INTERNATIONAL STANDARD

ISO/IEC/IEEE 21451-4

First edition
2010-05-15

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Information technology — Smart transducer interface for sensors and actuators —

Part 4:

Mixed-mode communication protocols and Transducer Electronic Data Sheet (TEDS) formats

Technologies de l'information — Interface de transducteurs intelligente pour capteurs et actuateurs —

Partie 4: Protocoles de communication en mode mixte et formats des feuilles de données électroniques du transducteur (TEDS)

Reference number
ISO/IEC/IEEE 21451-4:2010(E)

© IEEE 2004

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat, the IEC Central Office and IEEE do not accept any liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies and IEEE members. In the unlikely event that a problem relating to it is found, please inform the ISO Central Secretariat or IEEE at the address given below.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

COPYRIGHT PROTECTED DOCUMENT

© IEEE 2004

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO or IEEE at the respective address below.

ISO copyright office
Case postale 56 CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York NY 10016-5997, USA
E-mail stds.ipr@ieee.org
Web www.ieee.org

ISO version published 2010

Published in Switzerland

© IEEE 2004 – All rights reserved

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

The main task of ISO/IEC JTC 1 is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is called to the possibility that implementation of this standard may require the use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISO/IEEE is not responsible for identifying essential patents or patent claims for which a license may be required, for conducting inquiries into the legal validity or scope of patents or patent claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance or a Patent Statement and Licensing Declaration Form, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from ISO or the IEEE Standards Association.

ISO/IEC/IEEE 21451-4 was prepared by the Technical Committee on Sensor Technology of the IEEE Instrumentation and Measurement Society of the IEEE (as IEEE Std 1451.4-2004). It was adopted by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*, in parallel with its approval by the ISO/IEC national bodies, under the “fast-track procedure” defined in the Partner Standards Development Organization cooperation agreement between ISO and IEEE. IEEE is responsible for the maintenance of this document with participation and input from ISO/IEC national bodies.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

(blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

IEEE Std 1451.4™-2004

IEEE Standards

1451.4™

IEEE Standard for A Smart Transducer Interface for Sensors and Actuators—Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

IEEE Instrumentation and Measurement Society

Sponsored by the
Technical Committee on Sensor Technology TC-9

IEEE

3 Park Avenue, New York, NY 10016-5997, USA

15 December 2004

Print: SH95225
PDF: SS95225

© IEEE 2004 – All rights reserved

(blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Recognized as an
American National Standard (ANSI)

IEEE Std 1451.4™-2004

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Sponsor

Technical Committee on Sensor Technology
of the
IEEE Instrument and Measurement Society

Approved 25 August 2004
American National Standards Institute

Approved 25 March 2004
IEEE-SA Standards Board

Abstract: This standard defines the protocol and interface that allows analog transducers to communicate digital information with an IEEE 1451 object. It also defines the format of the Transducer TEDS. The Transducer TEDS is based on the IEEE 1451.2™ TEDS. The standard does not specify the transducer design, signal conditioning, or the specific use of the TEDS.
Keywords: appended TEDS, basic TEDS, device configuration file, family code, IEEE 1451.4 interface, IEEE 1451.4 transducer, mixed-mode Interface (MMI), mixed-mode transducer (MMXD-CR), network capable application processor (NCAP), plug-and-play, smart transducer, TEDS, template, template ID, transducer electronic data sheet (TEDS), template description language, transparent protocol, template description language, tbom schema, transducer block

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2004 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 14 December 2004. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

1-Wire & MicroLAN are trademarks of Maxim-Dallas Semiconductor Corporation.

Print: ISBN 0-7381-4007-4 SH95225
PDF: ISBN 0-7381-4008-2 SS95225

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331USA

NOTE—Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Introduction

[This introduction is not part of IEEE Std 1451.4-2004, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats.]

The main objectives of this standard are to

- Enable plug-and-play at the transducer level by providing a common IEEE 1451.4 Transducer communication interface compatible with legacy transducers.
- Enable and simplify the creation of smart transducers.
- Facilitate the support of multiple networks.
- Simplify the setup and maintenance of instrumentation systems.
- Provide a bridge between the legacy instrumentation systems and the smart mixed-mode transducers.
- Enable implementation of smart transducers with minimal use of memory.

There was previously no defined common digital communication interface standard between mixed-mode transducers and network capable application processors (NCAPs). Each transducer manufacturer defined its own interface. Consequently, transducer manufacturers could not support all of the control networks for which their products might be suitable. A universally accepted mixed-mode transducer interface standard will facilitate the development of compliant smart sensors and actuators and could lead to lower development costs. This common interface allows the transducer manufacturers to support multiple control networks easily and helps to preserve the user's investment if it becomes necessary to migrate to a different network standard. In addition, this standard will make systems much easier to implement and use.

This standard simplifies the development of smart mixed-mode transducers by defining hardware and software blocks that are independent of specific control networks. The standard describes the following:

- An IEEE 1451.4 Transducer containing a Mixed-Mode Interface (MMI) and a transducer electronic data sheet (TEDS).
- The MMI, which is a master-slave, multidrop, serial connection. It requires a master device to initiate each transaction with each slave or node according to a defined digital communication protocol. The MMI may contain circuitry to detect and report a hotswap of transducers. The MMI may use either separate digital and analog connections, or two wires for power supply and time-shared analog signal and digital TEDS data. The MMI is used to access the TEDS.
- The TEDS, which is fixed and dynamic data, contained in one or more memory nodes on the MMI.
- A template, which is a software object describing the data structure of TEDS. It is implemented in the Template Description Language and resides in the Transducer Block.
- The Template Description Language, which is a scripted and tagged language providing a standard method to describe the functionality of IEEE 1451.4 Transducer.
- A Transducer Block, which is a software object describing the IEEE 1451.4 Transducer. It resides in the NCAP, which is the master device (e.g., an instrument or data acquisition system). The Transducer Block is used to access, decode, and encode TEDS using the TDL.

Furthermore, the Working Group has defined a set of TEDS templates for various transducers to facilitate the creation of sensor systems containing plug-and-play smart transducers.

The IEEE 1451.4 Transducer provides a self-describing capability, via the TEDS. The TEDS contains fields that describe the identity, type, operation, and attributes of the transducer. The IEEE 1451.4 Transducer is a sensor or actuator with one or more addressable devices, referred to as nodes, on a 2-conductor digital bus. The TEDS is required to be either physically, or virtually, associated with the IEEE 1451.4 Transducer. The resulting hardware partition encapsulates the measurement aspects inside the IEEE 1451.4 Transducer, while the application related aspects may reside either in the NCAP or in the TEDS.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

The IEEE 1451.4 Transducer is a sensor or actuator with one, or more, addressable devices, which herein will be referred to as nodes, containing TEDS.

A digital communication protocol is defined for transactions on the bus. The transactions are as follows:

— Read (Read TEDS)

— Write (Write TEDS)

— Configure (Set Gain, Change Mode, Set Filter)

— Check status (Read Settings)

The IEEE 1451.4 MMI may be used for control networks and data acquisition in a variety of applications, such as portable instruments and data acquisition plug-in cards for PCs.

The Transducer Block object located in the NCAP describes the behavior of the IEEE 1451.4 Transducer. It interprets TEDS data according to the data structure defined in templates. Further processing of the data may take place both in the NCAP and in other processors in larger systems. The NCAP includes an IEEE 1451.1 object model with an IEEE 1451.4 Transducer Block.

The standard does not constrain competitive differentiation in areas of quality, feature set, and cost, and at the same time, offers the opportunity to design to a common interface, which can be used in a wide variety of applications.

Acknowledgements

The working group would like to acknowledge the following individuals who made special contributions to the development of this standard:

Steven Chen, Former Chair, who proposed the mixed-mode transducer interface concept and initiated the development of the standard.

Jørgen Bække, Former Vice Chair, who was instrumental in getting the 2-conductor bus interface and transducer description language concept accepted by the group.

The IEEE has defined a common digital communication interface standard for mixed-mode transducers utilizing a single wire serial bus technology developed by Maxim/Dallas Semiconductor Corporation.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Notice to users

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

© IEEE 2004 – All rights reserved

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention. A patent holder or patent applicant has filed a statement of assurance that it will grant licenses under these rights without compensation or under reasonable rates and nondiscriminatory, reasonable terms and conditions to applicants desiring to obtain such licenses. The IEEE makes no representation as to the reasonableness of rates, terms, and conditions of the license agreements offered by patent holders or patent applicants. Further information may be obtained from the IEEE Standards Department.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Contents

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

1. Overview

This standard is divided into nine clauses. Clause 1 provides the scope of this standard. Clause 2 lists references to other standards that are useful in applying this standard. Clause 3 provides definitions that are either not found in other standards or have been modified for use with this standard. Clause 4 describes the IEEE 1451.4™ Interface and the IEEE 1451.4 Transducer and levels of compliance with this standard. Clause 5 describes the TEDS. Clause 6 describes the usage of the template structure. Clause 7 describes the syntax and semantics of the language used in the templates. Clause 8 describes the Mixed-Mode Transducer Interface (MMI) that ensures the robust transfer of an analog transducer signal and the digital TEDS data. Clause 9 describes the Transducer Block, which is the collective logic required to manage the transducer bus and all external components.

This standard also contains several annexes. Annex A lists the IEEE templates. Annex B lists the definitions of properties used in templates. Annex C contains the Template Description Language (TDL) formal grammar. Annex D gives a template file checksum example. Annex E gives information about the Family Code in the Unique Registration Number (URN). Annex F gives the Device Configuration File format needed for a parser to be able to understand Family Codes and act accordingly. Annex G contains an XML device description schema to be used to add support for new devices. Annex G contains information about the transparent protocol facilitating the communication with 2-conductor bus devices. Annex H describes the procedure for adding new IEEE templates and TDL items. Annex I contains information about the early draft of the standard IEEE P1451.4, version 0.9. Annex J lists the IEEE 1451.4 Manufacturer IDs. Annex K gives the IEEE 1451.4 Transducer Block Object Model (TBOM) schema. Annex L defines the Transducer Block adapter class that shall be used to represent transducers adhering to IEEE 1451.4 within an IEEE 1451.1™ environment.[1] Annex M is the bibliography.

[1]Information on references can be found in Clause 2.

© IEEE 2004 – All rights reserved

1.1 Scope

This standard defines the protocol and interface that allows analog transducers to communicate digital information with an IEEE 1451 object. It also defines the format of the Transducer TEDS. The Transducer TEDS is based on the IEEE 1451.2™ TEDS. The standard does not specify the transducer design, signal conditioning, or the specific use of the TEDS.

1.2 Purpose

An independent and openly defined standard for MMI and TEDS serves the following purposes:

— Provide interoperability, which enables plug-and-play capability

— Simplify the implementation of mixed-mode smart transducer systems

— Accelerate the emergence and acceptance of the MMI and TEDS

1.3 Conformance, shall, should, may, and can

Several keywords are used to differentiate among various levels of requirements, as follows:

The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (shall equals *is required to*).

The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

2. References

This standard shall be used in conjunction with the following standards publications. When the following standards are superseded by an approved revision, the revision shall apply.

ANSI X3.4-2000, US-ASCII. Coded Character Sets—7-Bit American Standard Code for Information Interchange.[2]

IEEE Std 754™-1985 (Reaff 1990), IEEE Standard for Binary Floating-Point Arithmetic.[3,4]

IEEE Std 1451.1-1999, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Network Capable Application Processor (NCAP) Information Model.

[2]ANSI publications are available from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[3]The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

[4]IEEE publications are available from the Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

IEEE Std 1451.2-1997, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats.

ISO/IEC 10646-1:2000, Information technology—Universal Multiple-Octet Coded Character Set (UCS)—Part 1: Architecture and Basic Multilingual Plane.[5]

Le Système international d'unités (SI), The International System of Units (SI), 7th Edition (1998), with Supplement 2000.[6]

3. Definitions and abbreviations

This clause provides definitions that are either not found in other standards or have been modified for use with this standard.

