INTERNATIONAL STANDARD ISO 10303-50:2001

TECHNICAL CORRIGENDUM 2

Published 2014-07-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION

Industrial automation systems and integration — Product data representation and exchange —

Part 50:

Integrated generic resource:

Mathematical constructs

TECHNICAL CORRIGENDUM 2

Systèmes d' automatisation industrielle et intégration – Représentation et échange de données de produits

- Partie 50: Ressources génériques intégrées: Constructions mathematiques

RECTIFICATIF TECHNIQUE 2

Technical Corrigendum 2 to International Standard ISO 10303-50:2001 was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 4, *Industrial data.*

This Technical Corrigendum is intended to be used in conjunction with ISO 10303-50:2001/Cor.1:2010. Included SEDS reports: SEDS 1299. *Included Bugzilla reports: Bug* 979, *Bug 1109, Bug* 2574, *Bug 4114, Bug 5046, Bug 5053, Bug 5059*

ICS 25.040.40 **Ref. No. ISO 10303-50:2004/Cor.2:2014(E)**

Published in Switzerland

Introduction

This Technical Corrigendum applies to ISO 10303-50:2001 as modified by TC1.

The purpose of the modifications to the text of ISO 10303-50:2001 is to correct EXPRESS errors relating to incorrect data types in EXPRESS type, entity and function definitions, and to update the document identifiers in annex B.

Modifications to the text of ISO 10303-50:2001

Delete the current list of normative references and replace with the following undated references and move the reference to ISO//IEC 8824-1 to the bibligraphy:

ISO 10303-1, *Industrial automation systems and integration - Product data representation and exchange - Part 1 : Overview and fundamental principles.*

ISO 10303-11, *Industrial automation systems and integration - Product data representation and exchange - Part 11 : Description methods: The* EXPRESS *language reference manual.*

ISO 10303-41, *Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support.*

ISO 10303-42, *Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resource: Geometric and topological representation.*

ISO 13584-20, *Industrial automation systems and integration — Parts Library — Part 20: Logical resource: Logical model of expressions.*

Page 6, 4 Mathematical functions, EXPRESS specification
In the EXPRESS references to the schemas from ISO 13584-20 the wrong case is used in the schema names. Delete the line:
REFERENCE FROM ISO13584_generic_expressions_schema – ISO 13584-20
And replace with:


```
REFERENCE FROM iso13584_generic_expressions_schema -- ISO 13584-20
```

Delete the line:
REFERENCE FROM ISO13584_expressions_schema – ISO 13584-20
And replace with:

```
REFERENCE FROM iso13584_expressions_schema -- ISO 13584-20
```

In NOTE 1 change the schema names from ISO13584_generic_expressions_schema and ISO13584_-expressions_schema to:
iso13584_generic_expressions_schema and
iso13584_expressions_schema.

Page 34, 4.4.27, tuple_space
This type requires extensions in other parts of ISO 10303. Delete the current EXPRESS definition of the type **tuple_space** *and replace with:*

EXPRESS specification:

```
 *)
TYPE tuple_space = EXTENSIBLE GENERIC_ENTITY SELECT
   (product_space,
   extended_tuple_space);
END_TYPE;
 (*
```

Page 38, 4.5.5, complex_number_literal

The definition of complex_number_literal lacks a subtype to enable the definition of complex numbers by giving the values of modulus and argument. Immediately after clause 4.5.5 insert the following new definition as clause 4.5.6 and re-number the existing clauses 4.5.6 to 4.5.77 as 4.5.7 to 4.5.78.

4.5.6 complex_number_literal_polar

A **complex_number_literal_polar** is a type of **complex_number_literal** defined by the values of its modulus and argument.

EXPRESS specification:

```
*)
ENTITY complex_number_literal_polar
   SUBTYPE OF (complex_number_literal);
    modulus : REAL;
    argument : REAL;
  DERIVE
    SELF\complex_number_literal.real_part : REAL := modulus * cos(argument);
    SELF\complex_number_literal.imag_part : REAL := modulus * sin(argument);
  WHERE
    WR1: modulus >= 0;
    WR2: {0 <= argument <= 2*PI};
END_ENTITY;
(*
```


Attribute definitions:

modulus: The value of the modulus of the complex number. This is equal to the distance from the point representing the complex number to the origin of the complex plane.

argument: The value of the argument of the complex number. This is equal to the angle between the line joining the point representiong the complex number to the origin and the real axis.

