steps to clear the groove of chips.

5.4.3.6 Explicit turning strategy

Any turning strategy which cannot be described using any of the above given strategies can be specified using *explicit_turning_strategy*. In this case, an exact definition of all movements needs to be given in the attribute *its_toolpath* of the entity *workingstep*.

```
ENTITY explicit_turning_strategy
   SUBTYPE OF (turning_machining_strategy);
END_ENTITY;
```

5.4.4 Turning machining operation

This is the base class of all operations for turning. It includes all necessary attributes to describe technology and cutting strategy. It is a subtype of entity *machining_operation* defined in ISO 14649-10. If the *turning_machining_operation* is referenced by a *turning_workingstep*, the inherited attribute *start_point* refers to the first feature in the list of the associated *turning_feature*s.

In general, there are two types of machining operations: roughing and finishing. The roughing is to remove all material from the original raw piece surface down to the bottom or side of the feature minus the finishing allowance in multiple passes. The finishing will then remove the finishing allowance to yield the final form of the feature. *Turning_machining_operation* is one of the following: *facing*, *grooving*, *contouring*, *threading*, or *knurling*.

NOTE: Milling and drilling type operations such as milling, drilling, boring, center drilling, reaming (which can be done by turning operation as well as milling operation) are not described in this document. Although they are not defined as the subtypes of *turning_machining_operation*, the user can use them by referencing the respective operations defined in ISO 14649-11.

```
ENTITY turning_machining_operation
   ABSTRACT SUPERTYPE OF (ONEOF(facing, grooving, contouring, threading,
  knurling))
   SUBTYPE OF (machining_operation);
   approach               : OPTIONAL approach_retract_strategy;
   retract                : OPTIONAL approach_retract_strategy;
   its_machining_strategy : OPTIONAL turning_machining_strategy;
END_ENTITY;
```

approach : Optional information about approach strategy to reach the first cut. If multiple layers have to be cut, as specified by the attribute *allow_multiple_passes*, this strategy shall also be used to move from the back path movement to the start point of the next layer. Note that *approach_retract_strategy* is referenced from ISO 14649-11.

By default, the NC controller determines the approach strategy. It may decide not to use any approach movement at all if the start point of cutting coincides with the end point of cutting for the preceding operation. If *its_toolpath* is given, this attribute shall be ignored.

retract : Optional information about retract strategy after finishing the last cut. By default, the NC controller determines the retract strategy. It may decide not to use any retract movement at all if the end point of cutting coincides with the start point of cutting for the next operation. If *its_toolpath* is given, this attribute shall be ignored. Note that *approach_retract_strategy* is referenced from ISO 14649-11.

its_machining_strategy : Description of the strategy to be used while executing the turning operation

5.4.4.1 Facing

This entity describes a turning operation for face machining, which is used to describe machining of a plane surface on the end wall of a workpiece. The position of the feature is given through the attribute *feature_placement*. A plane face can also be obtained by cutting off the end of the workpiece (cutting off is dealt in a corresponding section 5.4.4.2).

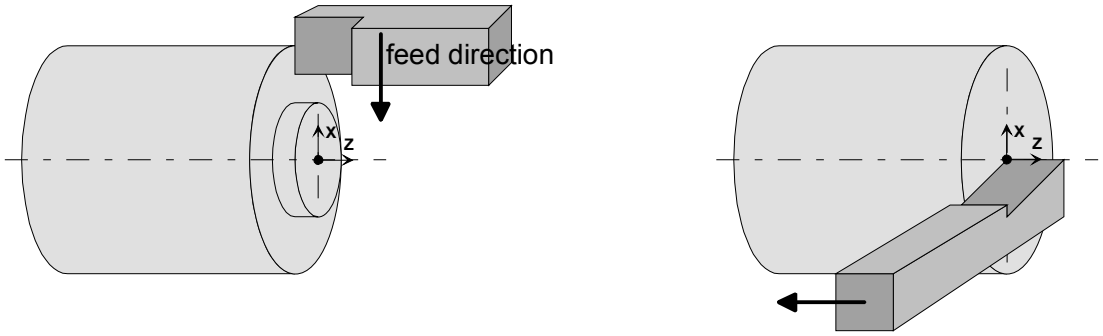


Figure 20: Facing (radial and axial tool movement)

```
ENTITY facing
   ABSTRACT SUPERTYPE OF (ONEOF(facing_rough, facing_finish))
   SUBTYPE OF (turning_machining_operation);
   allowance : OPTIONAL length_measure;
END_ENTITY;
```

allowance : The *allowance* is a depth of material which shall be left on top of the surface defined by the associated manufacturing feature (in the direction of the surface normal). The remaining material will be removed by the finishing operation.

5.4.4.1.1 Facing rough

This turning operation is used for roughing operation for facing. This operation makes it possible to remove all material until the allowance measured from the feature's position is reached, disregarded special surface requirements.

```
ENTITY facing_rough
   SUBTYPE OF (facing);
WHERE
   WR1: EXISTS(SELF.allowance) AND (SELF.allowance >= 0.0);
END_ENTITY;
```

5.4.4.1.2 Facing finish

This turning operation is used as finishing operation for facing. It makes it possible to remove all material until the feature's position is reached, applying an appropriate strategy to maintain the given tolerances. If *allowance* is given, other special operations like grinding shall be applied for removing the material left.

```
ENTITY facing_finish
   SUBTYPE OF (facing);
END_ENTITY;
```

5.4.4.2 Grooving

This entity describes a turning operation for machining a *groove*. *Grooving* is a plunge cut operation mainly for making a bottom surface. For machining the side or wall of the groove, other operations (such as *facing* and *contouring*) may be necessary.

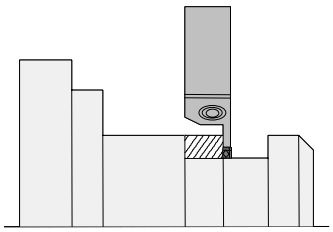


Figure 21: Grooving

```
ENTITY grooving
   ABSTRACT SUPERTYPE OF (ONEOF(grooving_rough, grooving_finish, cutting_in))
   SUBTYPE OF (turning_machining_operation);
   dwell     : OPTIONAL dwell_select;
   allowance : OPTIONAL length_measure;
END_ENTITY;
```

dwell: This is used to select the unit of dwell that the tool stays at the bottom of *groove*. It could be either time (seconds) or number of workpiece revolutions. If omitted, this attribute shall be implementation dependent.

allowance : The *allowance* is a depth of material which shall be left on top of the surface defined by the associated manufacturing feature (in the direction of the surface normal). The remaining material shall be removed by the finishing operation.

5.4.4.2.1 Grooving rough

Roughing operation for grooving. This operation makes it possible to remove all material until the allowance measured from the features position is reached, disregarded special surface requirements.

```
ENTITY grooving_rough
   SUBTYPE OF (grooving);
WHERE
   WR1: EXISTS(SELF.allowance) AND (SELF.allowance >= 0.0);
END_ENTITY;
```

5.4.4.2.2 Grooving finish

This turning operation is used for finishing grooving. It makes it possible to remove all material until the feature's position is reached, applying an appropriate strategy to maintain the given tolerances. If *allowance* is given, other special operations like grinding shall be applied for removing the material left.

```
ENTITY grooving_finish
```

```
   SUBTYPE OF (grooving);
END_ENTITY;
```

5.4.4.2.3 Cutting in

This entity describes a turning operation for cutting into the material at a certain position without moving sideways. In this case the shape of the groove is dependent on the used tool. The *groove* machined by this operation is at most as wide as the tool. Since the requirements on the geometry of grooves machined by *cutting_in* operation are comparatively low, no roughing and finishing for *cutting_in* is distinguished. Figure 22 shows the *cutting_in* in radial and axial direction.

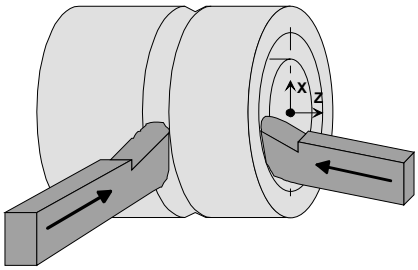


Figure 22: Cutting in (radial and axial direction)

If a shaft has a radial cut in the tool must not be moved in axial direction (i.e. perpendicular to the feed). If a *grooving_strategy* is applied and attribute *travel_length* is set, it will be ignored. The cutting in operation can be used for both, the cutting into the material and the cutting off. As mentioned in section 5.4.4.1 it is possible to generate a face by cutting off the end of a shaft. For that the cutting in operation has to be done up to the center of the workpiece. In this case the tool tip must be able to reach this center (intersection between the x and z axis).

```
ENTITY cutting_in
   SUBTYPE OF (grooving);
WHERE
   WR1: NOT(EXISTS(SELF.allowance));
END_ENTITY;
```

5.4.4.2.4 Dwell select

This type is used to select the unit of dwell, that the tool stays at the bottom of *groove*. It could be either time (seconds) or number of workpiece revolutions. For instance, depending on the *dwell_select* type, value of 1.5 means the tool has to stay for 1.5 seconds, or one and a half revolutions at the bottom.

```
TYPE dwell_select = SELECT (dwell_time, dwell_revolution);
END_TYPE;
```

5.4.4.2.5 Dwell time

This type is used to specify the time that the tool stays at the bottom of a *groove* in seconds.

```
TYPE dwell_time = time_measure;
END_TYPE;
```

5.4.4.2.6 Dwell revolution

This type is used to specify the number of workpiece revolutions that the tool stays at the bottom of a *groove*.

```
TYPE dwell_revolution = REAL;
END_TYPE;
```

5.4.4.3 Contouring

This operation may be used for turning of *outer_round* and *general_revolution*. It has to be distinguished between roughing and finishing operations. *Contouring* makes it possible to turn the contour of a given feature.

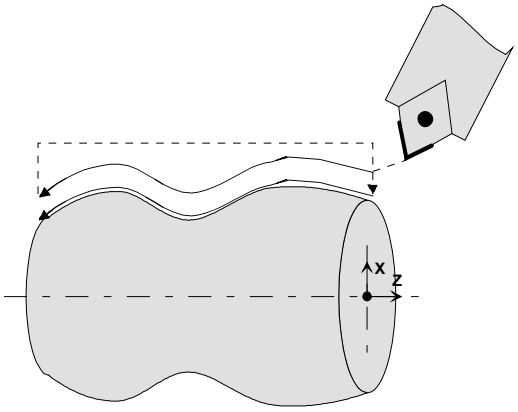


Figure 23: Contouring

```
ENTITY contouring
   ABSTRACT SUPERTYPE OF (ONEOF(contouring_rough, contouring_finish))
   SUBTYPE OF (turning_machining_operation);
   allowance : OPTIONAL length_measure;
END_ENTITY;
```

allowance : The *allowance* is a depth of material which shall be left on top of the surface defined by the associated manufacturing feature (in the direction of the surface normal). The remaining material shall be removed by the finishing operation.