3.1 Terms

This subclause contains key terms as they are used in this standard.

3.1.1 active mode: A condition of a transducer where its analog output is enabled.

3.1.2 actuator: A transducer that accepts an electrical signal and converts it into a physical, chemical, or biological action.

3.1.3 address: A character or group of characters that identifies a register, a particular part of storage, or some other data source or destination.

3.1.4 Appended TEDS: An Appended TEDS is a TEDS not located on the transducer. It provides a mechanism for defining properties without memory size constraints.

3.1.5 Basic TEDS: TEDS data that follows the defined IEEE 1451.4 format of manufacturer, model, version letter, version number, and serial number.

3.1.6 byte: A group of eight bits, also known as an octet.

3.1.7 calibration: The determination of the data to reside in the TEDS and to be used for correction.

3.1.8 channel: A single flow path for digital data or an analog signal, usually in distinction from other parallel paths.

3.1.9 correction: The evaluation of a function using information from the TEDS together with data from the same channel.

3.1.10 data sheet: A set of information on a device that defines the parameters of operation and conditions of usage (usually produced by the device's manufacturer).

[5]ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO/IEC publications are also available in the United States from Global Engineering Documents, 15 Inverness Way East, Englewood, CO 80112, USA (http://global.ihs.com/). Electronic copies are available in the United States from the American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[6]This publication is available from the Bureau International des Poids et Mesures, Pavillion de Breteuil F, 92312 Sèvres, Cedex, France (http://www1.bipm.org/en/publications/brochure/).

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

3.1.11 data structure: A group of digital data fields organized in some logical order for some specific purpose. A two-dimensional paper version of a data structure is an empty fill-in-the-blanks form or an empty tabular chart with organized column and row headings. A data structure is the template by which data is stored in computer memory.

3.1.12 digital interface: A set of wires and a protocol for transferring information by binary means only.

3.1.13 electronic data sheet: A data sheet stored in some form of electrically readable memory (as opposed to a piece of paper).

3.1.14 enumeration: The listing of the meaning associated with each binary numeric value possible in a data field's storage. Binary numbers are usually expressed in decimal terms for human convenience. Not all possible numeric values need to have a specific meaning. Values without meaning are declared to be unused or reserved for future use. Enumeration is the process of declaring the encoding of human interpretable information in a manner convenient for digital electronic machine storage and interchange. The subclause that defines each TEDS data field that is of data type enumeration shall contain a table that defines the meaning of the data field for each binary number possible. The meanings encoded in each data field shall be specific and unique to that data field and only that data field. The value becomes meaningless if not associated with the data field and its defining table.

3.1.15 Family Code: The part of the URN for a device, which is used to identify the device functions and specific communication commands for the node.

3.1.16 hotswap: The act of connecting or disconnecting an IEEE 1451.4 Transducer and a higher level object such as an NCAP without first turning off the power that the higher level object supplies to the IEEE 1451.4 Transducer over the MMI.

3.1.17 IEEE 1451.4 Interface: A mixed-mode interface (MMI) permitting analog and digital signal transmission according to IEEE 1451.4 specifications.

3.1.18 IEEE 1451.4 Transducer: An entity containing the TEDS, at least one node and one transducer that meet IEEE 1451.4 specifications.

3.1.19 least significant bit (lsb): The bit in the binary notation of a number that is the coefficient of the lowest exponent possible.

3.1.20 least significant byte (LSByte): The byte in a multibyte word that represents the least significant values.

3.1.21 metadata: Literally described as data about data. Metadata provides a higher level of abstraction about data.

3.1.22 Mixed-Mode Interface or Mixed-Mode Transducer Interface (MMI): The same as IEEE 1451.4 Interface.

3.1.23 Mixed-Mode Transducer (MMXdcr): The same as IEEE 1451.4 Transducer.

3.1.24 most significant bit (msb): The bit in the binary notation of a number that is the coefficient of the highest exponent possible.

3.1.25 most significant byte (MSByte): The byte in a multibyte word that represents the most significant values.

3.1.26 node: Addressable device.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

3.1.27 Node-List: If more than one node is included in an IEEE 1451.4 Transducer, one of the nodes shall have a memory block that holds the Node-List, which is a list of the IDs of the other nodes included in the same transducer.

3.1.28 negative logic: An electronic logic system where the voltage representing one, active or true, has a more negative value that the voltage representing zero, inactive or false. Also known as negative-true logic. Normally used in electronic and computing data and communications switching systems for noise immunity reasons.

3.1.29 network capable application processor (NCAP): A device between the MMI and the network that performs network communications, MMI communications, and data conversion functions.

3.1.30 passive mode: A condition of a transducer where its analog output is disabled.

3.1.31 plug-and-play: To retrieve information from the IEEE 1451.4 Transducer and thereby enable automatic configuration.

3.1.32 positive logic: An electronic logic system where the voltage representing one, active or true, has a more positive value that the voltage representing zero, inactive or false. Normally used in industrial and commercial control switching systems for safety reasons.

3.1.33 sensor: A transducer that converts a physical, biological, or chemical parameter into an electrical signal.

3.1.34 signal conditioning: Sensor signal processing involving operations such as amplification, compensation, filtering, and normalization.

3.1.35 smart transducer: A transducer (actuator or sensor) with an associated TEDS and a circuit that allows both the signal and the TEDS data to be transferred through the MMI under defined conditions.

3.1.36 TEDS service: A TEDS service is an interface specification grouping operations that provide support for locating Appended TEDS, encoding TEDS, decoding TEDS, and a template database.

3.1.37 template: A software object describing the data structure of TEDS. It is implemented in the TDL and is available to the transducer block.

3.1.38 Template ID: An identification number at the beginning of each template that identifies the template.

3.1.39 transducer: A device converting energy from one domain into another. The device may either be a sensor or an actuator.

3.1.40 Transducer Electronic Data Sheet (TEDS): A data sheet describing a transducer stored in some form of electrically readable memory.

3.1.41 transducer interface: The physical connection by which a transducer communicates with the control or data systems that it is a member of, including the physical connector, the signal wires used, and the rules by which information is passed across the connection.

3.1.42 transfer: The act or process of moving a block of information from one digital device to another.

3.1.43 Unique Registration Number (URN): The URN is an electronically stored ID number, including an 8-bit Family Code, 48-bit unique serial number, and 8-bit cyclic redundancy check (CRC) byte.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

3.2 Abbreviations

This subclause contains abbreviations of key terms as they are used in this standard.

#	number
AC	alternating current
ADC	analog-to-digital converter
ASCII	American Standard Code for Information Interchange
COI	Common Object Interface
CRC	cyclic redundancy check
DAC	digital-to-analog converter
DC	direct current
EEPROM	electrically erasable programmable read-only memory
ID	identification
lsb	least significant bit
LSByte	least significant byte
MMI	Mixed-Mode Interface, identical to Mixed-Mode Transducer Interface
MMXdcr	Mixed-Mode Transducer
msb	most significant bit
MSByte	most significant byte
NCAP	network capable application processor
OTP	one time programmable
r/w	read/write
ROM	read-only memory
SI	Le Système International d'unités (SI), The International System of Units (SI)
STIM	smart transducer interface module
T-block	Transducer Block
TBOM	Transducer Block Object Model
TDL	Template Description Language

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

TEDS Transducer Electronic Data Sheet

UML Unified Modeling Language

Xdcr transducer

4. IEEE 1451.4 Transducer

This clause describes the IEEE 1451.4 Interface and the IEEE 1451.4 Transducer and levels of compliance with IEEE Std 1451.4-2004.

4.1 Foundation

IEEE Std 1451.4-2004 defines TEDS and MMI. The context for the Mixed-Mode Transducer and Interface is shown in Figure 1.

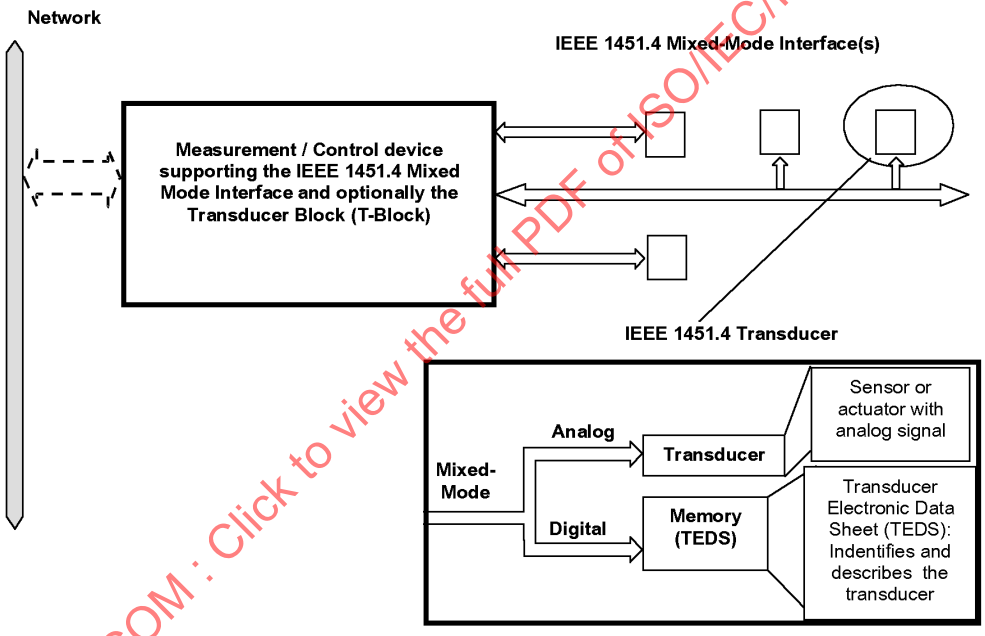


Figure 1—Context for the IEEE 1451.4 Transducer and Interface

The measurement/control device can be or contain an NCAP conforming with IEEE Std 1451.1-1999 and giving direct network access to the transducers.

An IEEE 1451.4 Transducer may be used to sense or control multiple physical phenomena. Each phenomenon sensed or controlled shall be associated with a node. If more than one node is included in an IEEE 1451.4 Transducer, one of the nodes shall have a memory block that holds the Node-List. The Node-List contains the IDs of the other nodes within the IEEE 1451.4 Transducer.

An IEEE 1451.4 Transducer has no more than one Node-List. If there is only one node inside an IEEE 1451.4 Transducer, there is no Node-List. Each IEEE 1451.4 Transducer is connected to an IEEE 1451.4 MMI. Each IEEE 1451.4 MMI can have several IEEE 1451.4 Transducers attached provided they are each capable of being in a passive mode.

To communicate with the nodes inside an IEEE 1451.4 Transducer with shared wires, the IEEE 1451.4 Transducer is switched into Digital Mode. In Class 1 (see 8.1.2), this is achieved by reversing the polarity of

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

the power supply. While in Digital Mode, the shared lines are used for the digital communication described in this standard and other uses are inhibited. In Class 2 (see 8.1.2), where separate wires are used for the IEEE 1451.4 communication, any other communication may take place simultaneously.

Multiple IEEE 1451.4 Transducers with switch nodes may be connected in a multidrop configuration with a maximum of one of these Transducers in “active” mode and the rest in “passive” mode. The switch nodes may be used to change the functional mode of each IEEE 1451.4 Transducer.

Node-Lists are used to specify which nodes correspond to which IEEE 1451.4 Transducer(s). This is needed if two or more nodes (in one or more IEEE 1451.4 Transducers) are connected to the same IEEE 1451.4 MMI.

A Node-List shall not be included if an IEEE 1451.4 Transducer contains only one node. If more nodes are present on the MMI and no Node-List is found, then each node will belong to an IEEE 1451.4 Transducer. For example, this permits a single thermometer node to be an IEEE 1451.4 Transducer.