Formal propositions:

WR1: The **modulus** shall not be negative.

WR2: The **argument** shall be between 0 and 2π.

Page 54, 4.5.32, extended_tuple_space

This entity contains an incorrect data type for the first attribute. Delete the current EXPRESS definition of the entity type **extended_tuple_space** *and replace with:*

EXPRESS specification:

```
*)
  ENTITY extended_tuple_space
    SUBTYPE OF (maths_space, generic_literal);
    base : tuple_space;
    extender : maths_space;
  WHERE
   WR1: expression_is_constant(base) AND  expression_is_constant(extender);
   WR2: no_cyclic_space_reference(SELF, []);
   WR3: extender <> the_empty_space;
 END_ENTITY;
(*
```


Remove the description given for the first attribute **base** *and replace with:*

The **tuple_space** describing the common initial component spaces of all the ordered tuples belonging to this tuple space. When there are no ocommon initial components, the value of **base** will be the zero-tuple space.

Page 73, 4.5.52, linearized_table_function

This entity contains an incomplete reference path in WR2. Delete the current EXPRESS definition of the entity type **linearized_table_function** *and replace with:*

EXPRESS specification:

```
 *)
   ENTITY linearized_table_function
     SUPERTYPE OF (ONEOF (standard_table_function,
                          regular_table_function,
                          triangular_matrix,
                          symmetric_matrix,
                          banded_matrix))
     SUBTYPE OF (explicit_table_function, unary_generic_expression);
     SELF\unary_generic_expression.operand : maths_function;
     first  : integer;
   DERIVE
     source : maths_function := SELF\unary_generic_expression.operand;
   WHERE
     WR1: function_is_1d_array(source);
     WR2: member_of(first, source\maths_function.domain);
   END_ENTITY;
 (*
```


Page 79, 4.5.57, symmetric_matrix

The entity **symmetric_matrix** *contains incorrect data types in WR3 and WR4. Delete the current EXPRESS definition of the entity* **symmetric_matrix** *and replace with:*

EXPRESS specification:

```
 *)
 ENTITY symmetric_matrix
  SUBTYPE OF (linearized_table_function);
  symmetry : symmetry_type;
  triangle : lower_upper;
  order    : ordering_type;
WHERE
  WR1: SIZEOF (SELF\explicit_table_function.shape) = 2;
  WR2: SELF\explicit_table_function.shape[1] =
       SELF\explicit_table_function.shape[2];
  WR3: NOT (symmetry = symmetry_type.skew) OR (
       (space_dimension(SELF\linearized_table_function.source.range) = 1) AND
        subspace_of_es(factor1(SELF\linearized_table_function.source.range),
        es_numbers));
  WR4: NOT ((symmetry = symmetry_type.hermitian) OR
              (symmetry = symmetry_type.skew_hermitian)) OR (
       (space_dimension(SELF\linearized_table_function.source.range) = 1) AND
        subspace_of_es(factor1(SELF\linearized_table_function.source.range),
        es_complex_numbers));
END_ENTITY;
 (*
```


Page 141, 4.6.31 derive_function_range

The function **derive_function_range** *contains EXPRESS errors of incompatible types for the assignment statements with function calls to the* **make_uniform_product_space** *funtion. A new local variable is introduced to correct this problem. Remove completely the existing EXPRESS definition and replace with:*