5.4.4.3.1 Contouring rough

This turning operation is used for contour roughing. It makes it possible to remove of all material until the allowance measured from the position of the feature is reached, disregarded special surface requirements.

```
ENTITY contouring_rough
```



```
   SUBTYPE OF (contouring);
WHERE
   WR1: EXISTS(SELF.allowance) AND (SELF.allowance >= 0.0);
END_ENTITY;
```

5.4.4.3.2 Contouring finish

This turning operation is used for contour finishing. It makes it possible to remove all material until the feature's position is reached, applying an appropriate strategy to maintain the given tolerances. If *allowance* is given, other special operation like grinding shall be applied for removing the material left.

```
ENTITY contouring_finish
   SUBTYPE OF (contouring);
END_ENTITY;
```

5.4.4.4 Threading

This operation is used for turning threads. It has to be distinguished between roughing and finishing operations, leading to higher surface quality.

```
ENTITY threading
   ABSTRACT SUPERTYPE OF (ONEOF(threading_rough, threading_finish))
   SUBTYPE OF (turning_machining_operation);
   allowance : OPTIONAL length_measure;
END_ENTITY;
```

allowance : The *allowance* is a depth of material which shall be left on top of the surface defined by the associated manufacturing feature (in the direction of the surface normal). The remaining material shall be removed by the finishing operation.

5.4.4.4.1 Threading rough

Roughing operation for turning threads. This operation makes it possible to remove all material until the allowance is reached, disregarded special surface requirements.

```
ENTITY threading_rough
   SUBTYPE OF (threading);
WHERE
   WR1: EXISTS(SELF.allowance) AND (SELF.allowance >= 0.0);
END_ENTITY;
```

5.4.4.4.2 Threading finish

This turning operation is used for finishing a thread. It makes it possible to remove all material until the feature's position is reached, applying an appropriate strategy to maintain the given tolerances. If *allowance* is given, other special operation like grinding shall be applied for removing the material left.

```
ENTITY threading_finish
   SUBTYPE OF (threading);
END_ENTITY;
```

5.4.4.5 Knurling

This operation is used for machine knurls. Knurling is done with special tools. It has not to be distinguished between roughing and finishing operations.

```
ENTITY knurling
   SUBTYPE OF (turning_machining_operation);
END_ENTITY;
```

6 Conformance requirements

There is no need for a breakdown of conformance requirements into classes so far. If necessary this will be given in the later version of this standard.

Annex A:
(normative)

EXPRESS expanded listing

The following section shows the EXPRESS listing of ISO 14649-12: Process Data for Turning.

```
SCHEMA turning_schema;

(*
   Version : 14
   Date    : 24.10.2003
   Author  : ISO TC184/SC1/WG7
   Contact : Suk-Hwan Suh <shs@postech.ac.kr>
             Stefan Heusinger <stefan.heusinger@isw.uni-stuttgart.de>
*)


(* ************************************************************************** *)
(* Types from machining_schema                ISO 14649-10                    *)
(* ************************************************************************** *)

REFERENCE FROM machining_schema(
   axis2_placement_3d,
   bounded_curve,
   cartesian_point,
   direction,
   general_profile,
   identifier,
   in_process_geometry,
   label,
   length_measure,
   linear_profile,
   machine_functions,
   machining_operation,
   manufacturing_feature,
   material,
   open_profile,
   partial_area_definition,
   partial_circular_profile,
   plane_angle_measure,
   positive_length_measure,
   positive_ratio_measure,
   pressure_measure,
   property_parameter,
   rot_speed_measure,
   round_hole,
   speed_measure,
   taper_select,
   technology,
   thread,
   time_measure,
   toleranced_length_measure,
   two5D_manufacturing_feature,
```

```
   vee_profile,
   workingstep);

(* ************************************************************************ *)
(* Types from milling_ schema                ISO 14649-11                   *)
(* ************************************************************************ *)

REFERENCE FROM milling_schema(
   adaptive_control,
   approach_retract_strategy,
   process_model_list
);



(* * Turning features ***************************************************** *)

ENTITY turning_feature
   ABSTRACT SUPERTYPE OF (ONEOF(outer_round, revolved_feature, knurl))
   SUBTYPE OF (two5D_manufacturing_feature);
END_ENTITY;


ENTITY outer_round
   ABSTRACT SUPERTYPE OF (ONEOF (outer_diameter, outer_diameter_to_shoulder))
   SUBTYPE OF (turning_feature);
END_ENTITY;


ENTITY outer_diameter
   SUBTYPE OF (outer_round);
   diameter_at_placement : toleranced_length_measure;
   feature_length        : toleranced_length_measure;
   reduced_size          : OPTIONAL taper_select;
END_ENTITY;


ENTITY outer_diameter_to_shoulder
   SUBTYPE OF (outer_round);
   diameter_at_placement : toleranced_length_measure;
   v_shape_boundary      : vee_profile;
END_ENTITY;


ENTITY revolved_feature
   ABSTRACT SUPERTYPE OF (ONEOF (revolved_round, revolved_flat, groove,
  general_revolution))
   SUBTYPE OF (turning_feature);
   material_side : OPTIONAL direction;
   radius        : length_measure;
END_ENTITY;


ENTITY revolved_flat
   SUBTYPE OF (revolved_feature);
   flat_edge_shape : linear_profile;
END_ENTITY;


ENTITY revolved_round
```

```
   SUBTYPE OF (revolved_feature);
   rounded_edge_shape : partial_circular_profile;
END_ENTITY;


ENTITY groove
   SUBTYPE OF (revolved_feature);
   sweep : open_profile;
END_ENTITY;


ENTITY general_revolution
   SUBTYPE OF (revolved_feature);
   outer_edge_profile : general_profile;
END_ENTITY;


ENTITY knurl
   ABSTRACT SUPERTYPE OF (ONEOF(straight_knurl, diagonal_knurl, diamond_knurl,
  tool_knurl))
   SUBTYPE OF (turning_feature);
   base_feature     : turning_feature;
   partial_profile  : OPTIONAL partial_area_definition;
   tooth_depth      : OPTIONAL toleranced_length_measure;
   diametral_pitch  : OPTIONAL toleranced_length_measure;
   root_fillet      : OPTIONAL toleranced_length_measure;
   number_of_teeth  : OPTIONAL INTEGER;
   major_diameter   : OPTIONAL toleranced_length_measure;
   nominal_diameter : OPTIONAL toleranced_length_measure;
END_ENTITY;


ENTITY straight_knurl
   SUBTYPE OF (knurl);
END_ENTITY;


ENTITY diagonal_knurl
   SUBTYPE OF (knurl);
   helix_angle : plane_angle_measure;
END_ENTITY;


ENTITY diamond_knurl
   SUBTYPE OF (knurl);
   helix1_angle : plane_angle_measure;
   helix2_angle : OPTIONAL plane_angle_measure;
END_ENTITY;


ENTITY tool_knurl
   SUBTYPE OF (knurl);
END_ENTITY;



(* ***** Turning workingstep ********************************************** *)

ENTITY turning_workingstep
   SUBTYPE OF (workingstep);
```

```
   its_features  : LIST [2:?] OF manufacturing_feature;
   its_operation : turning_machining_operation;
   its_effect    : OPTIONAL in_process_geometry;
END_ENTITY;



(* ***** Turning technology ********************************************** *)

ENTITY turning_technology
   SUBTYPE OF (technology);
   spindle_speed             : speed_select;
   feedrate_per_revolution   : OPTIONAL feed_select;
   sync_spindle_and_z_feed   : BOOLEAN;
   inhibit_feedrate_override : BOOLEAN;
   inhibit_spindle_override  : BOOLEAN;
   its_adaptive_control      : OPTIONAL adaptive_control;
END_ENTITY;


TYPE speed_select = SELECT (const_spindle_speed, const_cutting_speed);
END_TYPE;


ENTITY const_spindle_speed;
   rot_speed : rot_speed_measure;
END_ENTITY;


ENTITY const_cutting_speed;
   speed     : speed_measure;
   max_speed : OPTIONAL rot_speed_measure;
END_ENTITY;


TYPE feed_select = SELECT (feed_velocity_type, feed_per_rev_type);
END_TYPE;


TYPE feed_velocity_type = speed_measure;
END_TYPE;


TYPE feed_per_rev_type = REAL;
END_TYPE;



(* ***** Turning machine functions ************************************** *)

ENTITY turning_machine_functions
   SUBTYPE OF (machine_functions);
   coolant               : BOOLEAN;
   coolant_type          : OPTIONAL coolant_select;
   coolant_pressure      : OPTIONAL pressure_measure;
   axis_clamping         : LIST [0:?] OF identifier;
   chip_removal          : OPTIONAL BOOLEAN;
   oriented_spindle_stop : OPTIONAL direction;
   its_process_model     : OPTIONAL process_model_list;
   other_functions       : SET [0:?] OF property_parameter;
```

```
   tail_stock            : OPTIONAL BOOLEAN;
   steady_rest           : OPTIONAL BOOLEAN;
   follow_rest           : OPTIONAL BOOLEAN;
END_ENTITY;


TYPE coolant_select = ENUMERATION OF (flood, mist, through_tool);
END_TYPE;



(* ***** Turning strategy ********************************************** *)

ENTITY turning_machining_strategy
   ABSTRACT SUPERTYPE OF (ONEOF (unidirectional_turning, bidirectional_turning,
  thread_strategy, contour_turning, grooving_strategy,
  explicit_turning_strategy));
   overcut_length        : OPTIONAL length_measure;
   allow_multiple_passes : OPTIONAL BOOLEAN;
   cutting_depth         : OPTIONAL LIST[0:?] OF length_measure;
   variable_feedrate     : OPTIONAL positive_ratio_measure;
END_ENTITY;


ENTITY unidirectional_turning
   SUBTYPE OF (turning_machining_strategy);
   feed_direction      : OPTIONAL direction;
   back_path_direction : OPTIONAL direction;
   lift_direction      : OPTIONAL direction;
   stepover_direction  : OPTIONAL direction;
   lift_height         : OPTIONAL length_measure;
   lift_feed           : OPTIONAL feed_select;
   stepover_feed       : OPTIONAL feed_select;
END_ENTITY;


ENTITY bidirectional_turning
   SUBTYPE OF (turning_machining_strategy);
   feed_direction     : OPTIONAL direction;
   stepover_direction : OPTIONAL direction;
   stepover_feed      : OPTIONAL feed_select;
END_ENTITY;


ENTITY contour_turning
   SUBTYPE OF (turning_machining_strategy);
   back_path_direction    : OPTIONAL direction;
   lift_direction         : OPTIONAL direction;
   stepover_direction     : OPTIONAL direction;
   lift_height            : OPTIONAL length_measure;
   lift_feed              : OPTIONAL feed_select;
   stepover_feed          : OPTIONAL feed_select;
   variable_stepover_feed : OPTIONAL positive_ratio_measure;
END_ENTITY;


ENTITY thread_strategy
   SUBTYPE OF (turning_machining_strategy);
   cut_in_amount_function : thread_cut_depth_type;
   threading_direction    : threading_direction_type;
```

```
   path_return_angle      : OPTIONAL plane_angle_measure;
   lift_height            : OPTIONAL positive_length_measure;
END_ENTITY;


TYPE thread_cut_depth_type = ENUMERATION OF (constant_depth, variable_depth,
  constant_removal_amount);
END_TYPE;


TYPE threading_direction_type = ENUMERATION OF (left, right, center, left_zigzag,
  right_zigzag);
END_TYPE;


ENTITY grooving_strategy
   SUPERTYPE OF (multistep_grooving_strategy)
   SUBTYPE OF (turning_machining_strategy);
   grooving_direction : OPTIONAL direction;
   travel_distance    : OPTIONAL length_measure;
END_ENTITY;


ENTITY  multistep_grooving_strategy
   SUBTYPE OF (grooving_strategy);
   retract_distance : length_measure;
END_ENTITY;


ENTITY explicit_turning_strategy
   SUBTYPE OF (turning_machining_strategy);
END_ENTITY;



(* ***** Turning operations ********************************************** *)

ENTITY turning_machining_operation
   ABSTRACT SUPERTYPE OF (ONEOF(facing, grooving, contouring, threading,
  knurling))
   SUBTYPE OF (machining_operation);
   approach               : OPTIONAL approach_retract_strategy;
   retract                : OPTIONAL approach_retract_strategy;
   its_machining_strategy : OPTIONAL turning_machining_strategy;
END_ENTITY;


ENTITY facing
   ABSTRACT SUPERTYPE OF (ONEOF(facing_rough, facing_finish))
   SUBTYPE OF (turning_machining_operation);
   allowance : OPTIONAL length_measure;
END_ENTITY;


ENTITY facing_rough
   SUBTYPE OF (facing);
WHERE
   WR1: EXISTS(SELF.allowance) AND (SELF.allowance >= 0.0);
END_ENTITY;
```