4.2 IEEE 1451.4 Transducer configuration

The TEDS shall contain information about the node configuration in the IEEE 1451.4 Transducer and the possibility to be used in a multidrop configuration.

4.2.1 Multiple transducers on one IEEE 1451.4 MMI

An IEEE 1451.4 Interface can only handle one analog signal. A Node-List shall be used to enable more than one transducer to share this interface and to distinguish among IEEE 1451.4 Transducers.

4.2.2 Multiple nodes in one IEEE 1451.4 Transducer

If a transducer includes more than one node, there shall be a memory node with a template, which includes a list of the node IDs in the transducer. The Node-List’s own ID is not included in the list; the number of nodes in the list is the total number of nodes minus one. The first node in the list is the next TEDS node, if any. If there are more nodes with memory, the memory shall be considered as contiguous, ranked by this list of nodes.

4.2.3 Communication with an IEEE 1451.4 Transducer

Configuration of the IEEE 1451.4 Transducer shall be performed by the use of dedicated nodes. The TDL contains the information needed to change the configuration.

Configuration changes allow a number of features:

- Self-test
- Reset of transducer
- Multiplexing between transducers in an IEEE 1451.4 Transducer
- Gain shift in an IEEE 1451.4 Transducer
- Offset adjustment
- Change of filter settings in an IEEE 1451.4 Transducer

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

4.2.4 Communication to several IEEE 1451.4 Transducers on one IEEE 1451.4 MMI (multiplexing)

Enabling or disabling the analog function of different IEEE 1451.4 Transducers permits several IEEE 1451.4 Transducers to be connected to the same IEEE 1451.4 Interface. The Node-List groups the nodes for each IEEE 1451.4 Transducer making multiplexing possible.

4.3 Compliance with this standard, IEEE Std 1451.4-2004

This clause describes compliance levels with this standard for systems. The tiers are based on the following:

- Support of transducers with Basic TEDS and Class 1 or Class 2 MMI.
- Support of transducers with more information than Basic TEDS and Class 1 or Class 2 MMI (including use of dynamic TEDS, and Assign statements but not multinode support and use of Extended Functionality).
- Full support of transducers with all types of TEDS and Class 1 or Class 2 MMI (including multinode support and use of Extended Functionality, dynamic TEDS and Assign statements)

A TEDS is dynamic when it contains data structures of variable size, e.g., StructArray and Strings (whereas Select Case is not).

4.3.1 Tier 1—Minimal system capability

As a minimum, any system capable of using or delivering Basic TEDS data from transducers with Class 1 or Class 2 MMI and comprising a memory device, which contains data conforming to the this standard's Basic TEDS (data that follows the defined format of manufacturer, model, version letter, version number and serial number), shall be considered Tier 1-compliant.

4.3.2 Tier 2—Standard system capability

Any system capable of using or delivering data from transducers as described in Tier 1, and from transducers using advanced templates with, for example, dynamic TEDS or Assign statements, shall be considered Tier 2-compliant. (This does not require capability to handle multinode transducers and Extended Functionality).

4.3.3 Tier 3—Extended system capability

Any system capable of using or delivering data from transducers as described in Tier 2, and that can use and control transducers with multinodes and Extended Functionality, shall be considered Tier 3-compliant.

4.3.4 Information about systems using external data

For systems using Appended TEDS (see 7.4.10.2), the following is defined:

> Any system fulfilling any of the above tiers and capable of using Appended TEDS shall be designated by the words: Uses Appended TEDS.

Although the case with no memory in or attached to the transducer has nothing directly to do with the transducers, this is included to facilitate the use of data stored elsewhere. If a conflict occurs where data are found both in a transducer and in a database, the data in the transducer shall have priority.

To give the users a possibility to describe systems which are not capable of using transducers with MMI as described in the above tiers, the following is defined:

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

— Any system without an MMI but capable of using external data for transducers not comprising a memory device with Basic TEDS, but for which ID data are given in some other way and data following the rules in this standard are found in a defined database, shall be designated by the words: Uses IEEE 1451.4 formatted data.

Table 1—Compliance table

Compliance	Basic TEDS and MMI support	Standard TEDS support (not multinode and Extended Functionality)	Extended TEDS support
Tier 1	Yes	No	No
Tier 2	Yes	Yes	No
Tier 3	Yes	Yes	Yes

5. Transducer Electronic Data Sheet

The TEDS consists of data pertinent to an IEEE 1451.4 object or transducer, stored in memory, and conforming to storage and transmission formats contained in this standard, IEEE Std 1451.4-2004.

5.1 Basic TEDS

As a minimum, the TEDS in an IEEE 1451.4 Transducer shall contain a Basic TEDS that uniquely identifies the transducer.

5.1.1 Basic TEDS content

The Basic TEDS shall be comprised of 64 bits, including the Manufacturer ID (14 bits), model number (15 bits), version letter (5-bit character code), version number (6 bits), and serial number of the device (24 bits). This data shall be organized according to the format described in Table 2. The Basic TEDS shall be contained in a non-volatile memory. A checksum byte, insuring the integrity of the Basic TEDS data, and the 31 bytes following the checksum byte shall be provided as described in 6.2.2. Only the total 64 bits give uniqueness. Model numbers and serial numbers can be the same with different manufacturers, and a manufacturer can have the same serial number for several models, etc.

NOTE—Many transducers have been manufactured using draft versions (IEEE P1451.4, version 0.9) of this standard, which described a different format for the basic TEDS than that used in this standard. For information about these, see Annex I.[7]

A memory map of the Basic TEDS is shown in Table 3.

[7]Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table 2—Basic TEDS content

Manufacturer ID	Model number	Version letter	Version number	Serial number
14 bits (17–16381) [a]	15 bits (0–32767)	5 bits Char (A–Z, data type Chr5)	6 bits (0–63)	24 bits (0–16777215)

[a]The Manufacturer ID values 0–16, 16382, and 16383 shall be used as described in Figure 6.

The Manufacturer ID is an enumeration of the manufacturers see Annex J.

The model number may optionally be defined by the manufacturer as an enumeration of a model number or name (see Annex J). The distribution of this list is the responsibility of the manufacturer.

Basic TEDS content and all other TEDS data shall be written and read lsb first.

5.1.2 Basic TEDS format and placement

In commercially available nodes containing a one-time programmable (OTP) Application Register (see "DS2430A 256-bit 1-Wire® EEPROM" [B3][8], for example), Basic TEDS shall occupy the Application Register to safeguard the data from tampering or accidental destruction. Basic TEDS shall follow the format described in Table 3. (Note that the numbers in parentheses after each item, e.g., Manufacturer ID (7:0) indicates a bit count starting at zero). The checksum for Basic TEDS and the first 31 bytes of TEDS data shall be calculated as described in 6.2.2 and shall be contained in the first byte of the TEDS memory. In nodes containing an Application Register, the content of the Status Register indicates whether the OTP Application Register is locked, i.e., the location of the Basic TEDS data, or unused (locked = IEEE 1451.4 Basic TEDS; not locked = not IEEE 1451.4).

5.2 IEEE, User, and Manufacturer TEDS

TEDS data shall be contained in non-volatile memory. If the TEDS occupies more than one memory node, it shall contain a Node-List in one of the nodes, according to the format defined in 6.3.1, to determine the exact extent of the TEDS data. The TEDS shall contain data defined by an IEEE-defined template contained in Annex A, a template published by the manufacturer of the device containing the TEDS, or a template defined by the device user.

5.3 Data format and templates

IEEE 1451.4 TEDS data formats shall be defined by templates as described in Clause 6. A template is a documented definition of the placement and significance of each piece of data stored within the TEDS memory. The template shall not be contained within the TEDS data, but the TEDS data shall identify which template is to be referenced in interpreting the TEDS data, as defined in Clause 6. Templates shall be accessible to the program code (T-block) used to write and read the TEDS data.

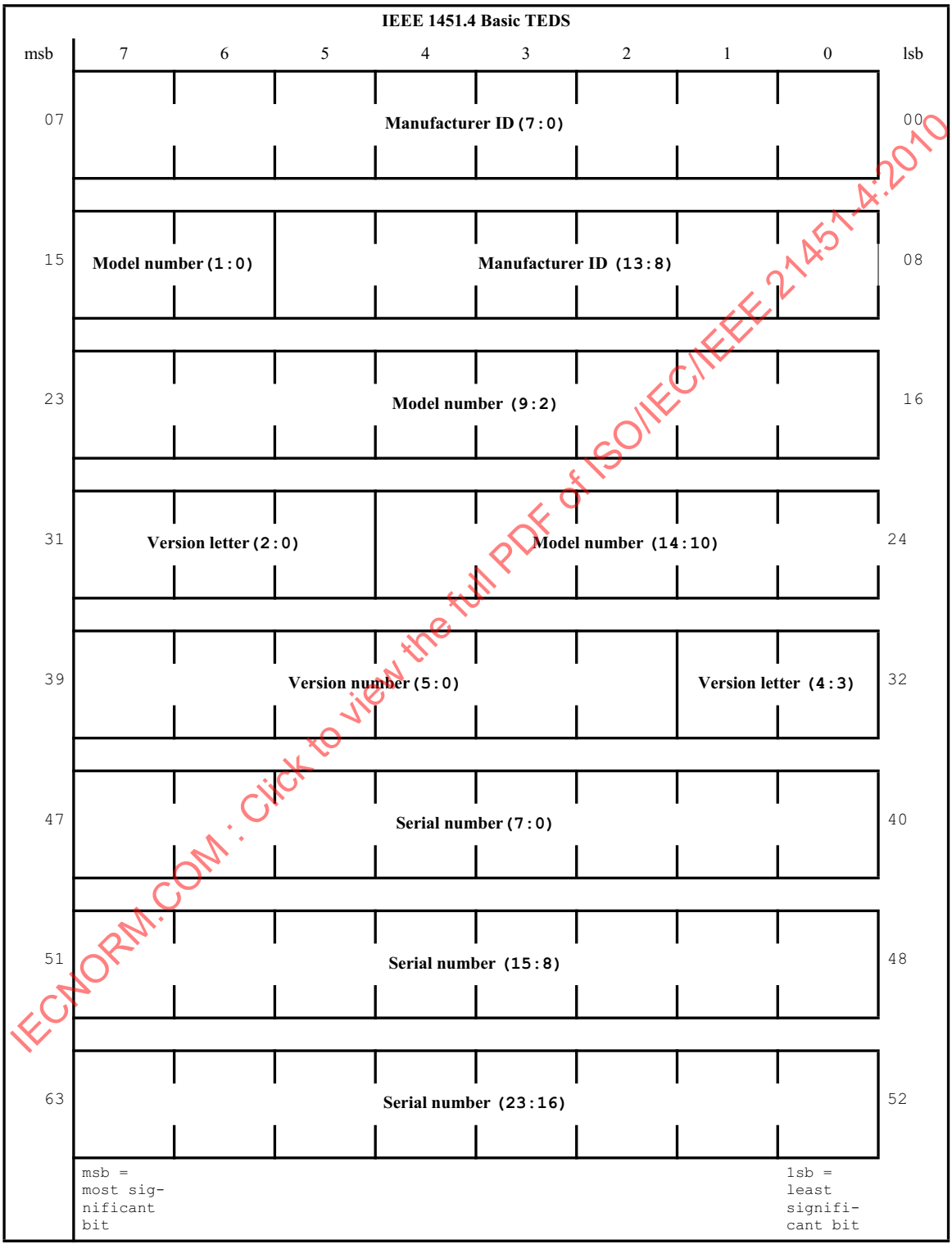
The TDL, as defined in Clause 7, shall be the language used in writing templates. A definition of the code (T-block) necessary to interpret the TEDS data, through the use of a template, is contained in Clause 9.