EXPRESS specification:

```
  *)
FUNCTION derive_function_range(func : maths_function) : tuple_space;
  LOCAL
    typenames : SET OF STRING := stripped_typeof(func);
    tspace : tuple_space := make_listed_product_space ([]);
    m, n : nonnegative_integer := 0;
    temp : INTEGER := 0;
  END_LOCAL;
  IF 'FINITE_FUNCTION' IN typenames THEN
    RETURN (derive_finite_function_range (func\finite_function.pairs));
  END_IF;
  IF 'CONSTANT_FUNCTION' IN typenames THEN
    RETURN (one_tuples_of (make_finite_space ([func\constant_function.sole_output])));
  END_IF;
  IF 'SELECTOR_FUNCTION' IN typenames THEN
    tspace := func.domain;
    IF (space_dimension(tspace) = 1) AND ((schema_prefix + 'TUPLE_SPACE') IN
      TYPEOF (tspace)) THEN
      tspace := factor1 (tspace);
    END_IF;
    RETURN (one_tuples_of (factor_space (tspace, func\selector_function.selector)));
  END_IF;
  IF 'ELEMENTARY_FUNCTION' IN typenames THEN
    RETURN (derive_elementary_function_range (func\elementary_function.func_id));
  END_IF;
  IF 'RESTRICTION_FUNCTION' IN typenames THEN
    RETURN (one_tuples_of (func\restriction_function.operand));
  END_IF;
  IF 'REPACKAGING_FUNCTION' IN typenames THEN
    tspace := func\repackaging_function.operand.range;
    IF func\repackaging_function.output_repack = ro_wrap_as_tuple THEN
      tspace := one_tuples_of (tspace);
    END_IF;
    IF func\repackaging_function.output_repack = ro_unwrap_tuple THEN
      tspace := factor1 (tspace);
    END_IF;
    IF func\repackaging_function.selected_output > 0 THEN
      tspace := one_tuples_of (factor_space (tspace,
        func\repackaging_function.selected_output));
    END_IF;
    RETURN (tspace);
```



```
  END_IF;
  IF 'REINDEXED_ARRAY_FUNCTION' IN typenames THEN
    RETURN (func\unary_generic_expression.operand\maths_function.range);
  END_IF;
  IF 'SERIES_COMPOSED_FUNCTION' IN typenames THEN
    RETURN (func\series_composed_function.operands[SIZEOF
      (func\series_composed_function.operands)].range);
  END_IF;
  IF 'PARALLEL_COMPOSED_FUNCTION' IN typenames THEN
    RETURN (func\parallel_composed_function.final_function.range);
  END_IF;
  IF 'EXPLICIT_TABLE_FUNCTION' IN typenames THEN
    IF 'LISTED_REAL_DATA' IN typenames THEN
      RETURN (one_tuples_of (the_reals));
    END_IF;
    IF 'LISTED_INTEGER_DATA' IN typenames THEN
      RETURN (one_tuples_of (the_integers));
    END_IF;
    IF 'LISTED_LOGICAL_DATA' IN typenames THEN
      RETURN (one_tuples_of (the_logicals));
    END_IF;
    IF 'LISTED_STRING_DATA' IN typenames THEN
      RETURN (one_tuples_of (the_strings));
    END_IF;
    IF 'LISTED_COMPLEX_NUMBER_DATA' IN typenames THEN
      RETURN (one_tuples_of (the_complex_numbers));
    END_IF;
    IF 'LISTED_DATA' IN typenames THEN
      RETURN (one_tuples_of (func\listed_data.value_range));
    END_IF;
    IF 'EXTERNALLY_LISTED_DATA' IN typenames THEN
      RETURN (one_tuples_of (func\externally_listed_data.value_range));
    END_IF;
    IF 'LINEARIZED_TABLE_FUNCTION' IN typenames THEN
      RETURN (func\linearized_table_function.source.range);
    END_IF;
    IF 'BASIC_SPARSE_MATRIX' IN typenames THEN
      RETURN (func\basic_sparse_matrix.val.range);
    END_IF;
    -- Unreachable, as no other subtypes of explicit_table_function are permissible
    -- without first modifying this function to account for them.
    RETURN (?);
  END_IF;
  IF 'HOMOGENEOUS_LINEAR_FUNCTION' IN typenames THEN
    RETURN (one_tuples_of (make_uniform_product_space
      (factor1 (func\homogeneous_linear_function.mat.range),
      func\homogeneous_linear_function.mat\explicit_table_function.shape
      [3 - func\homogeneous_linear_function.sum_index])));
  END_IF;
  IF 'GENERAL_LINEAR_FUNCTION' IN typenames THEN
    RETURN (one_tuples_of (make_uniform_product_space
      (factor1 (func\general_linear_function.mat.range),
```



```
    func\general_linear_function.mat\explicit_table_function.shape