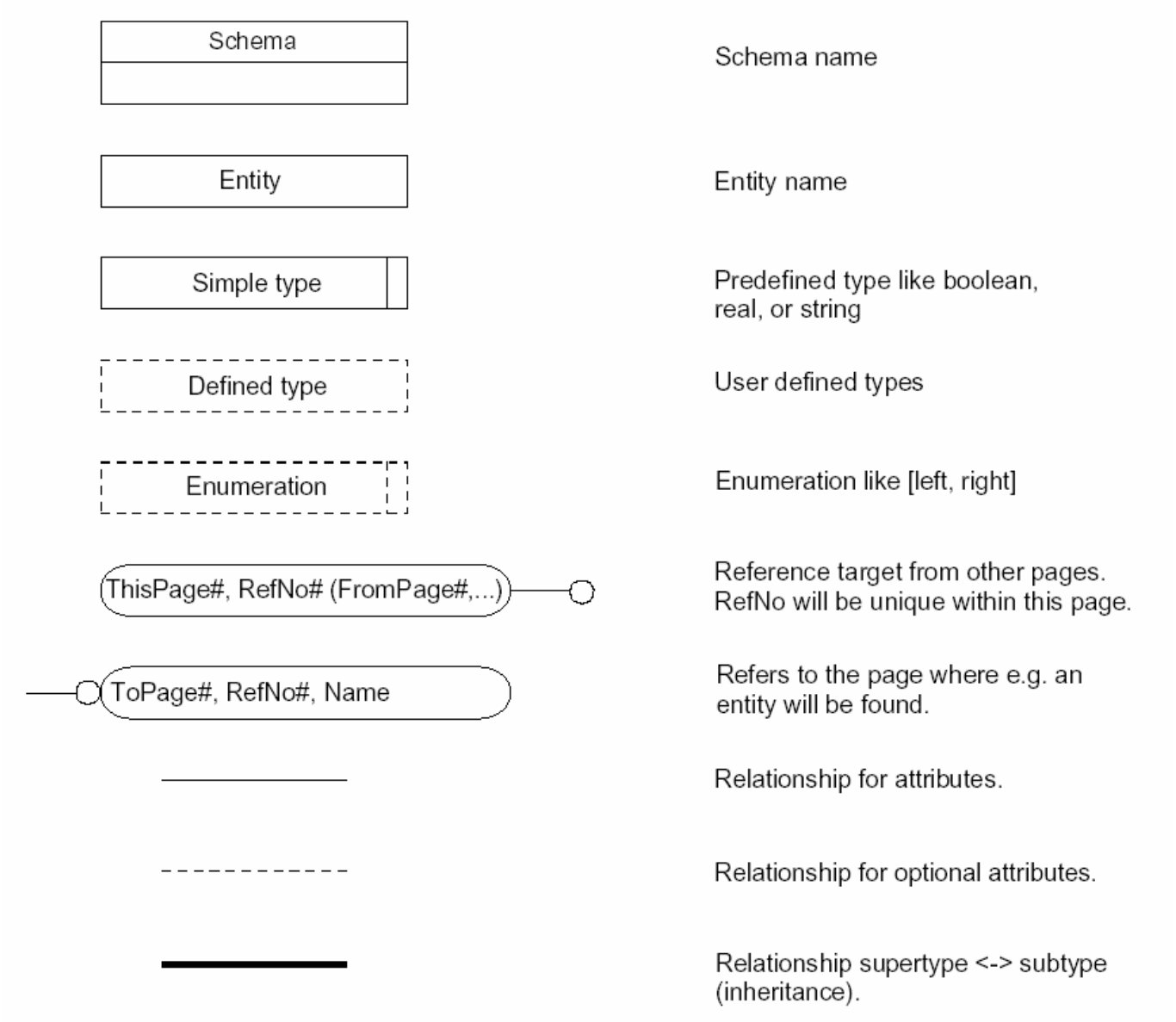
```
ENTITY facing_finish
   SUBTYPE OF (facing);
END_ENTITY;


ENTITY grooving
   ABSTRACT SUPERTYPE OF (ONEOF(grooving_rough, grooving_finish, cutting_in))
   SUBTYPE OF (turning_machining_operation);
   dwell     : OPTIONAL dwell_select;
   allowance : OPTIONAL length_measure;
END_ENTITY;


ENTITY grooving_rough
   SUBTYPE OF (grooving);
WHERE
   WR1: EXISTS(SELF.allowance) AND (SELF.allowance >= 0.0);
END_ENTITY;


ENTITY grooving_finish
   SUBTYPE OF (grooving);
END_ENTITY;


ENTITY cutting_in
   SUBTYPE OF (grooving);
WHERE
   WR1: NOT(EXISTS(SELF.allowance));
END_ENTITY;


TYPE dwell_select = SELECT (dwell_time, dwell_revolution);
END_TYPE;


TYPE dwell_time = time_measure;
END_TYPE;


TYPE dwell_revolution = REAL;
END_TYPE;


ENTITY contouring
   ABSTRACT SUPERTYPE OF (ONEOF(contouring_rough, contouring_finish))
   SUBTYPE OF (turning_machining_operation);
   allowance : OPTIONAL length_measure;
END_ENTITY;


ENTITY contouring_rough
   SUBTYPE OF (contouring);
WHERE
   WR1: EXISTS(SELF.allowance) AND (SELF.allowance >= 0.0);
END_ENTITY;


ENTITY contouring_finish
```

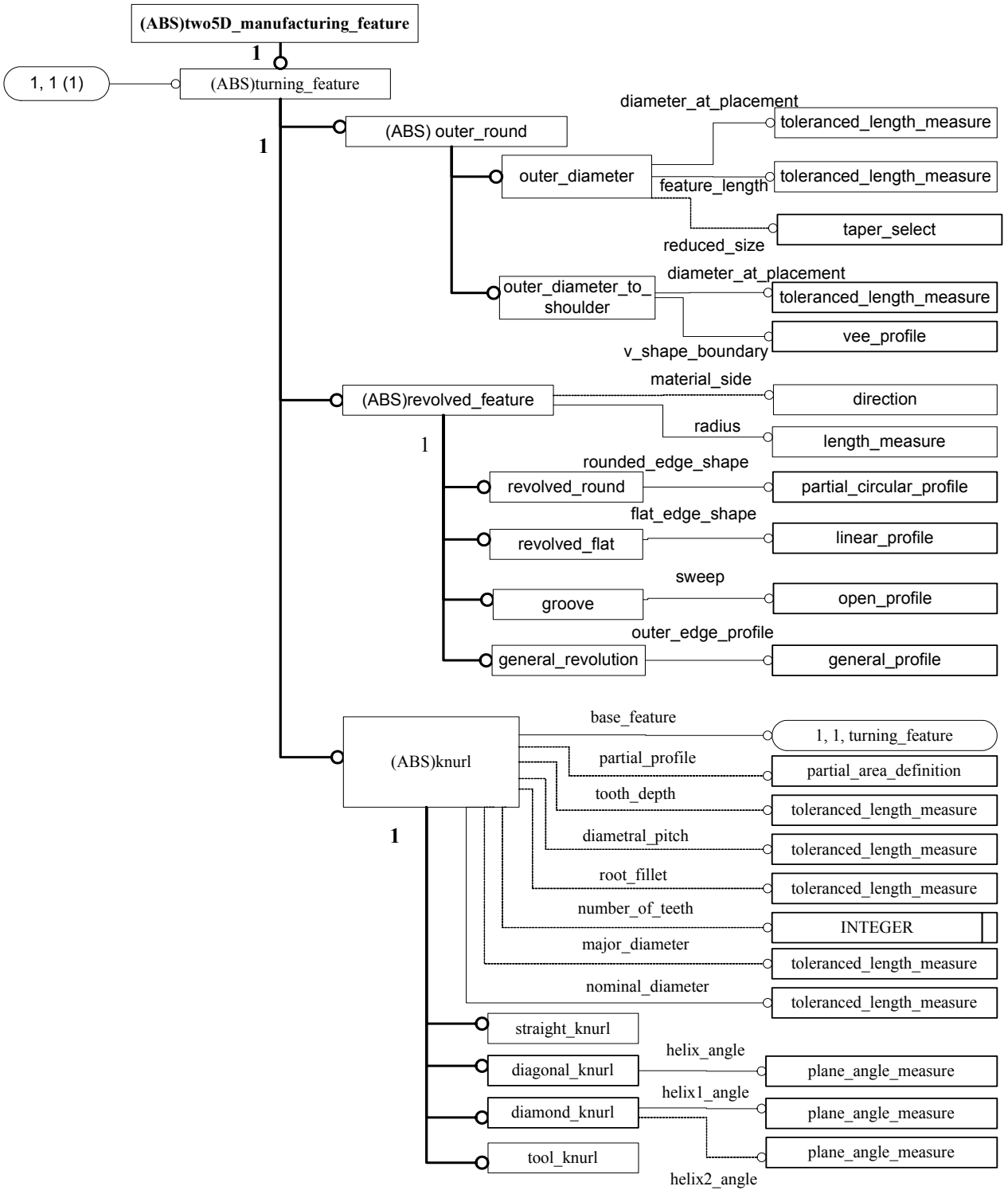
```
   SUBTYPE OF (contouring);
END_ENTITY;


ENTITY threading
   ABSTRACT SUPERTYPE OF (ONEOF(threading_rough, threading_finish))
   SUBTYPE OF (turning_machining_operation);
   allowance : OPTIONAL length_measure;
END_ENTITY;


ENTITY threading_rough
   SUBTYPE OF (threading);
WHERE
   WR1: EXISTS(SELF.allowance) AND (SELF.allowance >= 0.0);
END_ENTITY;


ENTITY threading_finish
   SUBTYPE OF (threading);
END_ENTITY;


ENTITY knurling
   SUBTYPE OF (turning_machining_operation);
END_ENTITY;


END_SCHEMA; (* turning_schema *)
```