[8]The numbers in brackets correspond to those of the bibliography in Annex M.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table 3—Memory map, IEEE 1451.4 Basic TEDS



IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

5.4 Nodes, addresses, Family Codes, URN, and CRC

TEDS data stored in memory not directly accessible to a processor or controller shall be contained in one or more independently addressable memories, called nodes. Any processor, controller, or other device which contains TEDS data in its memory and which can transfer this data to another processor, controller, device or program shall be considered to be a node. Nodes may also contain functions other than memory.

Each node shall contain a permanently stored 64-bit URN, used to control access to the TEDS memory. See Annex H for information about the procedures to obtain URNs. Node-control devices supplied by manufacturers shall contain URNs. The URN is the basis upon which node-addressable, digital communication takes place, within the multidrop architecture. Nodes shall contain, as a part of the URN, a Family Code, which is used to identify the device functions and specific communication commands for the node. Family Codes and Command Description Files for commercially available node devices are listed in Annex E and "1-Wire Master Device Configuration" [B1].

The URN shall be read lsb first. The first 8 bits read from the 64-bit URN shall contain the Family Code of the node to be followed by a 48-bit unique serial number and then by an 8-bit CRC code, to insure data integrity. The CRC value shall be generated with a shift register punctuated by XOR functions, as illustrated in Figure 2, to generate the CRC polynomial "$X^8 + X^5 + X^4 + 1$." Beginning with the shift register cleared to zero, if the Family Code and serial number (the first 56 bits) are shifted into the register, the resultant register contents will be the CRC value. Continuing to shift the 8-bit CRC byte into the register will clear it to zero, once again, if no read errors have occurred.

CRC errors are most often caused by data collisions between two nodes transmitting simultaneously. The wired-AND nature of the bus will give priority to logic state 0. This is the basis of the Search ROM Command, in which nodes return one bit of URN data followed by its complement, to the master. Should both returned values be zero, the master shall surmise that two (or more) nodes are present, with unequal values for that bit. The master then sends either a one or a zero, and the node(s) not matching that value, for that bit, drops into an inactive mode, awaiting reset. Continuation of this bit-by-bit elimination process inactivates all but one node, after all 64-bits are polled.

Additional tutorial information on the use of the CRC is contained in "DS2430A 256-bit 1-Wire EEPROM" [B3], Smiczek et al. [B12], Annex E, and Annex G.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

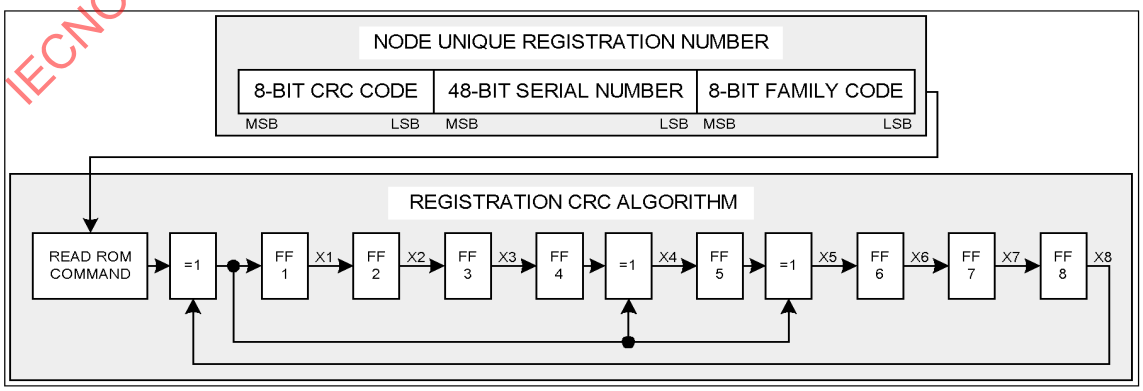


Figure 2—URN and CRC generation

© IEEE 2004 – All rights reserved

5.5 Data transmission

Transmission of data to and from IEEE 1451.4 TEDS memories, or other functions, which are contained in nodes, shall be according to the addressable, multidrop, serial, 2-conductor bus protocol described in Clause 8. An example memory node device is described in "DS2430A 256-bit 1-Wire EEPROM" [B3] and guidelines for the implementation of the protocol are contained in "1-Wire Master Device Configuration" [B1], Smiczek et al. [B12], Annex E, and Annex G. Transmission of data to and from an IEEE 1451.4 TEDS memory, which is contained in a separate IEEE 1451.4-compliant transducer package, shall utilize serial data transmission according to the MMI definition in Clause 8.

5.6 Structure of the TEDS data system

The block diagram of the TEDS data system in Figure 3 shows a hierarchy of functions necessary to house the TEDS data within a physical memory, and to allow accessibility of that data, via a Data Transmission Protocol. Note that the figure illustrates all parts of a fully implemented IEEE 1451.4-compliant TEDS system, in which the TEDS is contained within a node, located inside a transducer, which contains an MMI. TEDS data are available to a measurement/control device containing the 1451.4 Transducer Interface and Transducer Block and a Template Library, to fit into the IEEE 1451.1 NCAP Object Model. Definitions of the templates and their structure are contained in Clause 6 and Clause 7. The Transducer Interface and Transducer Block are defined in Clause 8 and Clause 9, respectively. The NCAP Object Model is described fully in IEEE Std 1451.1-1999.

Subsets of this implementation shall be recognized, under this standard, reduced to a minimal implementation, in which the IEEE 1451.4 TEDS has been assimilated into the Transducer Block, along with a single template, as might be the case for a small instrument containing an integral transducer.

Included, for reference, in the block diagram are functions not defined in this standard, but which may be found in a typical application. Following are brief descriptions of these functions.

Within the IEEE 1451.4 Transducer, an analog transducer shares the MMI. The transducer connection is described in Clause 8.

TEDS data enters or exits a 1451.4 Node via the Data Interface, which is a control and transmit/receive function described in Annex E and "1-Wire Master Device Configuration" [B1]. Each transaction through the Data Interface is qualified by the URN, which controls communication with individual nodes connected to the MMI.

A node might contain additional functions, such as switches, counters, temperature measurement, etc., as represented by the Extended Function, the access for which shall be found in the device specification, as implied by the Family Code, contained in the URN, listed in "1-Wire Master Device Configuration" [B1].

Within the measurement/control device, analog data enters or leaves the application program block through an analog system, which shares the 1451.4 MMI. All IEEE 1451 functions are parts of the overall program code, which may also include application program code for the specific function of the instrument, a graphic user interface (GUI) and display, network interface, and so on. The 1451.4 Transducer Interface contains a low-level command structure necessary to execute the communication protocol required by the TEDS nodes. Examples to guide the implementation of this protocol are found in "DSA2430A 256-bit 1-Wire EEPROM" [B3], "1-Wire Master Device Configuration" [B1], Smiczek et al. [B12], Annex E, and Annex G.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

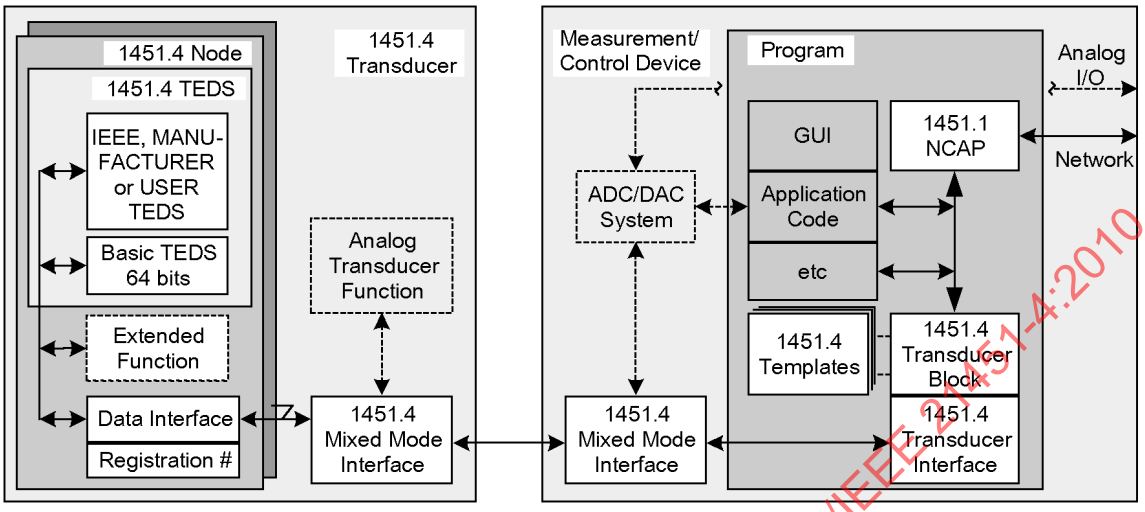


Figure 3—IEEE 1451.4 TEDS data system block diagram

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

6. Templates

This clause describes the usage of the template structure to read or write nodes and extract TEDS data. A measurement/control device will be attached to a bus that will have one or more IEEE1451.4 Transducers attached to the bus. Each attached IEEE1451.4 Transducer will contain one or more nodes. Each node of an attached IEEE1451.4 Transducer may be accessed by the measurement/control device via the MMI of the IEEE1451.4 Transducer through that node's Data Interface.

6.1 Overview

A template describes the memory structure of TEDS data that contains information about the identity of the transducer and transducer specific data. It shall be written in the TDL and templates shall reside in the T-block. Templates shall be made available to the T-block as one or more 8-bit ASCII text template description files with the extension .tdl. The IEEE1451.4 T-block provides an interface between transducers and application software in an IEEE 1451-compliant device. It includes methods for extracting and encoding IEEE1451.4 TEDS data using the templates. The T-block also permits the IEEE1451.4 Transducers and Interfaces to be configured and managed. See Clause 9 for more details.

The template structure is designed with the main objective to use small-sized memory in an efficient manner. Therefore, the template used defines the memory structure. This means that a specific part of the memory may contain different types of information depending on the specific template used in a particular transducer.

The Basic TEDS, common to all TEDS, provides transducer identification. A template provides a memory structure targeting the specific data set for a given transducer. Templates (in the form of template description files) for several transducer types are defined and more may be added (see Annex A).

The behavior of the template structure is described in the following flowcharts.

© IEEE 2004 – All rights reserved

6.2 Discovery of the transducer(s) present

The first action in order to eventually read the content of the TEDS on an interface is to switch to Digital Mode and initialize the bus communication (for details, see Clause 8 and "DSA2430A 256-bit 1-Wire EEPROM" [B3]) as shown in the flowchart Figure 4.

The system shall then read all the URNs in the nodes present on the interface (see Clause 8 and "DSA2430A 256-bit 1-Wire EEPROM" [B3] for details). The first 8 bits read (reading and transmission is from lsb to msb) of the identification number shall contain the Family Code, which gives information about the size and structure of the physical memory (see "DSA2430A 256-bit 1-Wire EEPROM" [B3] for a description of the memory of a commonly used device. Family Codes and Command Description Files for commercially available node devices are also listed in Annex E and "1-Wire Master Device Configuration [B1]). The next 48 bits shall contain a unique serial number, and the last 8 bits read shall contain a CRC code of the first 56 bits (Family Code and unique serial number) (see Clause 5 or "DSA2430A 256-bit 1-Wire EEPROM" [B3] for more details).

Data memory may now be read. A checksum test should be made for each page read. The checksum byte(s) shall be removed before further processing of the data. If a device with ID has no additional memory, that device shall be ignored for TEDS reading purposes.

The result after the discovery phase is a collection of data for each node comprising the 64-bit URN and the data memory stripped of the checksums.

6.2.1 Bit error check system for the URN

CRC error checking is well-known and easy to construct. More details are given in 5.4 and may be found in "DSA2430A 256-bit 1-Wire EEPROM" [B3].

6.2.2 Bit error check system for the data memory

A checksum test shall be used for bit error checking. When data is stored, data memory shall be split into pages of 256 bits (32 bytes), and a checksum calculation shall be made for each page. The last 31 bytes are added without carry, and two's complement of the resultant byte is stored in the first byte, the checksum byte. When reading the memory, all 32 bytes are added without carry and if the resultant number is zero the transmission was successful. For devices containing an OTP memory part (sometimes called an Application Register), the checksum in the first page after this shall be the checksum of the OTP and the first page.