    [3 - func\general_linear_function.sum_index])));
END_IF;
IF 'B_SPLINE_BASIS' IN typenames THEN
  RETURN (one_tuples_of (make_uniform_product_space (the_reals,
    func\b_spline_basis.num_basis)));
END_IF;
IF 'B_SPLINE_FUNCTION' IN typenames THEN
  tspace := factor1 (func\b_spline_function.coef.domain);
  m := SIZEOF (func\b_spline_function.basis);
  n := space_dimension (tspace);
  IF m = n THEN
    RETURN (one_tuples_of (the_reals));
  END_IF;
  IF m = n - 1 THEN
    RETURN (one_tuples_of (make_uniform_product_space (the_reals,
      factor_space (tspace, n)\finite_integer_interval.size)));
  END_IF;
  tspace := extract_factors (tspace, m+1, n);
  RETURN (one_tuples_of (make_function_space (sc_equal, tspace, sc_subspace,
    number_superspace_of (func\b_spline_function.coef.range))));
END_IF;
IF 'RATIONALIZE_FUNCTION' IN typenames THEN
  tspace := factor1 (func\rationalize_function.fun.range);
  n := space_dimension (tspace);
  RETURN (one_tuples_of (make_uniform_product_space (number_superspace_of (
    factor1 (tspace)), n-1)));
END_IF;
IF 'PARTIAL_DERIVATIVE_FUNCTION' IN typenames THEN
  RETURN (drop_numeric_constraints (
    func\partial_derivative_function.derivand.range));
END_IF;
IF 'DEFINITE_INTEGRAL_FUNCTION' IN typenames THEN
  RETURN (drop_numeric_constraints (
    func\definite_integral_function.integrand.range));
END_IF;
IF 'ABSTRACTED_EXPRESSION_FUNCTION' IN typenames THEN
  RETURN (one_tuples_of(values_space_of(func\abstracted_expression_function.expr)));
END_IF;
IF 'EXPRESSION_DENOTED_FUNCTION' IN typenames THEN
  RETURN (values_space_of (func\expression_denoted_function.expr)\function_space.
    range_argument);
END_IF;
IF 'IMPORTED_POINT_FUNCTION' IN typenames THEN
    temp := dimension_of (func\imported_point_function.geometry);
  RETURN (one_tuples_of (make_uniform_product_space (the_reals, temp)));
END_IF;
IF 'IMPORTED_CURVE_FUNCTION' IN typenames THEN
    temp := dimension_of (func\imported_curve_function.geometry);
  RETURN (one_tuples_of (make_uniform_product_space (the_reals, temp)));
END_IF;
IF 'IMPORTED_SURFACE_FUNCTION' IN typenames THEN
```

```
      temp := dimension_of (func\imported_surface_function.geometry);
    RETURN (one_tuples_of (make_uniform_product_space (the_reals, temp)));
  END_IF;
  IF 'IMPORTED_VOLUME_FUNCTION' IN typenames THEN
      temp := dimension_of (func\imported_volume_function.geometry);
    RETURN (one_tuples_of (make_uniform_product_space (the_reals, temp)));
  END_IF;
  IF 'APPLICATION_DEFINED_FUNCTION' IN typenames THEN
    RETURN (func\application_defined_function.explicit_range);
  END_IF;
  -- Unreachable, as no other subtypes of maths_function are permissible without
  -- first modifying this function to account for them.
  RETURN (?);
END_FUNCTION;  -- derive_function_range
 (*
```

Page 190, 4.6.76 make_extended_tuple_space

The function **make_extended_tuple_space** *contains an EXPRESS error giving the wrong SELECT TYPE for the first argument* **base** *and errors in the argument descriptions. Remove completely the existing EXPRESS definition and replace with:*

EXPRESS specification:


```
  *)
  FUNCTION make_extended_tuple_space(base : tuple_space;
  extender : maths_space) : extended_tuple_space;
  RETURN (extended_tuple_space (base, extender)
  || maths_space ()
    || generic_expression()
     || generic_literal ()
        || simple_generic_expression() );
END_FUNCTION; -- make_extended_tuple_space
  (*
```

Remove completely the Agument definitions and replace with :

Argument definitions:

base: The **tuple_space** to be extended.

extender: The source of the extension.

return: (output) The constructed complex entity instance of **extended_tuple_space**.

Page 254, 4.6.144 stripped_typeof

The function **stripped_typeof** *contains EXPRESS errors of incompatible types for the additions to* **stypes**