Annex B:
(informative)

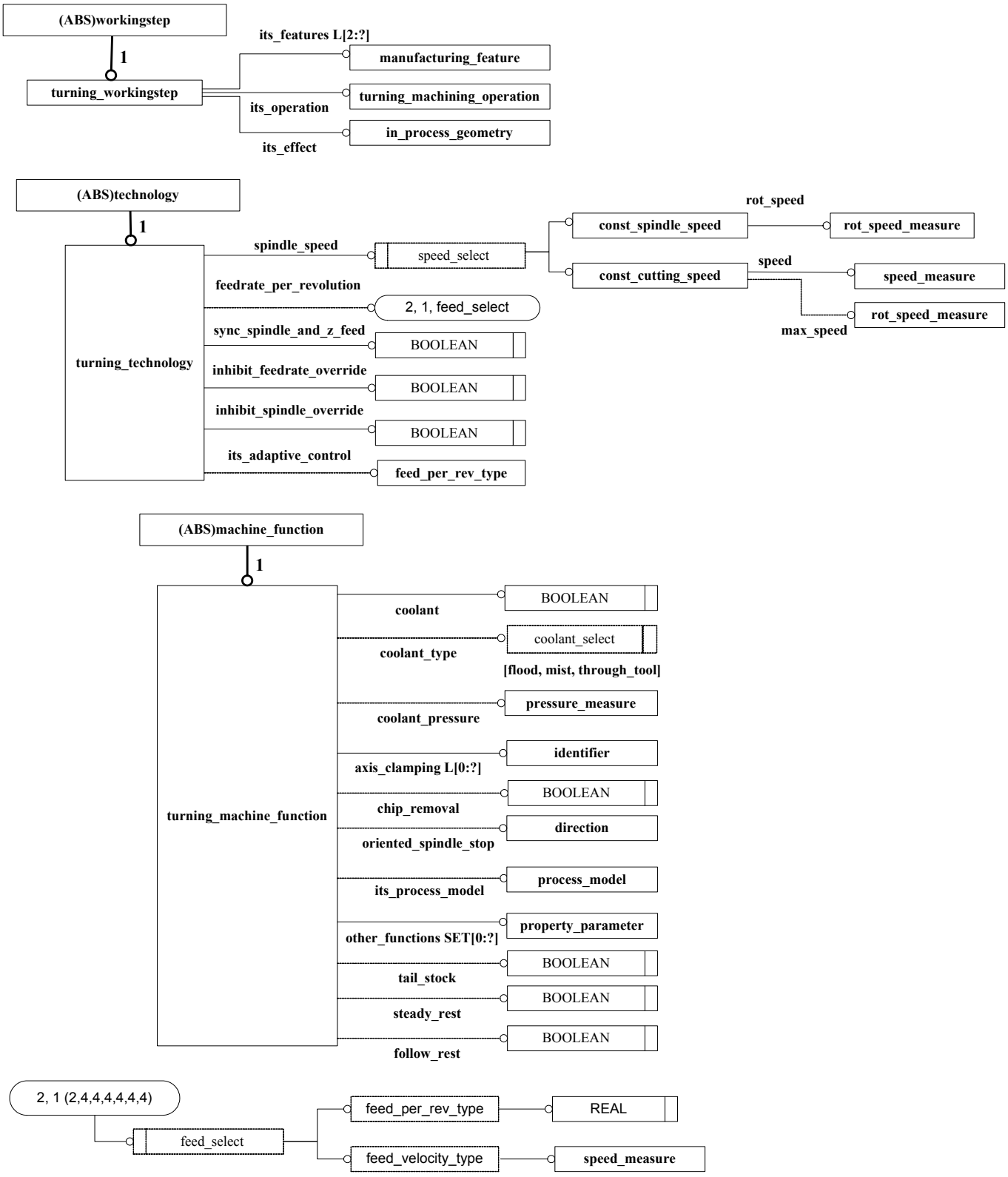
EXPRESS-G diagram

The following section shows the EXPRESS-G figures of ISO 14649-12: Process Data for Turning. According to the notation of EXPRESS-G the used symbols and their respective meaning are listed in brief.

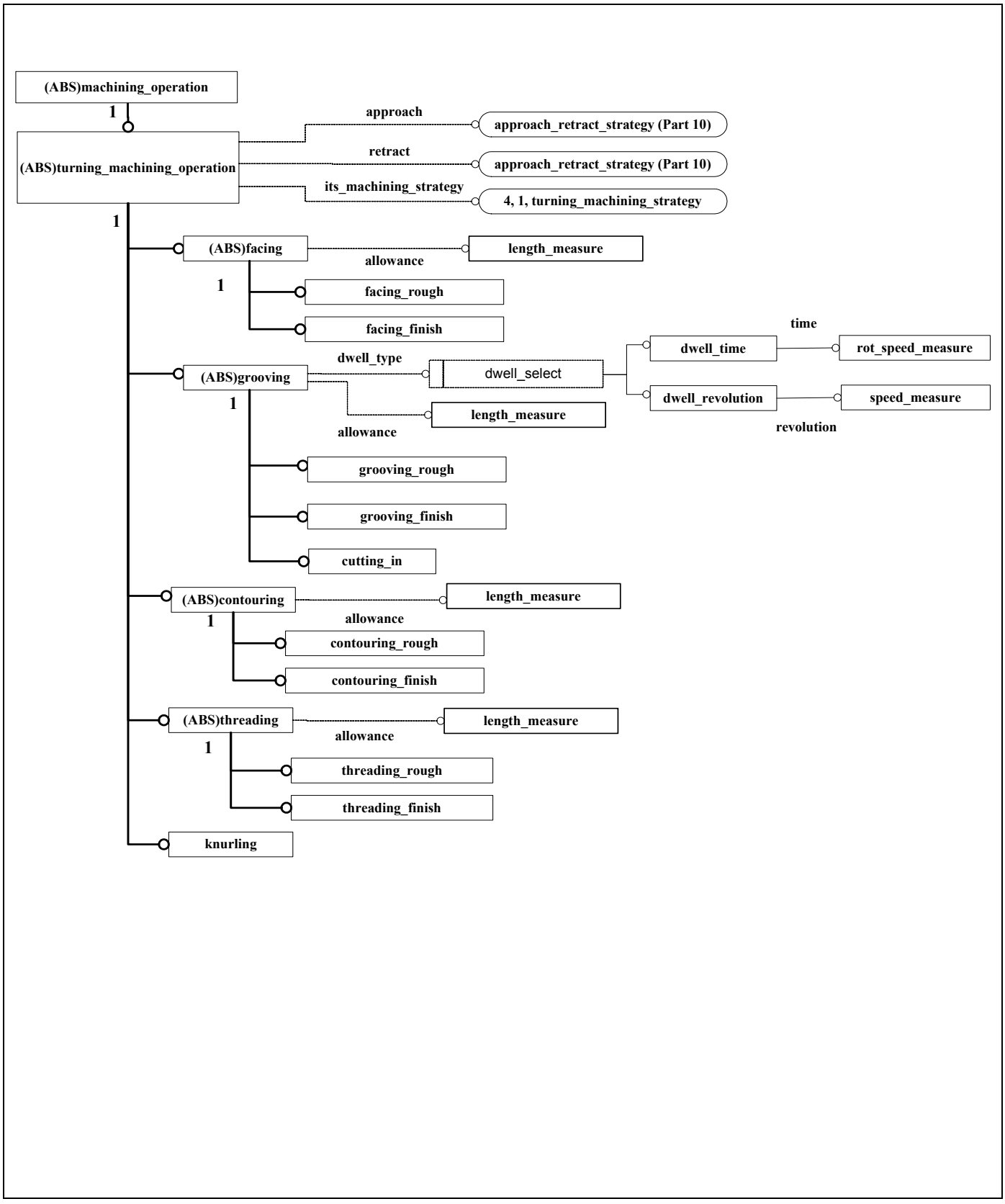




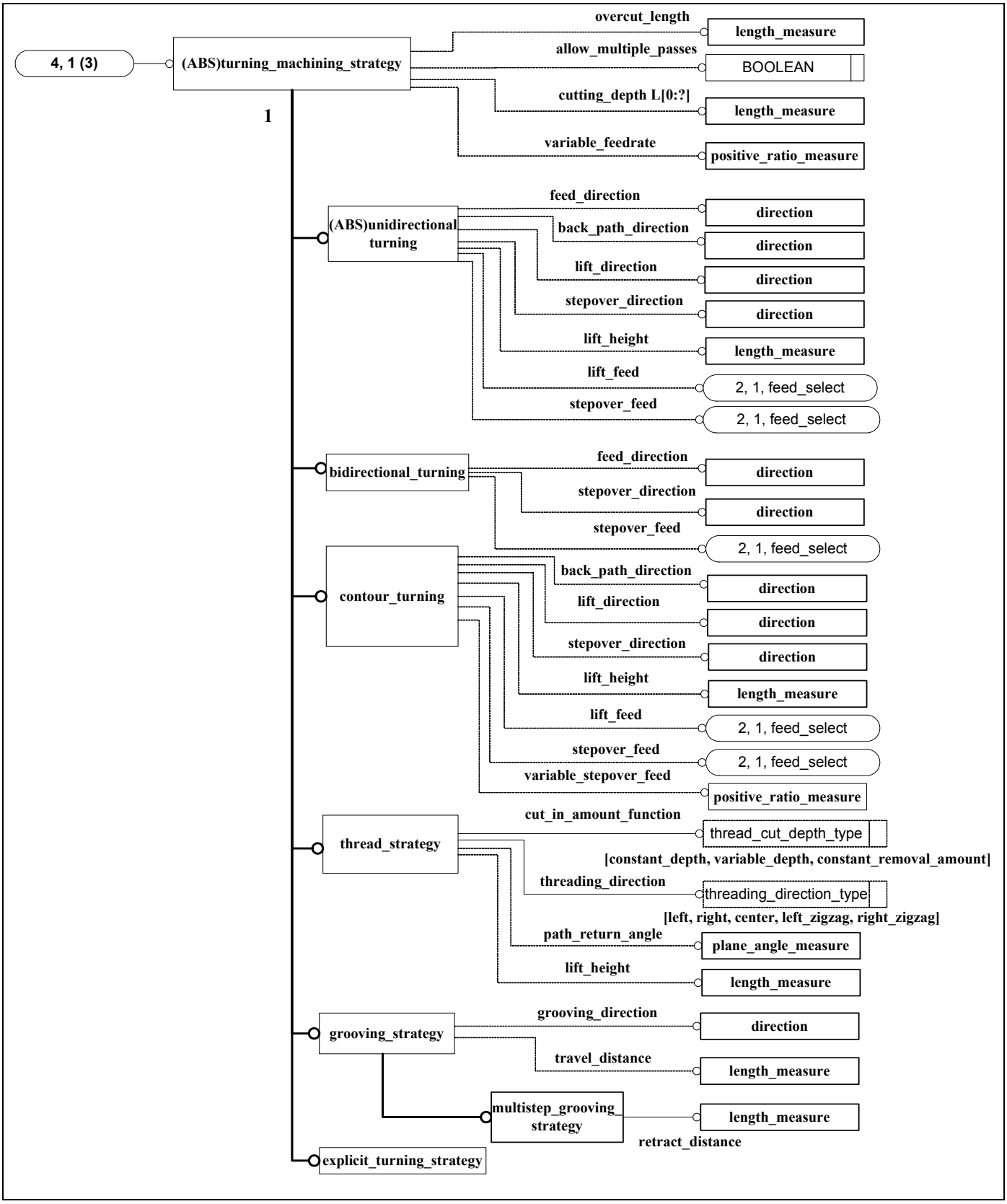
Page B.1



Page B.2



Page B.3



Annex C:
(informative)

Turning specific features

Turning features defined in the main text of this document are referenced from AP 224 of ISO 10303; i.e., fully compliant with ISO 10303 AP 224. These features are taken as standard for STEP-NC data model, for the sake of compliance with the existing standard ISO 10303 AP 224. In addition to these features, there exist turning specific features, such as *circular_face* and *cut_in*, which are often used in the industrial practice. Thus, these features can be formally implemented by the vendor. In Annex C, definition of turning specific features and relationship with *turning_features* are given.