6.3 Identification of transducers and their nodes

After the discovery, data memory readout and a bit error check for data integrity, data read from each of the contained nodes has to be concatenated and assigned to its respective transducers to describe the TEDS data for each transducer. This process is depicted in the flowchart in Figure 5. The process creates a Transducer Collection containing all URNs for each node contained in each transducer and includes the corresponding data memory stripped of checksum bytes for each transducer.

The process is described in three steps:

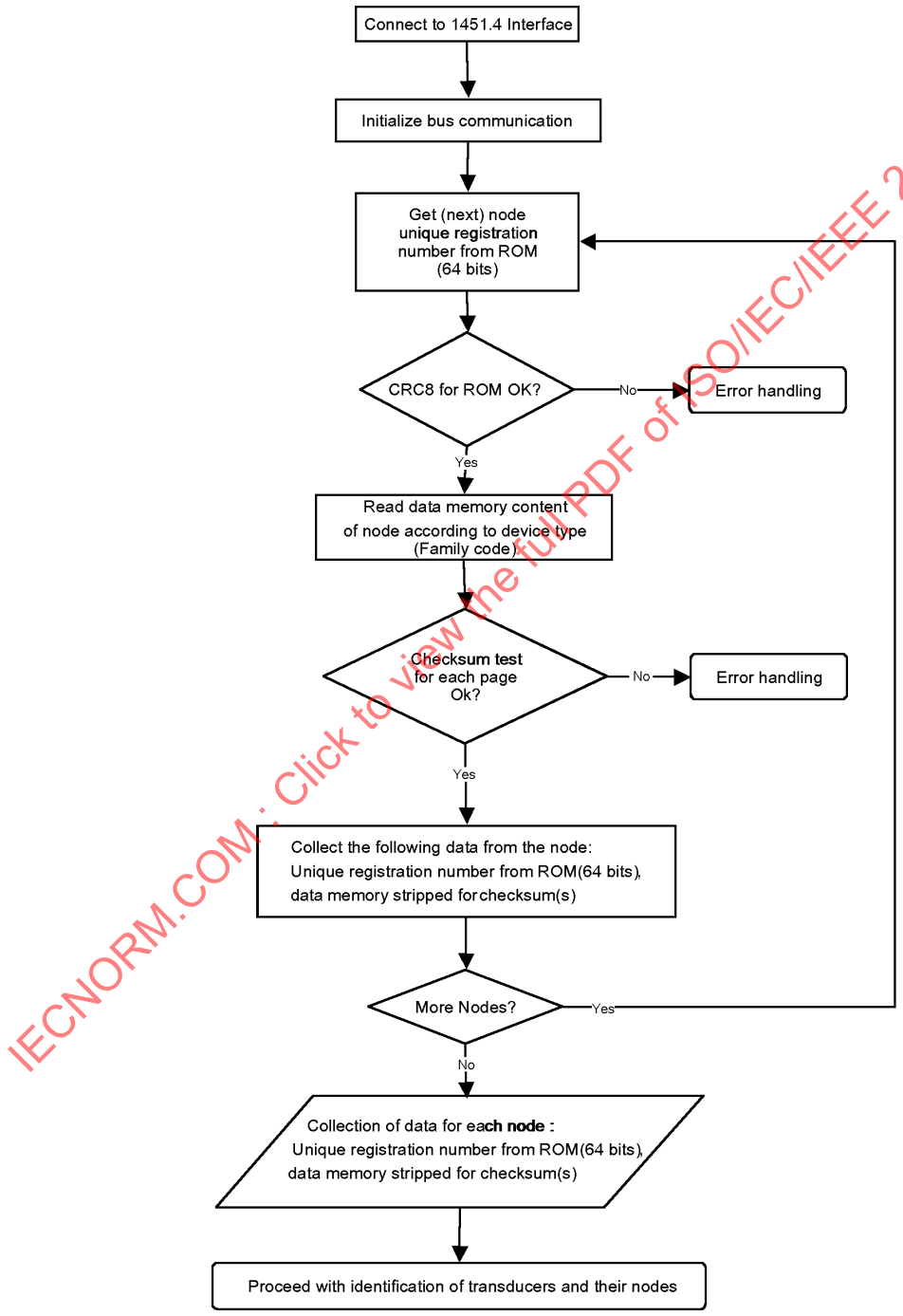
1) Data from each node is read and examined to detect the presence of a Node-List. The manufacturer or special use selector (see 6.3.1) is contained in the first 14 bits of memory. If a node has a selector containing the value 1, this node contains the Node-List for this IEEE1451 Transducer. If an IEEE1451.4 Transducer contains more than one node, then one of the nodes will contain exactly one Node-List. When a Node-List is detected for a given transducer, that transducer is added to the Transducer Collection data structure.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

2) Then, all the listed URNs (64 bits) are read and appended to the Node-List for the new transducer, and all these nodes are marked as processed.

3) Any remaining (unmarked) nodes must then be individual transducers, and they are added as such to the Transducer Collection. (If a transducer contains only one node, a Node-List does not exist for this transducer; however, the URN for the one node contained on this transducer is added to the Transducer Collection).



IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Figure 4—Flowchart showing the discovery of the transducer(s) present

© IEEE 2004 – All rights reserved

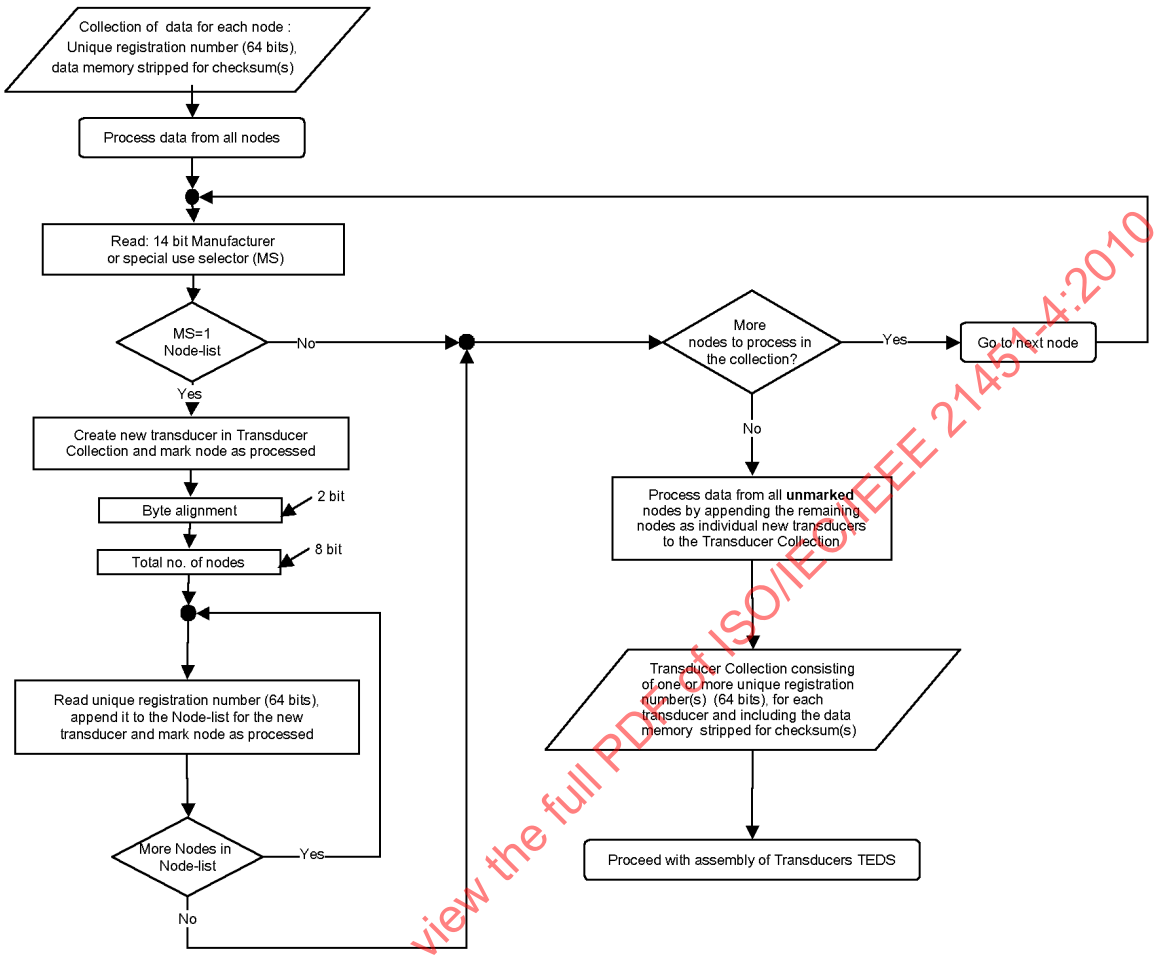


Figure 5—Flowchart showing the identification of the transducer(s) and their nodes

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

6.3.1 Manufacturer or special use selector

The first 14 bits read from a node's data memory shall contain a selector defined in Table 2. The majority of the valid selector values (17–16381) identify the manufacturer using a list maintained by IEEE. This standard describes how TEDS are stored in a compact form. The presence of a manufacturer number from the IEEE list implies compact TEDS.

A short verbal description of the meaning of the remaining fields follows:

— The values 0, 3, 5–9, 11–13, and 15–16 have been set aside for future use in a revised standard.

— The value 1 indicates that this node contains a Node-List.

— The value 2 indicates that the remaining part of the memory contains TEDS in a free-form format. This requires additional information not covered by this standard to be used by an application.

— The value 4 indicates the remaining data memory is a continuation of the previous node's data memory to be read and appended to the current TEDS image; do not append the selector value.

— The values 10 and 14 have been used by the beta implementation of this standard and shall therefore not be used.

— The value 16382 indicates a user-defined template. To be used when a transducer, containing the basic TEDS from a manufacturer, is provided with additional special information by a user. The user

© IEEE 2004 – All rights reserved

shall supply the template description file and store it next to the original description file in order for the system to read and interpret the TEDS data.

— The last value, 16383, indicates the remaining data memory shall not be read. Reading shall continue with the next node listed in the Node-List.

Table 4—Manufacturer or special use selector

Value	Meaning
0	Future use
1	IEEE 1451.4—Node-List
2	IEEE 1451.4— free-form TEDS only
3	Future use
4	Memory continued
5–9	Future use
10	Reserved by beta format
11–13	Future use
14	Reserved by beta format
15–16	Future use
17–16381	Enumeration of Manufacturers
16382	User defined template
16383	Continue in next node of Node-List

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

6.4 Assembling the Transducer TEDS

Each individual TEDS, from each node of each attached transducer to be added to the Transducer Collection, shall be read from each node's data memory and appended to create the Transducer TEDS. To obtain the Transducer TEDS from the Transducer Collection, each individual TEDS shall be assembled. This process is shown in the flowchart in Figure 6.

The final collection of individual Transducer TEDS is composed of each of the TEDS read from each of the single-node transducers, read directly, and the concatenated TEDS, which is obtained by reading each of the nodes as indicated in the Node-List for each multinode transducer and read following the rules indicated by the manufacturer or special selector described in 6.3.1.

An example of the process using Node-Lists with three nodes present is described in the following:

— Read the Node-List.

— Read the first node in the Node-List, which is the start of the TEDS image containing a valid Manufacturer ID. Maintain the manufacturer or special use selector.

— Read the second node in the Node-List. The manufacturer or special use selector is 4, which means continued memory. Append the memory to the TEDS image without selector.

© IEEE 2004 – All rights reserved

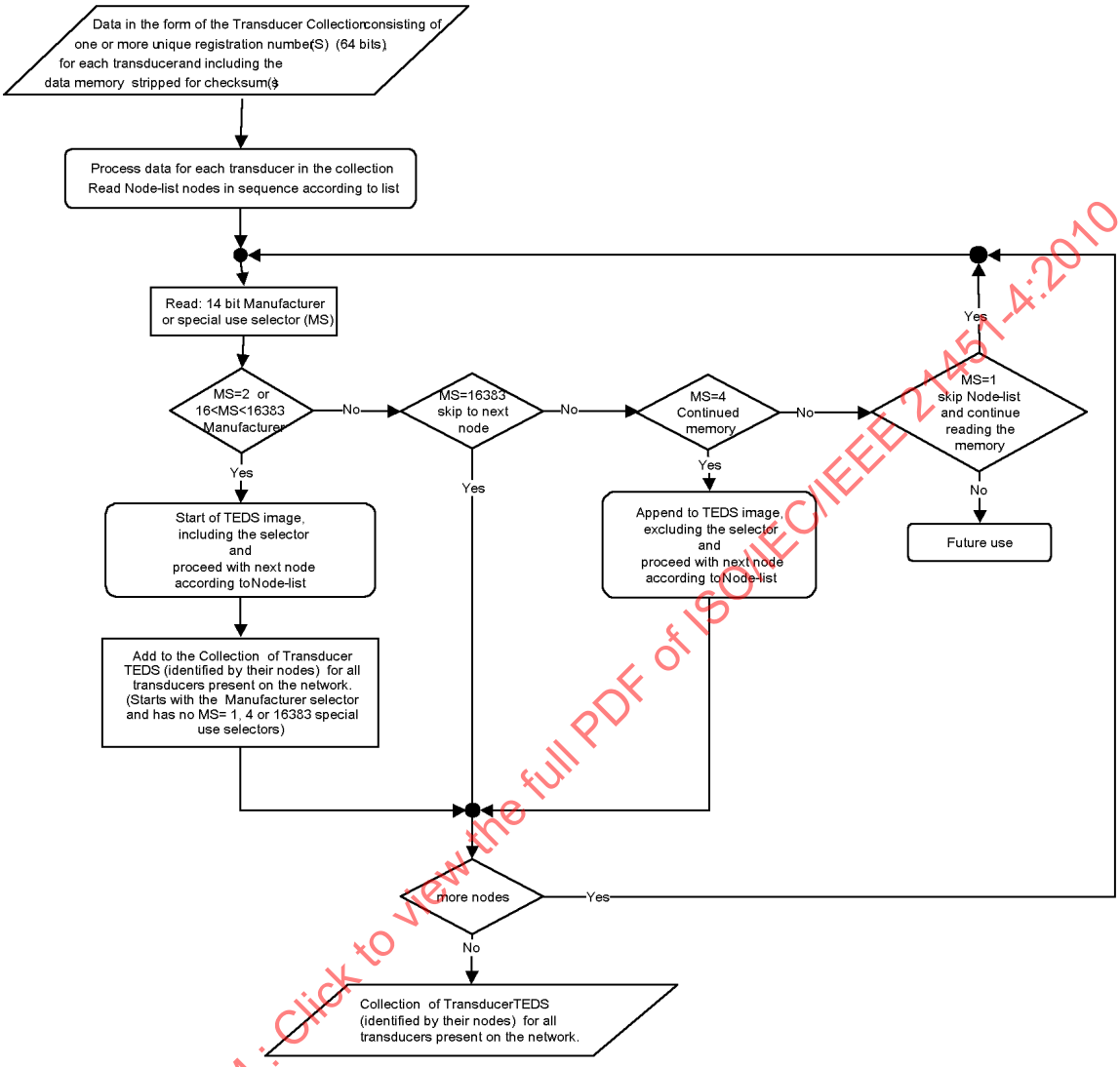


Figure 6—Flowchart showing the assembly of the Transducer TEDS

— Read the Node-List node contents after the Node-List. The manufacturer or special use selector is 16383, which means skip to next node, which means the reading is finished, as there were no more nodes. (If the selector had been 4, the memory content should have been appended to the TEDS image without the selector).

6.5 Parsing the Transducer TEDS

The last step in the process to obtain the complete transducer data is to parse the individual Transducer TEDS in the Collection of Transducer TEDS. This is described in the flowchart shown in Figure 7.

The Manufacturer or special use selector described in 6.3.1 indicates now either an Embedded Template (including Basic TEDS) or a manufacturer or user-defined TEDS. The manufacturer selector and the following 50 bits give the identification of the transducer described as Basic TEDS in 5.1.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

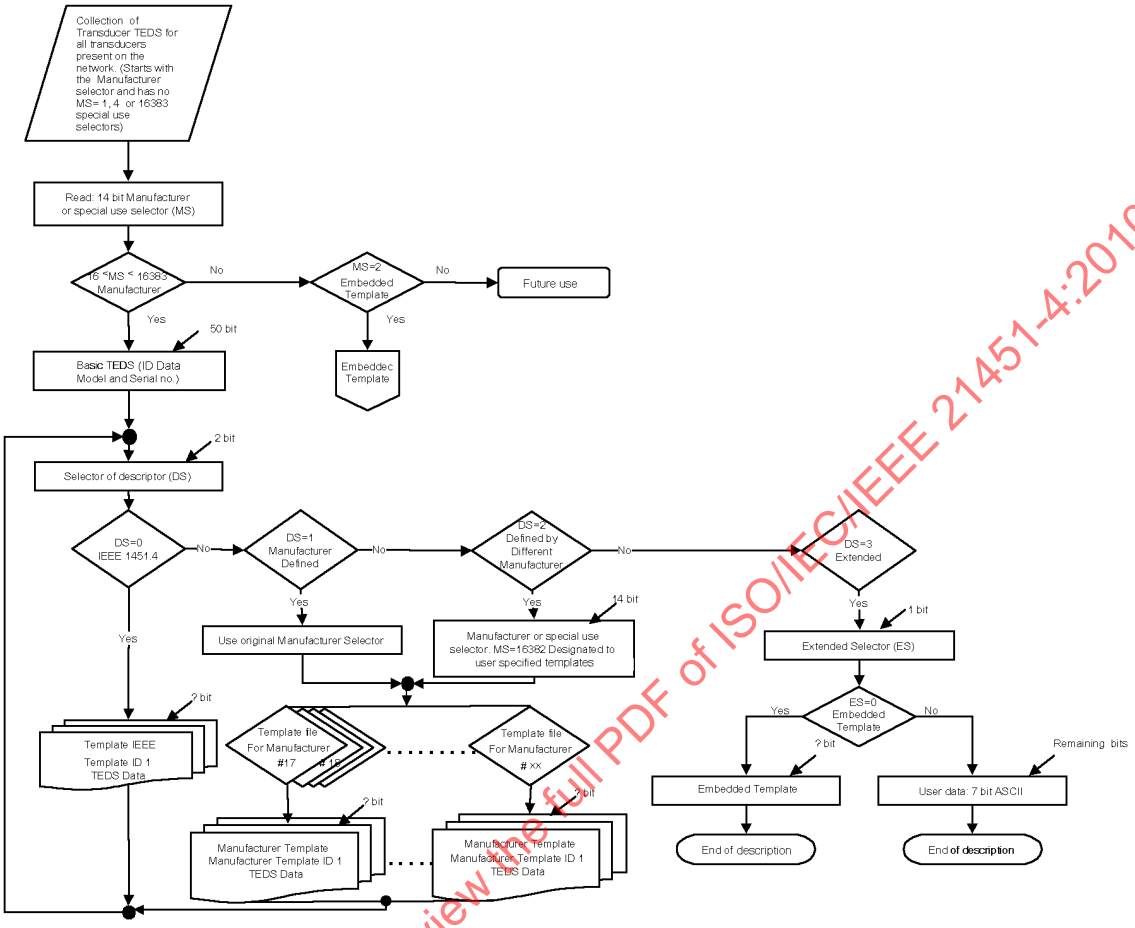


Figure 7—Flowchart showing parsing of the Transducer TEDS

A 2-bit selector of descriptor, described in 6.5.1, follows. This selector together with the original or a new Manufacturer ID leads to the correct set of subtemplates. A new 2-bit selector shall follow after reading a template. If the value of the 2-bit selector is 3, an extended template description is following. This may be an Embedded Template (not including Basic TEDS) or user data in 7-bit ASCII format. This will typically be the way many templates end, leaving the remaining memory for the user.

6.5.1 Selector of descriptor

The selector of descriptor is the field describing whether the template is an IEEE 1451.4 defined template or not. The field is a 2-bit selector with the range 0 to 3, with the following meaning:

If the value of the 2-bit selector is:

0. (zero), the template is defined by IEEE 1451.4 (see 6.3.1).

1. (one), a manufacturer-described area is following. The Manufacturer ID given by the first manufacturer selector is used to select the group of subtemplates among which the one with the number read out from the TEDS shall be found in the description file provided by the manufacturer.

2. (two), another manufacturer or the user has defined the template. A new manufacturer or special use selector shall follow. If the selector is 16382, it is the user who defined the subtemplates (a validation

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

code for the description file shall be included in the user-defined TEDS). For IEEE registered Manufacturer IDs, this defines the group of subtemplates.

3. (three), extended. A 1-bit extended selector shall follow. If the extended selector is 0, a free-form TEDS follows. The template to be used to read this free-form TEDS shall be found in the description file provided by the user or manufacturer of this free-form TEDS.

6.5.2 Template ID

A Template ID at the beginning of each template shall identify the template. The number of bits used to identify the template shall be unique for the descriptor of the template. The descriptor of a template shall be either IEEE, manufacturer, or user. See also Clause 7.

7. Template Description Language (TDL)

This clause describes the syntax and semantics of the language used in the templates. Using this language, each template defines the bitmapping of its associated TEDS.

7.1 Overview

A formal grammar providing the exact syntax of the TDL is provided in Annex C. The functional, semantic usage of this grammar is described in detail in this clause.

7.1.1 Purpose

Manufacturers may store the relevant details associated with a transducer in the TEDS and provide the information necessary to decode those details in a template written in the TDL. A user is thus assured that transducer information contained in a TEDS can be retrieved if the user has the appropriate template and a parser capable of interpreting TDL. This overcomes difficulties in providing either new transducers or upgrades to existing transducers. That is, when a new transducer is produced, or the TEDS information for an existing transducer is updated, the ability to read the TEDS can be communicated simply by providing a new template.

7.1.2 Commands

The TDL commands are described via functionality. Control commands provide information regarding the general flow through the template. Identification commands uniquely identify the template and associated TEDS. The property commands are, in some sense, the core of the template, in that they provide the mechanism for reading in the values for the properties of the transducer. The Extended Functionality operator augments the ability of these core property commands.

7.1.3 Bitmapping

It is important to understand that what the template does is to map out the TEDS bit-by-bit. Although some of the TDL commands serve metafunctions and don't indicate any mapping of particular bits, the majority of commands explicitly indicate that the next *n* bits, as indicated in the command syntax, are to be interpreted in a specific manner. Thus, as each command in a template is parsed and implemented, if the command maps bits in the TEDS, those bits are read from the TEDS and the input pointer to the TEDS is advanced to the next bit following the bits just read.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

7.1.4 Basic TEDS

Templates do not describe the Basic TEDS, which is described in 5.1.

7.1.5 Case sensitivity

The TDL shall not be case sensitive.

7.1.6 Character strings

There are several places where a template developer may specify an unquoted character string. More specifically, these can be specified:

— In the enumerate command.

— In the structarray command.

— As a suffix to the %MDEF_properties.

An unquoted character string shall be a contiguous set of characters (no white space) consisting of alphanumeric characters (A–Z and 0–9) and any of the special characters ^, $, _, !, @, #, or &. Its first character shall not be a number (0–9).

A quoted character string (or quoted string) shall be delimited by double quotes and may contain any printable ASCII character except double quotes, carriage returns, or line feeds.