C.1 Circular face

A *circular_face* is a plane face in radial direction. It can be used to describe plane surfaces perpendicular to the rotation axis with special surface requirements. How the circular face looks like depends on its lower diameter (Figure C.1). If the lower diameter is omitted or zero, the face will be the end wall of the workpiece (*end_face*; left figure). Otherwise it is a kind of transition (*step_face*; right figure) between two features.

NOTE: These two kinds of *circular_face* (*end_face* and *step_face*) can be represented by *revolved_flat* and *outer_diameter_to_shoulder*, respectively.

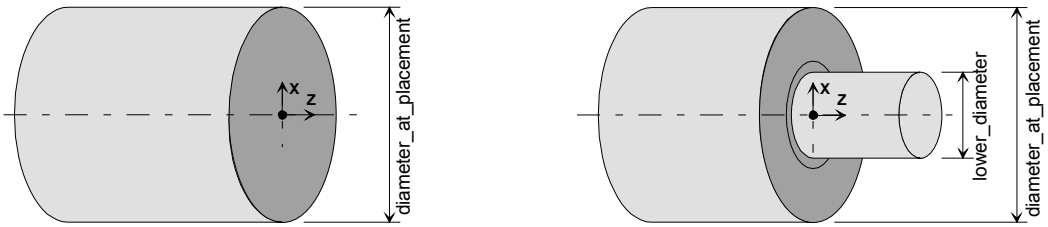


Figure C.1 : Circular face (left: end face; right: step face).

```
ENTITY circular_face;
   SUBTYPE OF (turning_feature);
   diameter_at_placement : toleranced_length_measure;
   lower_diameter        : OPTIONAL toleranced_length_measure;
   transition            : OPTIONAL bottom_transition;
   its_condition         : OPTIONAL face_radiused;
   (* Informal proposition : If lower_diameter is not omitted, it has to be lower
  than diameter_at_placement.
   *)
END_ENTITY;
```

diameter_at_placement : This attribute describes the diameter at the side of the feature, where the origin of the coordinate system of the feature's placement is defined.

lower_diameter : The optional property *lower_diameter* is the smaller diameter of the feature. If this parameter is omitted or zero, no boss will be left around the center of the rotation axis. In this case the property *transition* will not be evaluated.

transition : This optional attribute describes the shape of the transition at the lower diameter (bottom) of the feature, such as *bottom_transition* see section as defined in C.1.2.

its_condition : This optional attribute makes it possible to define the shape of the surface, such as face_radiused as defined in C.1.1. If omitted, a straight wall perpendicular to the z-axis is assumed.

C.1.1 Face radiused

This entity makes it possible to describe a circular face with a radiused shape. It is obvious that the applied circle (or ellipse) has to be symmetrical to the z-axis.

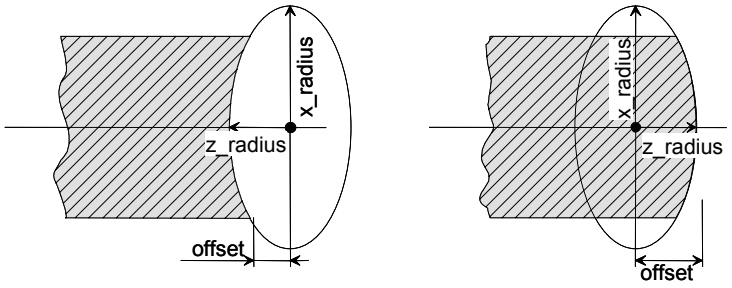


Figure C.2 : Circular face with radiused shape.

```
ENTITY face_radiused;
   offset   : length_measure;
   z_radius : toleranced_length_measure;
   x_radius : OPTIONAL toleranced_length_measure;
END_ENTITY;
```

offset : This attribute gives the distance between the placement of the wall and the center of the ellipse. Because the workpiece rotates, the shape will be symmetrical referring to the z-axis. Thus, only the distance in direction of z is given. If this radius points from the outside to the surface, the shape will be concave, in the other case convex. Positive values are pointing in positive z-direction.

z_radius : The attribute *z_radius* is the radius of the ellipse's semi axis, which is parallel to the z-axis. If this radius points from the outside to the surface, the shape will be concave, in the other case convex. Positive values are pointing in positive z-direction.

x_radius : The optional attribute *x_radius* is the radius of the ellipse's semi axis, which is parallel to the x-axis. If omitted or both radii are equal, the ellipse builds as special case a circle.

C.1.2 Bottom transition

This entity is the abstract base class for the actual description of the transition at the bottom of a turning feature. Two bottom transitions are foreseen: a slope and a round.

```
ENTITY bottom_transition;
```

```
    ABSTRACT SUPERTYPE OF (ONEOF(bottom_transition_slope,
   bottom_transition_round));
END_ENTITY;
```

C.1.2.1 Bottom_transition_slope

This entity describes the linear transition from the vertical to the horizontal part at the bottom of a turning feature. The shape of the slope is determined by its offset in the horizontal and its angle against the bottom of the feature. The origin of the coordinate system is at the point where the z-axis intersects the plane of the wall.

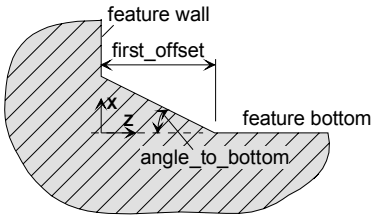


Figure C.3 : Slope at the bottom.

```
ENTITY bottom_transition_slope;
   SUBTYPE OF (bottom_transition);
   angle_to_bottom : plane_angle_measure;
   first_offset    : toleranced_length_measure;
END_ENTITY;
```

angle_to_bottom : This attribute describes an angle that is 180 degree minus the angle between the slope and the bottom of the feature.

first_offset : This attribute defines the distance between the wall and the bottom along the z-axis.

C.1.2.2 Bottom_transition_round

This entity describes the curved transition from the vertical to the horizontal part at the bottom of a turning feature. The shape of the curve is determined by two offsets and its radius.

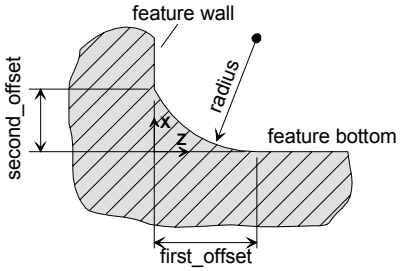


Figure C.4 : Round between bottom and wall.

```
ENTITY bottom_transition_round;
   SUBTYPE OF (bottom_transition);
   radius        : toleranced_length_measure;
   first_offset  : OPTIONAL toleranced_length_measure;
```

```
   second_offset : OPTIONAL toleranced_length_measure;
END_ENTITY;
```

radius : Radius of the bottom transition.

first_offset : This attribute defines the distance between the wall and the bottom along the z-axis. If omitted the curve is tangent to the bottom. Note that it is not allowed that the attribute *first_offset* has a larger value than the radius.

second_offset : This attribute defines the distance between the wall and the bottom along the x-axis. If omitted the curve is tangent to the wall. Note that it is not allowed that the attribute *second_offset* has a larger value than the radius.

C.2 Cut in

Cut_in is a kind of slot or groove; i.e, the geometrical shape of *cut_in* is similar to that of groove, but its shape is identical to the shape of the used tool. The orientation of the *cut_in* can be defined by attribute *direction*, and the amount of *cut_in* along the direction by attribute *depth*. The coordinate system of the feature is placed at the point of the part surface where the tool first contacts along *cut_in_direction* without regarding tool gouging.

Note 1: *Cut_in* can be represented by groove defined in section 5.2.3.3.

Note 2: *Cut_in* may be represented by a toolpath_feature.

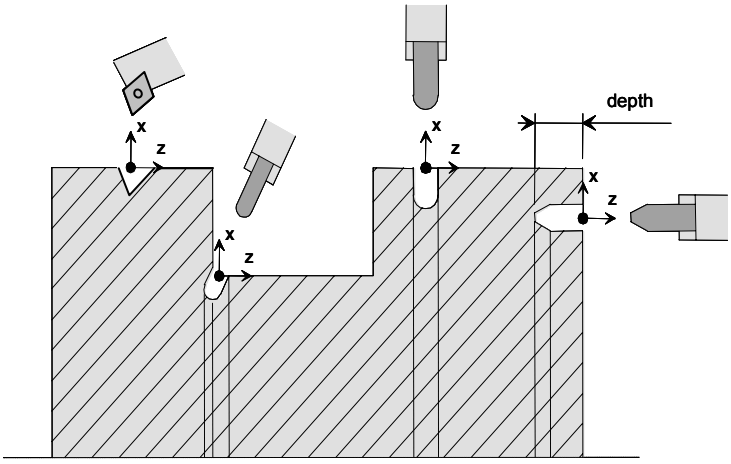


Figure C.5 : Cut_in.