7.2 Identification commands

These commands provide information to uniquely define a template. A template is the concatenation of one or more templates. Bits in the TEDS are used to uniquely identify each template. This process is described in 6.5.

7.2.1 Template and EndTemplate

These commands define the beginning and end of a template. Every `Template` command shall have a matching `EndTemplate` command.

Syntax:

```
Template <Manufacturer ID>, <ID number of bits>, <Template ID>,<title>
        <commands>
EndTemplate
```

The <*Manufacturer ID*> shall be a decimal integer. It shall have value 0 for an IEEE 1451.4 defined template; for a manufacturer-defined template, it shall have the value of the manufacturer that created the template. Subclause 6.3.1 describes these manufacturers' IDs. Note that 16382 is a special Manufacturer ID reserved for user-defined templates.

The <*ID number of bits*> shall be a decimal integer indicating the number of bits to be read.

The <*Template ID*> shall be a decimal integer.

The <*title*> shall be an optional quoted string.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

The <*commands*> shall be any set of valid commands excluding `Template` and `EndTemplate`.

The `Template` command shall be the first line in any template. The `Template` command shall map <*ID number of bits*> bits in the TEDS. The `EndTemplate` command shall be the last line of the template. The `EndTemplate` command shall not map any bits in the TEDS.

7.2.2 Template ID

Templates are associated with a Manufacturer ID and each template shall have a unique ID within this Manufacturer ID association. For each Manufacturer ID, the number of bits used by the Template ID shall be the same. As described in 6.5, when a new template is expected during the parsing of the TEDS, it is necessary to locate a template associated with the current Manufacturer ID to determine the number of bits to be read to interpret the Template ID.

Example:

```
Template 0, 8, 13, "Descriptive Title of Template"
EndTemplate
```

where

```
Manufacturer ID = 0
ID number of bits = 8
Template ID = 13
```

This refers to the IEEE 1451.4-defined template number 13.

7.2.3 Template concatenation

As illustrated in Figure 7, the general approach to bitmapping of the TEDS is through a series of templates. That is, after the Basic TEDS is read, several templates might be needed to completely parse the TEDS. For example, the TEDS may be defined by the Basic TEDS, followed by an IEEE standard template, then followed by a manufacturer-defined template, and completed by a user-defined template. Each of these templates is identified in the TEDS using template IDs.

7.2.4 Template validation

It is possible that an error could occur between the data read in the TEDS and the bit definitions in a template. The point of validation is at the end of a template. That is, if an error occurs during the parsing of a template, then it is assumed that the entire template is, and all further templates are, invalid. Data read from previous templates are assumed valid.

7.2.5 Template file checksum—Validation_Keycode command

A single file may contain more than one template (with each delineated by a `Template` and `EndTemplate` pair). The last line of each template file shall be a `Validation_Keycode` command.

Syntax:

Validation_Keycode <*checksum*>

The <*checksum*> shall be a decimal integer.

Specifically, the algorithm is

1) Initialize the 32-bit integer checksum value to 0.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

2) For each line in the file up to but excluding the last line containing the `Validation_Keycode` command, do step 3.
3) For each ASCII character, including the carriage returns and line feeds, in the line, do steps 4–5.
4) Convert the ASCII character to an unsigned 8-bit integer.
5) Add the 8-bit value to the 32-bit checksum.

Annex D contains a C++ example of implementing the checksum validation.

Example:

```
Validation_Keycode 12345
```

7.2.6 TDL_Version_Number

The `TDL_Version_Number` command identifies which version of the TDL was used at the time of creation of the template.

Syntax:

`TDL_Version_Number` <*version number*>

The <*version number*> shall be a positive decimal integer.

The value of <*version number*> indicates the version of the TDL at the time of template publication. Possible values and their meaning are provided in Table 5. There shall be one and only one `TDL_Version_Number` in a template. The `TDL_Version_Number` shall not read any bits from the TEDS.

Table 5—Version numbers

Version number	Meaning
0	Reserved for legacy use. Beta format.
1	Reserved for legacy use. IEEE P1451.4, version 0.9.
2	IEEE Std 1451.4-2004.

Condensed information about the legacy formats can be found in Annex J.

Example:

```
TDL_Version_Number 2
```

7.2.7 UDID (unique description identification)

The UDID is a unique identification for a template and the selections made in this template. UDID is not explicitly found in the TEDS nor the template description, but is calculated from the TEDS structure and template used. It is used as a way to reference the TEDS used in a transducer and in the UGID (see 7.2.8).

The syntax of the UDID shall be:

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

UDID $\{$I<tid>$\{$— <sc>$\}_0^\infty$ $\left|\right.$M<mid>:<tid>$\{$— <sc>$\}_0^\infty$ E$\left|\right.$U$\}_1^\infty$

where

I is an IEEE 1451.4 standard template,
M is a Manufacturer template,
E is an Embedded Template,
U means User data,
<tid> is the template identification number in decimal,
<sc> is the actual selected Select Case value in decimal,
<mid> is the Manufacturer ID number in decimal.

$\{\}_0^\infty$ indicates that this part can occurs between *0* to ∞ times in the parsing of the template.

$\{\}_1^\infty$ indicates at least one occurrence in the parsing of the template.

Example:

To identify IEEE 1451.4 template 28, with the following Select Cases:

SELECTCASE: “TEDS selection”
With CASE selected: “Combined TEDS for Preamplifier with Attached Microphone” (value = 1)

SELECTCASE “Extended Functionality”
With CASE selected: “Programmable Sens” (value = 1)

SELECTCASE “Additional Basic TEDS”
With CASE selected: “System” (value = 1)

SELECTCASE “System Test (Test Gain)”
With CASE selected: “Not available” (value = 0)

SELECTCASE “Transfer Function”
With CASE selected: “Specified” (value = 1)

and with User data.

This will result in UDID I28-1-1-1-0-1U.

7.2.8 UGID (unique group identification)

The group identifier may be used as a mechanism for grouping templates. How this is used is not defined in this standard. However, in order to maintain global uniqueness, rules for deriving UGID are defined as follows:

Syntax:

`UGID` <”*identifier*”>, <”*description*”>

The <”*identifier*”> shall be a quoted text field.

The <”*description*”> shall be a quoted text field.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

This command shall not map any bits in the TEDS.

7.2.8.1 UGID derivation rules

The <"*identifier*"> shall be the UDID value in quotes from one of the templates in the group.

Example:

UGID "I28-1-1-1-0-1", "Preamp. with programmable gain"

7.2.8.2 Multiple occurrences of UGID

There may be more than one `UGID` in a template (most notably inside Cases of a SelectCase command). If there is more than one `UGID` command in a template, they shall occur in the SelectCases in such a way that the parsing of the template always results in only one unique UGID.

7.3 Control commands

These commands provide information regarding the general flow through the template as well as general descriptive information. None of these commands provides transducer specific information.

7.3.1 Abstract

This command provides comments describing the template that is to be displayed to the user in automated template editing software.

Syntax:

`abstract` <*description*>

The <*description*> is an optional text field delimited by end-of-line.

This command contrasts with the general Comment command in that it is to be used by template editing software. The <*description*> could be used to display to a user. Comments provided by the Comment command are to be viewed only by those directly accessing the template. In its normal use, there would be several `Abstract` commands at the beginning of the template that provides an overview of the template, its history, or its usage. When displayed, each <*description*> would occupy a single line. The catenation of these <*description*>s would thus constitute the abstract of the template.

The `Abstract` command shall not map any bits in the TEDS.

Example:

```
abstract IEEE P1451.4 template version 0.91
abstract rev. 6 date: 990308
abstract rev. 5 was approved via IEEE 1451.4 telcon 990308
```

7.3.2 Comments //

This command provides general comments inserted and used by template authors.

Syntax:

// <*description*>

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

The <*description*> is an optional text field.

These comments are for internal use to the template. This is in contrast to the Abstract command, which is intended to provide a description to a user. The Comment command may appear after any command. (Comment commands are allowed between templates. That is, they can appear after an `ENDTEMPLATE` command and before a `TEMPLATE` command.) The text between the double backslashes, //, and the end of line shall be textual comments. The Comment command shall not map any bits in the TEDS.

Example:

```
// This is a comment
%Gain, “Amplifier Gain”, Cal, 7, UNINT, “”, “”// Set Gain
```

7.3.3 SelectCase and Case

These commands provide the ability to use bits in the TEDS to determine different cases.

Syntax:

```
SelectCase <”description”>, <access_level>, <number_of_bits>
            Case <”description”>, <bit value>
                  <command list>
      EndCase
            …
            Case <”description”>, <bit value>
                  <command list>
      EndCase
EndSelect
```

The <*"description"*> shall be a quoted text field.

The <*access_level*> shall be either omitted or be one of the access levels identified in Table 6.

The <*number_of_bits*> shall be a positive decimal integer.

The <*bit value*> shall be a positive decimal integer less than $2^{<number\_of\_bits>}$.

The <*command list*> shall be a sequence of TDL commands.

These statements implement a conditional selection based on a set of bits in the TEDS. When the TEDS is being parsed and a `SelectCase` statement is encountered in the template, the number of bits indicated by <*number_of_bits*> is read from the TEDS. The <*bit value*> thus read is used to determine which case is executed. Commands found in the <*command list*> associated with that case are executed. All commands in <*command list*>s for other cases are ignored.

This command shall map <*number_of_bits*> of the TEDS.

Example:

```
SelectCase “Gain Values”, Cal, 2
      Case “Gain=16”, 0
            %Gain, “Open loop Gain”, Cal, 7, UNINT, ,= 10
      EndCase
      Case “Gain=32”, 1
            %Gain, “Open loop Gain”, Cal, 7, UNINT, ,= 20
      EndCase
      Case “Gain=64”, 2
            %Gain, “Open loop Gain”, Cal, 7, UNINT, ,= 40
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
        EndCase
        Case “Gain=128”, 3
              %Gain, “Open loop Gain”, Cal, 7, UNINT, ,= 80
        EndCase
EndSelect
```

In this example, the gain is set based on the 2-bit switch. Since the gain occupies 7 bits, there is a saving of 5 bits in the TEDS over having to read the gain directly from the TEDS.

7.3.4 StructArray

This command allows the creation of a set of properties treated as a single structure as well as the ability to dynamically create an array by reading the size of the array from the TEDS.

Syntax:

`StructArray` <*name*>, <*"description"*>, <*access_level*>, <*number_of_bits*>

<*property_list*>

EndStructArray

The <*name*> shall be an unquoted character string.

The <*"description"*> shall be a quoted text field.

The <*access_level*> shall be either omitted or be one of the access levels identified in Table 6.

The <*number_of_bits*> shall be a positive decimal integer.

The <*property_list*> shall be any set of properties.

When a `StructArray` command is parsed, the number of bits indicated by <*number_of_bits*> is read from the TEDS. The <*property_list*> is then read the number of times indicated by this initial value. Note that only properties shall be contained in the `StructArray` and it shall not contain any other TDL commands except possibly other `StructArray` commands.

Example:

```
STRUCTARRAY CalTable, "Calibration table", CAL, 7
        %CalPoint_DomainValue, "Domain Calibration Point (% of full
        span)", CAL, 16, ConRes, 0, 0.00153, "0.00", "%"
        %CalPoint_RangeValue, "Range Calibration Deviation (% of
        full span)", CAL, 21, ConRes, -100, 0.0001, "0.0000", "%"
ENDSTRUCTARRAY
```

7.3.5 Spacing

This command suggests a separation in formatting to higher-level applications between the previous field and the next field.

Syntax:

```
Spacing
```

This command is for display formatting only. This command shall not map any bits in the TEDS.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Example:

```
Spacing
```

Upon reading this command, a template display program might display a line to visually separate the commands preceding the spacing command from the commands following the spacing command.

7.3.6 Align

This command allows realignment of the input pointer to the TEDS along word boundaries, in essence, potentially skipping over several bits in the TEDS.

Syntax:

```
Align <word_size>
```

The <*word_size*> shall be a positive decimal integer.