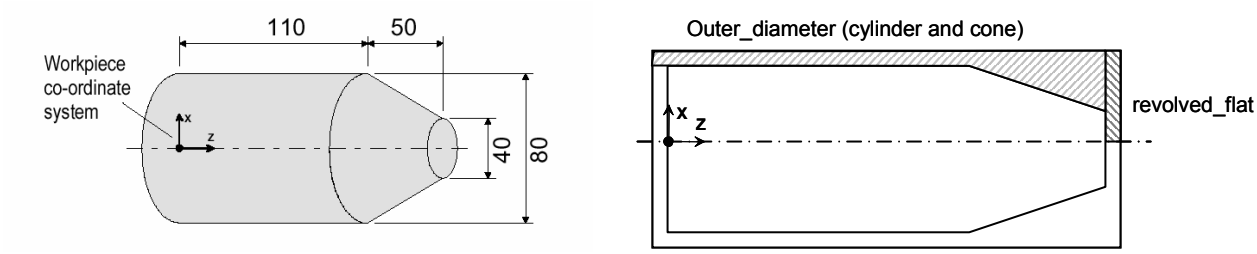
```
ENTITY cut_in;
   SUBTYPE OF (turning_feature);
   depth            : toleranced_length_measure;
   cut_in_direction : OPTIONAL direction;
END_ENTITY;
```

depth : The attribute depth stands for the depth of the *cut_in* measured from its coordinate system in the given direction. By using an appropriate tool (length), it is possible to cut off the end of a bar (shaft), if the z-axis can be reached. For example, if the tool direction is perpendicular to the z-axis and the depth is equal to the radius of the shaft, the end of the shaft is cut off.

cut_in_direction : This optional attribute makes it possible to specify the direction of the *cut_in*. If omitted, the direction of the tool movement will be perpendicular to the z-axis.

Annex D:
(informative)
Simple turning example

The following picture shows a simple workpiece. The workpiece has an end face and a cone on its right side. The measurement can be found in the picture. Below the listing of the NC program is printed.




```
ISO-10303-21;

HEADER;
FILE_DESCRIPTION(('ISO14649','SIMPLE EXAMPLE OF NC PROGRAM FOR TURNING: REVOLVED_FLAT,
OUTER_DIAMETER.'),'1');
FILE_NAME('EXAMPLE1.STP','2003-04-08',('STEFAN HEUSINGER','MICHAEL WOSNIK'),('ISW UNI-
STUTTGART','GERMANY'),"","");
FILE_SCHEMA(('MACHINING_SCHEMA','MILLING_SCHEMA','TURNING_SCHEMA','TURNING_MACHINE_TOOL
_SCHEMA'));
ENDSEC;

DATA;

/* ************************************************** */
/* ***** Workpiece definition ***** */
#1=WORKPIECE('SIMPLE WORKPIECE',#2,0.010,$,$,$,());
#2=MATERIAL('DIN EN 10027-1','E 295',(#3));
#3=NUMERIC_PARAMETER('ELASTIC MODULUS',2.E11,'pa');

/* ************************************************** */
/* ***** Manufacturing features ***** */
#10=REVOLVED_FLAT('END FACE',#1,(#20,#21),#70,#80,0.000,#91);
#11=OUTER_DIAMETER('CONE',#1,(#22,#23),#76,#83,#93,#95);
#12=OUTER_DIAMETER('CYLINDER',#1,(#22,#23),#78,#72,#74,$);

/* ************************************************** */
/* ***** Turning operations ***** */
#20=FACING_ROUGH($,$,'ROUGH END FACE',$,$,#100,#41,#40,#52,#53,#50,0.500);
#21=FACING_FINISH($,$,'FINISH END FACE',$,$,#110,#42,#40,#52,#53,#51,0.000);
#22=CONTOURING_ROUGH($,$,'ROUGH CONTOUR',$,$,#100,#43,#40,#56,#56,#54,0.500);
#23=CONTOURING_FINISH($,$,'FINISH CONTOUR',$,$,#110,#44,#40,#56,#56,#55,0.000);

/* ************************************************** */
/* ***** Project ***** */
#29=PROJECT('TURNING EXAMPLE 1',#30,(#1),$,$,$);
#30=WORKPLAN('MAIN WORKPLAN',(#31,#32,#33,#34),$,#37,$);
```

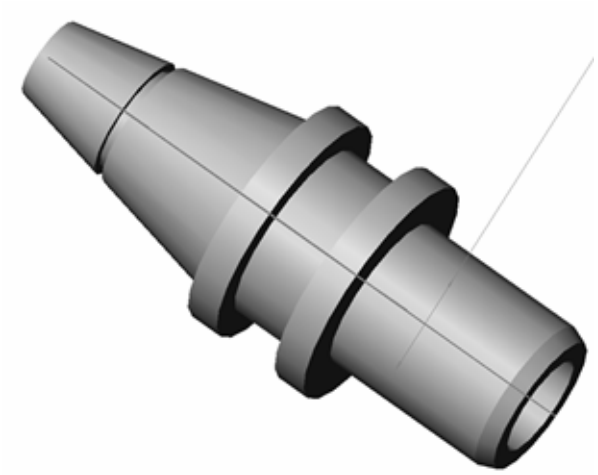
```
#31=MACHINING_WORKINGSTEP('WS ROUGH END FACE',#63,#10,#20,$);
#32=MACHINING_WORKINGSTEP('WS FINISH END FACE',#63,#10,#21,$);
#33=TURNING_WORKINGSTEP('WS ROUGH CONTOUR',#63,(#11,#12),#22,$);
#34=TURNING_WORKINGSTEP('WS FINISH CONTOUR',#63,(#11,#12),#23,$);
#37=SETUP('SETUP FOR TURNING EXAMPLE 1',$,#63,(#38));
#38=WORKPIECE_SETUP(#1,#64,$,$,());

/* ************************************************** */
/* ***** Functions / Technology ***** */
#40=TURNING_MACHINE_FUNCTIONS(.T.,$,$,(),.F.,$,$,(),$,$,$);
#41=TURNING_TECHNOLOGY($,.TCP.,#45,0.300,.F.,.F.,.F.,$);
#42=TURNING_TECHNOLOGY($,.TCP.,#46,0.200,.F.,.F.,.F.,$);
#43=TURNING_TECHNOLOGY($,.TCP.,#47,0.300,.F.,.F.,.F.,$);
#44=TURNING_TECHNOLOGY($,.TCP.,#48,0.200,.F.,.F.,.F.,$);
#45=CONST_SPINDLE_SPEED(5.000);
#46=CONST_CUTTING_SPEED(2.500,10.000);
#47=CONST_CUTTING_SPEED(2.500,10.000);
#48=CONST_CUTTING_SPEED(2.200,10.000);

/* ************************************************** */
/* ***** Strategies ***** */
#50=UNIDIRECTIONAL_TURNING($,$,(3.000),$,$,#82,$,$,2.000,$,$);
#51=UNIDIRECTIONAL_TURNING($,$,(0.500),$,$,#82,$,$,2.000,$,$);
#52=AP_RETRACT_TANGENT($,60.000);
#53=AP_RETRACT_ANGLE($,100.000,2.000);
#54=UNIDIRECTIONAL_TURNING($,$,(3.000),$,$,$,$,$,2.000,$,$);
#55=CONTOUR_TURNING($,$,(0.500),$,$,#81,$,$,$,$,$);
#56=AP_RETRACT_ANGLE($,45.000,4.000);

/* ************************************************** */
/* ***** Placements / Lengths / Planes ***** */
#63=PLANE('SECURITY PLANE',#68);
#64=AXIS2_PLACEMENT_3D('WORKPIECE',#65,#66,#67);
#65=CARTESIAN_POINT('WORKPIECE: LOCATION',(0.000,0.000,0.000));
#66=DIRECTION('WORKPIECE: AXIS',(0.000,0.000,1.000));
#67=DIRECTION('WORKPIECE: REF_DIRECTION',(1.000,0.000,0.000));
#68=AXIS2_PLACEMENT_3D('SECURITY PLANE',#69,$,$);
#69=CARTESIAN_POINT('SECPLANE: LOCATION',(90.000,0.000,200.000));
#70=AXIS2_PLACEMENT_3D('PLACEMENT END FACE',#71,$,$);
#71=CARTESIAN_POINT('END FACE: LOCATION',(0.000,0.000,160.000));
#72=TOLERANCED_LENGTH_MEASURE(80.000,#73);
#73=PLUS_MINUS_VALUE(0.100,0.100,1);
#74=TOLERANCED_LENGTH_MEASURE(110.000,#75);
#75=PLUS_MINUS_VALUE(0.100,0.100,1);
#76=AXIS2_PLACEMENT_3D('PLACEMENT CONE',#77,$,$);
#77=CARTESIAN_POINT('CONE: LOCATION',(0.000,0.000,160.000));
#78=AXIS2_PLACEMENT_3D('PLACEMENT CYLINDER',#79,$,$);
#79=CARTESIAN_POINT('CYLINDER: LOCATION',(0.000,0.000,110.000));
#80=DIRECTION('END FACE: FRONT',(0.000,0.000,-1.000));
#81=DIRECTION('STEPOVER DIRECTION FOR CONTOUR',(1.,0.,0.));
#82=DIRECTION('FACING DIRECTION',(-1.000,0.000,0.000));
#83=TOLERANCED_LENGTH_MEASURE(40.000,#90);
#89=PLUS_MINUS_VALUE(0.000,0.200,1);
#91=LINEAR_PROFILE($,#92);
#92=NUMERIC_PARAMETER('LINEAR PROFILE LENGTH',20.000,'mm');
#93=TOLERANCED_LENGTH_MEASURE(50.000,#94);
#94=PLUS_MINUS_VALUE(0.100,0.100,1);
#95=DIAMETER_TAPER(#96);
#96=TOLERANCED_LENGTH_MEASURE(80.000,#97);
```

```
#97=PLUS_MINUS_VALUE(0.100,0.100,1);

/* ************************************************** */
/* ***** Tools ***** */
#100=TURNING_MACHINE_TOOL('ROUGHING TOOL',#101,(#102),$,$,$);
#101=GENERAL_TURNING_TOOL(#103,.LEFT.,$,$,$,$);
#102=CUTTING_COMPONENT(50.,#104,$,$,$);
#103=TURNING_TOOL_DIMENSION($,$,$,10.000,$,20.000,$,0.300,$);
#104=MATERIAL('T15K6','CEMENT CARBIDE',(#105));
#105=NUMERIC_PARAMETER('ELASTIC MODULUS',3.E11,'pa');
#110=TURNING_MACHINE_TOOL('FINISHING TOOL',#111,(#112),$,$,$);
#111=GENERAL_TURNING_TOOL(#113,.LEFT.,$,$,$,$);
#112=CUTTING_COMPONENT(50.,#114,$,$,$);
#113=TURNING_TOOL_DIMENSION($,$,$,35.000,$,25.000,$,0.300,$);
#114=MATERIAL('T15K6','CEMENT CARBIDE',(#115));
#115=NUMERIC_PARAMETER('ELASTIC MODULUS',3.E11,'pa');
ENDSEC;

END-ISO-10303-21;
```

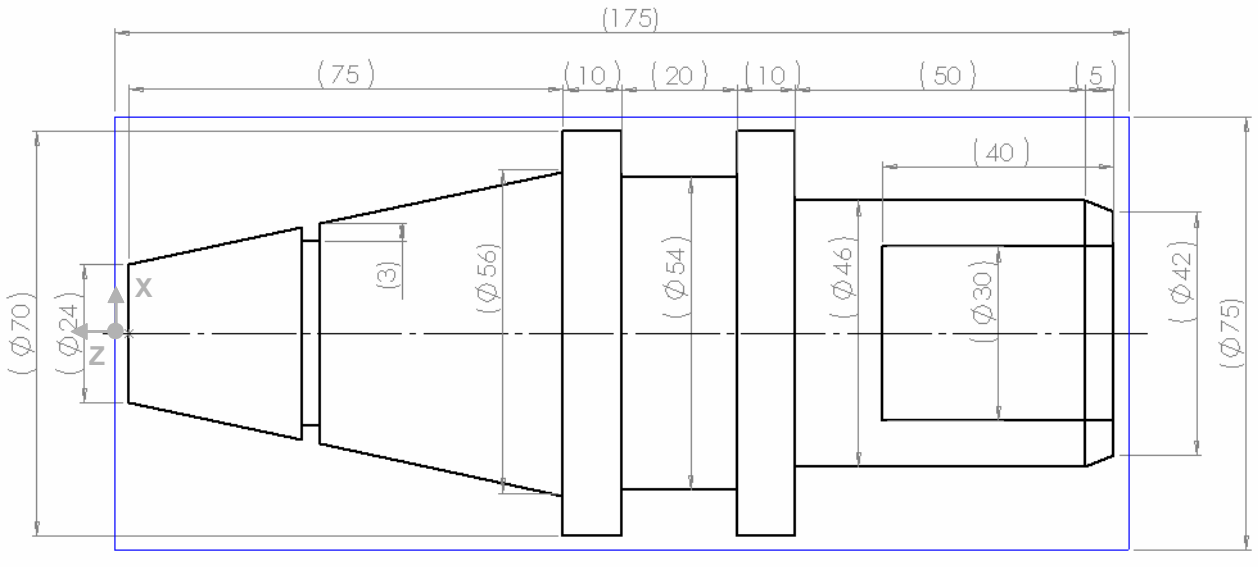
Annex E:
(informative)

Complex turning example

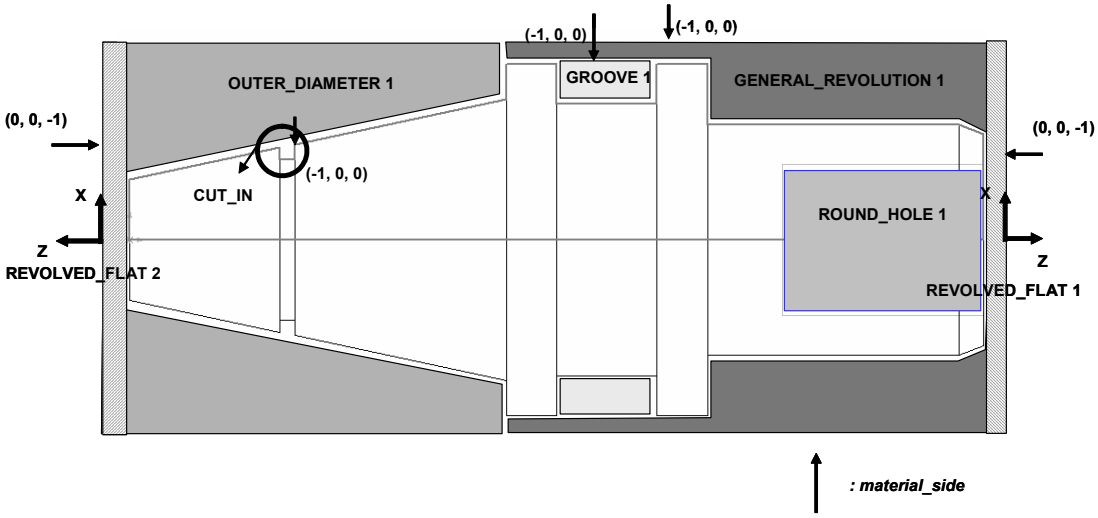
The following picture shows a more complex workpiece. The workpiece has 7 machining_features: 2 circular_face, 2 outer_diameter, 1 general_revolution, 1 groove, 1 cut_in. The final shape is machined by 2 setups; setup 1 is for right side features and setup 2 if for left side features. The measurement can be found in the picture. Below the listing of the NC program is printed.



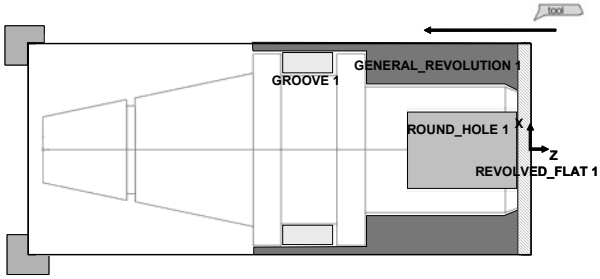
(a) Example part



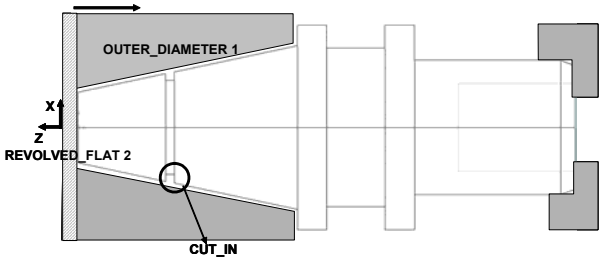
(b) Dimension of the example part



(c) Machining_features in example part



(d) Set-up 1



(e) Set-up 2

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('EXAMPLE OF NC PROGRAMME FOR TURNING: COMPLEX DESIGN.'),'1');
FILE_NAME('EXAMPLE1.STP',$,('ISO14649'),(''),'SUH','POSTECH', 'KOREA');
FILE_SCHEMA(('MACHINING_SCHEMA','TURNING_SCHEMA'));
ENDSEC;
DATA;

(* ************************************************** *)
(* ***** Workpiece definition ***** *)
#1=WORKPIECE('SIMPLE WORKPIECE',#2,0.01,$,$,$,());
#2=MATERIAL('ST-50','STEEL',(#3));
#3=PROPERTY_PARAMETER('E=200000N/M2');
#4=RIGHT_CIRCULAR_CYLINDER('WORKPIECE PIECE', #5,175.0, 35.0);
#5=AXIS1_PLACEMENT('WORKPIECE PIECE PLACEMENT',#6,#7);
#6=CARTESIAN_POINT('WORKPIECE PIECE: LOCATION ',(0.000,0.000,0.000));
#7=DIRECTION(' AXIS ',(0.000,0.000,1.000));
(* ************************************************** *)
(* ***** Manufacturing features ***** *)
#10=REVOLVED_FLAT('REVOLVED FLAT 1',#1,(#22,#23),#172,#176,21.0,#178);
#11=REVOLVED_FLAT('REVOLVED FLAT 2',#1,(#31,#32),#183#187,12.0,#189);
#12=GENERAL_REVOLUTION('GENERAL_REVOLUTION 1',#1,(#20,#21),#194,#198,21.0,#200);
#13=ROUND_HOLE('HOLE1 FLAT BOTTOM ',#1,(#26,#27,#28),#207,#215,#216,$,#217);
#14=OUTER_DIAMETER('OUTER_DIAMETER 1',#1,(#29,#30), #218,#222,#223,#224);
#15=GROOVE('GROOVE 1',#1,(#24,#25), #226,#230,35.0,#232);
#16=GROOVE('CUT_IN',#1,(#33),#236,#240,18.4,#242);
```

```
(* ************************************************** *)
(* ***** Turning operations ***** *)
#20=CONTOURING_ROUGH($,$,'ROUGH GENERALREVOLUTION1',30.000,$,#280,#61,#60,#130,#130,#131,0.5);
#21=CONTOURING_FINISH($,$,'FINISH GENERAL REVOLUTION 1',30.000,$,#280,#61,#60,#130,#130,#132,0.0);
#22=FACING_ROUGH($,$,'ROUGH CIRCULAR FACE 2',50.000,$,#280,#63,#60,#133,#134,#135,0.500);
#23=FACING_FINISH($,$,'FINISH CIRCULAR FACE 2',50.000,$,#280,#63,#60,#133,#134,#136,0.0);
#24=GROOVING_ROUGH($,$,'ROUGH GROOVE 1',30.000,$,#285,#65,#60,#137,#137,#138,$,0.500);
#25=GROOVING_FINISH($,$,'FINISH GROOVE 1',30.000,$,#285,#65,#60,#137,#137,#139,$,0.0);
#26= DRILLING($,$,'DRILL HOLE1',30.000,$,#289,#66,#67,$,$,$,$,$,#140);
#27= REAMING($,$,'REAM HOLE1',30.000,$,#293,#69,#67,$,$,$,$,$,#141,.T.,$,$);
#29=CONTOURING_ROUGH($,$,'ROUGH OUTER DIAMETER 1',30.000,$,#280,#61,#60,#130,#130,#131,0.5);
#30=CONTOURING_FINISH($,$,'FINISH OUTER_DIAMETER 1',30.000,$,#280,#61,#60,#130,#130,#132,0.0);
#31=FACING_ROUGH($,$,'ROUGH CIRCULAR FACE 1',50.000,$,#280,#63,#60,#133,#134,#135,0.500);
#32=FACING_FINISH($,$,'FINISH CIRCULAR FACE 1',50.000,$,#280,#63,#60,#133,#134,#136,0.0);
#33=CUTTING_IN($,$,'CUTTING IN 1',50.000,$,#297,#70,#60,#142,#142,#143,$,0.0);

(* ************************************************** *)
(* ***** Project ***** *)
#34=PROJECT('TURNING EXAMPLE 1',#35,(#1),$,$,$);
#35=WORKPLAN('MAIN WORKPLAN',(#36,#37),$,#52,$);
#36=WORKPLAN('WORK PLAN FOR SETUP1',(#38,#39,#40,#41,#42,#43,#44,#45),$,$,$);
#37=WORKPLAN('WORK PLAN FOR SETUP2',(#47,#48,#49,#50,#51),$,#54,$);
#38=MACHINING_WORKINGSTEP('WS ROUGH CIRCULAR_FACE 2',#56,#11,#22);
#39=MACHINING_WORKINGSTEP('WS FINISH CIRCULAR_FACE 2',#56,#11,#23);
#40=MACHINING_WORKINGSTEP('WS ROUGH GENERAL_REVOLUTION 1',#56,#12,#20);
#41=MACHINING_WORKINGSTEP('WS FINISH GENERAL_REVOLUTION 1',#56,#12,#21);
#42=MACHINING_WORKINGSTEP('WS ROUGH GROOVE 1',#56,#15,#24);
#43=MACHINING_WORKINGSTEP('WS FINISH GROOVE 1',#56,#15,#25);
#44=MACHINING_WORKINGSTEP('WS DRILLING',#56,#13,#26);
#45=MACHINING_WORKINGSTEP('WS REAMING',#56,#13,#27);
#47=MACHINING_WORKINGSTEP('WS ROUGH CIRCULAR_FACE 1',#56,#10,#30);
#48=MACHINING_WORKINGSTEP('WS FINISH CIRCULAR_FACE 1',#56,#10,#31);
#49=MACHINING_WORKINGSTEP('WS ROUGH OUTER_DIAMETER 2',#56,#14,#28);
#50=MACHINING_WORKINGSTEP('WS FINISH OUTER_DIAMETER 2',#56,#14,#29);
#51=MACHINING_WORKINGSTEP('WS FINISH CUT_IN 1',#56,#16,#32);

#52=SETUP('SETUP 1',#103,#56,(#53));
#53=WORKPIECE_SETUP(#1,#107,$,$,$);
#54=SETUP('SETUP 2',#111,#56,(#55));
#55=WORKPIECE_SETUP(#1,#115,$,$,$);
#56=PLANE('SECURITY PLANE',#119);

(* ************************************************** *)
(* ***** Functions / Technology ***** *)
#60=TURNING_MACHINE_FUNCTIONS(.T.,$,$,(),.F.,$,$,(),$,$,$);
#61=TURNING_TECHNOLOGY($,.TCP.,#62,0.300,.F.,.F.,.F.,$);
#62=CONST_SPINDLE_SPEED(500);
#63=TURNING_TECHNOLOGY($,.TCP.,#64,0.300,.F.,.F.,.F.,$);
#64=CONST_SPINDLE_SPEED(500);
#65=TURNING_TECHNOLOGY($,.TCP.,#66,0.300,.F.,.F.,.F.,$);
#66=CONST_SPINDLE_SPEED(200);
#67= MILLING_MACHINE_FUNCTIONS(.T.,$,$,.F.,$,(),.T.,$,$,());
#66= MILLING_TECHNOLOGY(0.030,.TCP.,$,16.000,$,.F.,.F.,.F.,$);
#69= MILLING_TECHNOLOGY(0.030,.TCP.,$,18.000,$,.F.,.F.,.F.,$);
#70=TURNING_TECHNOLOGY($,.TCP.,#71,0.300,.F.,.F.,.F.,$);
#71= CONST_SPINDLE_SPEED (100);

(* ************************************************** *)
(* ***** Strategies ***** *)
```

```
#130=PLUNGE_RAMP($,45.000);
#131=UNIDIRECTIONAL_TURNING($,$,(3.000),$,$,$,$,$,2.000,$,$);
#132=UNIDIRECTIONAL_TURNING($,$,(0.500),$,$,$,$,$,$,$,$);
#133=PLUNGE_RAMP($,30.000);
#134=PLUNGE_RAMP($,40.000);
#131=UNIDIRECTIONAL_TURNING($,.T.,(3.000),$,$,$,$,$,2.000,$,$);
#132=UNIDIRECTIONAL_TURNING($,.F.,(0.500),$,$,$,$,$,$,$,$);
#137=PLUNGE_TOOL_AXIS($);
#138=MULTISTEP_GROOVING_STRATEGY($,.T.,(3.000),$,$,5.0,3.0);
#139=CONTOUR_TURNING($,.F.,(0.500),$,$,$);
#140= DRILLING_TYPE_STRATEGY(75.000,50.000,5.000,50.000,75.000,40.000);
#141= DRILLING_TYPE_STRATEGY($,$,$,$,$,$);
#142=PLUNGE_TOOL_AXIS($);
#143= GROOVING_STRATEGY($,.T.,(1.0),$,$,5.000);

(* ************************************************** *)
(* ***** Placements / Lengths ***** *)
#103=AXIS2_PLACEMENT_3D('SETUP 1',#104,#105,#106);
#104=CARTESIAN_POINT('SETUP1: LOCATION ',(0.000,0.000,0.000));
#105=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#106=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#107=AXIS2_PLACEMENT_3D('WORKPIECE',#108,#109,#110);
#108=CARTESIAN_POINT('WORKPIECE1: LOCATION ',(0.000,0.000,0.000));
#109=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#110=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#111=AXIS2_PLACEMENT_3D('SETUP 2',#111,#112,#113);
#112=CARTESIAN_POINT('SETUP2: LOCATION ',(0.000,0.000,0.000));
#113=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#114=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#115=AXIS2_PLACEMENT_3D('WORKPIECE1',#116,#117,#118);
#116=CARTESIAN_POINT('WORKPIECE1: LOCATION ',(0.000,0.000,0.000));
#117=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#118=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#119=AXIS2_PLACEMENT_3D('SECURITY PLANE',#120,#121,#122);
#120=CARTESIAN_POINT('SECPLANE: LOCATION ',(0.000,0.000,50.000));
#121=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#122=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#172=AXIS2_PLACEMENT_3D('PLACEMENT END FACE 1',#173,#174,#175);
#173=CARTESIAN_POINT('END FACE 1: LOCATION ',(0.000,0.000,-2.500));
#174=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#175=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#176=DIRECTION(' MATERIAL_SIDE',(0.000,0.000,-1.000));
#178=LINEAR_PROFILE(' REVOLVED_FLAT_RADIUS',#179,21.000);
#179=AXIS2_PLACEMENT_3D('PLACEMENT END FACE 1',#180,#181,#182);
#180=CARTESIAN_POINT('END FACE 1: LOCATION ',(0.000,0.000,0.000));
#181=DIRECTION(' AXIS ',(0.000,0.000,1.000));
#182=DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));
#183=AXIS2_PLACEMENT_3D('PLACEMENT REVOLVED FLAT 2',#73,#74,#75);
#184=CARTESIAN_POINT(' REVOLVED FLAT2: LOCATION ',(0.000,0.000,-2.500));
#185=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#186=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#187=DIRECTION(' MATERIAL_SIDE',(0.000,0.000,-1.000));
#189=LINEAR_PROFILE(' REVOLVED_FLAT_RADIUS',#190,12.000);
#190=AXIS2_PLACEMENT_3D('LINEAR_PROFILE',#191,#192,#193);
#191=CARTESIAN_POINT('END FACE 1: LOCATION ',(0.000,0.000,0.000));
#192=DIRECTION(' AXIS ',(0.000,0.000,1.000));
#193=DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));
#194=AXIS2_PLACEMENT_3D('PLACEMENT GENERAL_REVOLUTION 1',#195,#196,#197);
#195=CARTESIAN_POINT(' GENERAL_REVOLUTION : LOCATION ',(0.000,0.000, -2.500));
```

```
#196=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#197=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#198=DIRECTION(' MATERIAL_SIDE',(-1.000,0.000,0.000));
#200=GENERAL_PROFILE($,#201);
#201=POLYLINE('',(#202,#203,#204,#205,#206));
#202=CARTESIAN_POINT('',(21.000,0.000, 0.000));
#203=CARTESIAN_POINT('',(23.000,0.000, 5.000));
#204=CARTESIAN_POINT('',(23.000,0.000, 55.000));
#205=CARTESIAN_POINT('',(35.000,0.000, 55.000));
#206=CARTESIAN_POINT('',(35.000,0.000, 95.000));
#207= AXIS2_PLACEMENT_3D('HOLE3',#208,#209,#210);
#208= DIRECTION('',(0.,0.,1.));
#209= DIRECTION('',(1.,0.,0.));
#210= CARTESIAN_POINT('',(0.,0.,0.));
#211=AXIS2_PLACEMENT_3D('',#212,#213,#214);
#212=CARTESIAN_POINT('',(0.000,0.000,-40.000));
#213=DIRECTION('',(0.000000,0.000000,1.000000));
#214=DIRECTION('',(1.000000,0.000000,0.000000));
#215=PLANE('',#211);
#216= TOLERANCED_LENGTH_MEASURE(15.0,#251);
#217= FLAT_HOLE_BOTTOM();
#218=AXIS2_PLACEMENT_3D('PLACEMENT OUTER_DIAMETER 1',#219,#220,#221);
#219=CARTESIAN_POINT(' OUTER_DIAMETER 2: LOCATION ',(0.000,0.000, -77.500));
#220=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#221=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#222=TOLERANCED_LENGTH_MEASURE(56.000,#251);
#223=TOLERANCED_LENGTH_MEASURE(75.000,#251);
#224=DIAMETER_TAPER(#225);
#225=TOLERANCED_LENGTH_MEASURE(24.000,#251);
#226=AXIS2_PLACEMENT_3D('PLACEMENT GROOVE 1',#227,#228,#229);
#227=CARTESIAN_POINT(' GROOVE 1: LOCATION ',(0.000,0.000, -67.500));
#228=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#229=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#230=DIRECTION(' MATERIAL_SIDE',(-1.000,0.000,0.000));
#232=SQUARE_U_PROFILE(#233,#234,0,#235,0);
#233=TOLERANCED_LENGTH_MEASURE(20.000,#251);
#234=TOLERANCED_LENGTH_MEASURE(0.000,#251);
#235=TOLERANCED_LENGTH_MEASURE(0.000,#251);
#236=AXIS2_PLACEMENT_3D('PLACEMENT CUT_IN 1',#237,#238,#239);
#237=CARTESIAN_POINT(' CUT_IN 1: LOCATION ',(0.000,0.000, -32.500));
#238=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#239=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));
#240=DIRECTION(' MATERIAL_SIDE',(-1.000,0.000,0.000));
#242=SQUARE_U_PROFILE(#243,#244,0.0,#245,0.0);
#243=TOLERANCED_LENGTH_MEASURE(3.000,#251);
#244=TOLERANCED_LENGTH_MEASURE(0.000,#251);
#245=TOLERANCED_LENGTH_MEASURE(0.000,#251);
#251= PLUS_MINUS_VALUE(0.100,0.100,3);

 (* ************************************************** *)
(* ***** Tools ***** *)
#280=TURNING_MACHINE_TOOL('',#281,(#283),120,40,$)
#281=GENERAL_TURNING_TOOL(#282,.LEFT.,40,60,.CW.);
#282=TOOL_DIMENSION($,$,$,$,25,5,7,3,5,0.5,$);
#283=CUTTING_COMPONENT(0.000000,$,$,$,$);
#285=TURNING_MACHINE_TOOL('',#286,(#288),120,40,$)
#286=GROOVING_TURNING_TOOL(#287,.LEFT.,40,60,.CW.,10.0, $ );
#287= TOOL_DIMENSION($,$,$,$,$,$,$,$,$,0.5,$);
#288=CUTTING_COMPONENT(40.000,$,$,$,$);
```

```
#289= MILLING_CUTTING_TOOL('SPIRAL_DRILL_15MM',#290,(#292),90.000,$,$);
#290= TWIST_DRILL(#290,2,.RIGHT.,.F.,0.840);
#291= MILLING_TOOL_DIMENSION(15.000,31.000,0.100,45.000,2.000,5.000,8.000);
#292= CUTTING_COMPONENT(90.000,$,$,$,$);
#293= MILLING_CUTTING_TOOL('REAMER_22MM',#294,(#296),100.000,$,$);
#294= REAMER(#295,6,$,.F.,$,$);
#295= MILLING_TOOL_DIMENSION(15.000,$,$,$,$,$,$);
#296= CUTTING_COMPONENT(100.000,$,$,$,$);
#297=TURNING_MACHINE_TOOL('',#298,(#300),$,$,$)
#298= USER_DEFINED_TURNING_TOOL(#299,.LEFT.,40,60,.CW.,10.0, $ );
#299= TOOL_DIMENSION($,$,$,$,$,$,$,$,$,$,$,$);
#300=CUTTING_COMPONENT(40.000,$,$,$,$);

ENDSEC;
END-ISO-10303-21;
```

Index