The <*word_size*> might be thought of as the word size in the memory storing the TEDS. The align command thus moves the TEDS input pointer to the beginning of the next word. This requires that the processor know two things: the Current Position in the TEDS data stream (i.e., where the TEDS input pointer is currently pointing to) and the <word_size>. Assuming a starting address of 0, the TEDS input pointer shall be incremented by

<*word_size*> – (Current_Position mod <*word_size*>)

The Align command shall skip bits in the TEDS to align to the proper word.

Example:

```
Align 8
```

If the TEDS Current Position = 238, then the next 2 bits of the TEDS are ignored. That is, the TEDS input pointer is incremented by 2, so that Current Pointer = 240. That is,

8 – (238 mod 8) = 2

7.4 Property commands (%)

Property commands are identified by preceding a property tag with the percent sign, %. These commands provide the mechanism for assigning transducer specific values to the appropriate properties. These values are normally read from the TEDS. However, there is also an assignment option to the property commands that allows explicit assignment of a property value without reading the value from the TEDS.

7.4.1 Command structure

Syntax:

%<*property_tag*>,<*"description"*>,<*access_level*>,<*data_type*>,<*format*>,

<physical_unit>

or

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

%<*property_tag*>,<*"description"*>,<*access_level*>,<*data_type*>,<*format*>,

<physical_unit> = <value>

or

%<*property_tag*><*[subproperty]*>,<*"description"*>,<*access_level*>,<*data_type*>,

<format>,<physical_unit>

or

%<*property_tag*><*[subproperty]*>,<*"description"*>,<*access_level*>,<*data_type*>,

<format>,<physical_unit> = <value>

The <*property_tag*> shall be one of the property tags identified in Annex B.

The <*"description"*> shall be a quoted text field.

The <*access_level*> shall be one of the access levels identified in Table 6.

The <*data_type*> shall follow the syntax defined in 7.4.5.

The <*format*> shall be a quoted format description as described in 7.4.6.

The <*physical_unit*> shall be a quoted text field or array as described in 7.4.7.

The <*subproperty* > shall be an Extended Functionality subproperty as described in 7.4.9.

The <*value*> shall be a numeric or text value appropriate for the associated data type; see 7.4.8 for details.

The <*property_tag*> is always preceded by a percent sign, %, with no space between. There is no comma between <*property_tag*> and <*subproperty*>. There are no commas around the equals sign, =. All other parameters are separated by a comma ",". All parameters except for the <*property_tag*> and <*data_type*> are optional. However, if a parameter is omitted, a comma shall delimit the parameter field.

Detailed descriptions of each of the parameters are given in the following clauses. The use of the equals sign is described under 7.4.8. The use of <*[subproperty]*> is described in 7.4.9. The non-assignment property commands shall read bits from the TEDS based on the size indicated in the <*data_type*>. These bits are interpreted according to the <*data_type*> parameter, and the resultant value assigned to this property. The assignment property commands shall not read any bits from the TEDS.

7.4.2 Property list

Annex B lists and describes the property tags defined by this standard. Manufacturers may also create their own property tags as described in 7.4.10.1.

7.4.3 Description

The <*"description"*> parameter is a quoted text field. It is intended as a brief description to be displayed to a user.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

7.4.4 Access_level

The <*access_level*> parameter assigns a data access level for the value assigned to this property. This is intended to provide a level of security for the data. That is, different users should only be able to write data if they have appropriate access. Table 6 provides the allowable access level values.

Table 6—Access levels for transducer properties

Access level	Description
ID	Data used to identify the specific IEEE 1451.4 Transducer.
Cal	Data that may be changed when the transducer is calibrated.
Usr	Data that may be changed by the user.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

7.4.5 Datatype

The <*data_type*> parameter defines the method by which the bits read from the TEDS for this property are decoded. Allowable data types are tabulated in Table 7. More detailed explanations are provided in the following clauses.

Syntax:

<*number_of_bits*>, <*dtype*>

or

<*number_of_bits*>, <*float_type*>, <*start_value*>, <*tolerance*>

or

<*number_of_bits*>, *single*

or

<*number_of_bits*>, <*enum_type*>

The <*number_of_bits*> shall be a decimal integer.

The <*dtype*>, <*float_type*>, *single,* and <*enum_type*> shall be one of the types indicated in Table 7.

The <*start_value*> shall be a decimal number.

The <*tolerance*> shall be a decimal number.

7.4.5.1 Number_of_bits

The <*number_of_bits*> parameter shall indicate the bit length in the TEDS for the value of this data type. That is, for this data type, <*number_of_bits*> bits are read from the TEDS, interpreted according to the <*dtype*>, <*float_type*>, or <*enum_type*> parameter, and the resultant value assigned to the current property.

© IEEE 2004 – All rights reserved

Table 7—Data types

<dtype>	Description
Date	Number of the day since 1 Jan 1998.
UnInt	Unsigned Integer.
Bit_Digit	Binary digit by 2 bits.
BitBin	Array of Bit_Digits.
Chr5	Character by five bits. See 7.4.5.2.3.
ASCII	Standard 7-bit ASCII.
Unicode	International Unicode. See ISO/IEC 10646-1:2000 and *The Unicode Standard, Version 2.0* [B14].
String5	Describes a structure that consists of a string length and an array of CHR5 characters.
String7	Describes a structure that consists of a string length and an array of ASCII characters.
String16	Describes a structure that consists of a string length and an array of Unicode characters.
<float_type>	**Description**
ConRelRes	Constant Relative Resolution. Only positive values. Provides a logarithmic mapping of an interval.
ConRes	Constant Resolution. Provides a linear mapping of an interval.
Single	IEEE 754-defined 4-byte single precision floating-point number.
<enum_type>	**Description**
(Defined enumeration type)	References an enumerated data type defined by the Enumeration command. See 7.4.5.4 for details on enumeration.

7.4.5.2 Standard data and string types <dtype>

7.4.5.2.1 Date

The date data type is an integer that counts days since 1 January 1998. The first day, 1 January 1998 is day 0. All "1" in RAW TEDS (binary) shall denote property not used.

Example:

```
%CalDate, "date of calibration", CAL, 14, Date, "", ""
```

The binary input for this might be:

00000000011111

which would be translated into:

1 February 1998.

Note that 14 bits provides dates into the year 2032.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

7.4.5.2.2 UnInt

This data type indicates a direct unsigned integer translation from binary to decimal (or other base system). All "1" in RAW TEDS (binary) shall denote property not used.

Example:

```
%Gain, "Amplifier Gain", Cal, 7, UnInt, "", ""
```

The binary input for this might be:

0000010

which would be translated into decimal:

2

7.4.5.2.3 Bit_Digit

Bit_Digit is a binary digit that can contain the bitvalues "0", "1", "x" and "," (comma), as shown in Table 8.

Table 8—Bit_Digit definition

Binary value	Bit_Digit
00	0
01	1
10	x
11	,

7.4.5.2.4 BitBin

BitBin is an array of Bit_Digits.

7.4.5.2.5 Chr5

This is a character set intended to use a minimal number of bits for the characters. Table 9 provides the encoding.

Example:

```
%CalInitials, "person who calibrated", CAL, 15, Chr5, "", ""
```

The binary input for this might be:

00001 00010 00011

which would be translated into:

```
ABC
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table 9—Chr5 Values

		lsb							
		000	001	010	011	100	101	110	111
msb	00	Space	A	B	C	D	E	F	G
	01	H	I	J	K	L	M	N	O
	10	P	Q	R	S	T	U	V	W
	11	X	Y	Z	,	.	/	_	@

7.4.5.2.6 ASCII

Bits using this data type shall be interpreted using the standard 7-bit ASCII code. The <*number_of_bits*> for this data type shall be a multiple of 7.

Example:

```
%CalInitials, "person who calibrated", CAL, 21, ASCII, "", ""
```

The binary input for this might be:

1000001 1000010 1000011

which would be translated into:

```
ABC
```

7.4.5.2.7 Unicode

Bits using this data type shall be interpreted using ISO/IEC 10646-1:2000 (see also *The Unicode Standard, Version 2.0* [B14]).

The <*number_of_bits*> for this data type shall be a multiple of 16.

Example:

```
%CalInitials, "person who calibrated", CAL, 48, Unicode, "", ""
```

The binary input for this might be:

0000000001000001 0000000001000010 0000000001000011

which would be translated into:

```
ABC
```

7.4.5.2.8 StringN

The data types `String5`, `String7`, and `String16` are provided to allow variable length strings within the TEDS. That is, the property in the template will contain a number of bits that represents the bit length of a field in the TEDS that then indicates the number of characters in the string.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

When a `StringN` command is parsed, the number of bits indicated by <*number_of_bits*> is read from the TEDS. The value read indicates the number of characters in the string, which are then read from the TEDS.

Example:

```
%CalInitials, "person who calibrated", CAL, 4, String5, "", ""
```

In this example, four bits will be read from the TEDS, and the value of this 4-bit field will indicate the number of 5-bit `Chr5` characters to be read for the string. If the TEDS bits for this example were as follows:

0011 00001 00010 00011

Then, the first four bits have a value of 3; so three Chr5 characters will then be read (15 bits total) to give as a value of the string:

```
ABC
```

7.4.5.3 Floating-point types <Float_Type>

There are two data types that provide compressed floating-point values. One provides a linear mapping over an interval and the other provides a logarithmic mapping. These data types, `ConRelRes` and `ConRel`, take two parameters <*start_value*> and <*tolerance*>.

For the floating-point types, a RAW-TEDS binary value of all ones shall indicate not a number (NaN). This conforms to IEEE Std 754-1985 with respect to the Single type. When NaN is read, the value of the property is undefined and the property shall not be used.

7.4.5.3.1 Start_Value

The `Start_Value` defines the value assigned to the property when the bits read from the TEDS are all zero and is used in calculating the final value assigned to the property for other values of the TEDS bits.

7.4.5.3.2 Tolerance

Tolerance is used for the step size in the ConRes.

In the ConRelRes, tolerance shall be defined as the maximum accepted difference between a desired value and the value to be stored in the TEDS. This is used to specify the needed numerical resolution, which is two times the tolerance as stated in the formulas for ConRelRes. The values should be chosen with respect to the calibration uncertainties.

7.4.5.3.3 ConRes

The constant resolution data type assigns a value according to this formula:

property name = <*start_value*> + [<*tolerance*> x <*teds_value*>]

Where <*teds_value*> is the positive integer value of the bits read from the TEDS.

This is equivalent to linearly mapping the positive integer values in the interval [0, (2^<*number_of_bits*> – 2)] to the real interval [<*start_value*>, maximum value]. Where:

maximum value = <*start_value*> + [<*tolerance*> x ($2^{<number\_of\_bits>}$ –2)]

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Note that the maximum value is not explicitly written in the command line. All "1" in RAW TEDS (binary) shall denote NaN.

Example:

```
%TempCoef, "Temperature coefficient (b)", Cal, 9, ConRes, -0.5,
0.002,"0.000","%/°C"
```

The binary input for this might be:

000000100

which would be used to calculate:

TempCoef = –0.5 + [0.002 × 8] = –0.484

maximum value = –0.5 + [0.002 × (2^9–2)] = 0.52

Example of integers:

A <start_value> of –128 and a <tolerance> of 1 is the equivalent to a signed integer type ranging from –128 to 126. Note that the maximum value is 126 and not 127, since a binary value of all ones is defined as NaN.

Example of number of bits required:

Creating a `ConRes` type with a certain range and granularity requires calculating the number of bits necessary to store the information. Consider a floating-point data type ranging from –100 to 100 with a granularity of 10^{-5}. Then there are 10^5 numbers per integer. This means there are 2×10^7 discrete numbers in the interval (–100, 100). Thus a `ConRes` data type with <*start_value*> of –100 and a <*tolerance*> of 10^{-5} requires 25 bits.

7.4.5.3.4 ConRelRes

The constant relative resolution is calculated using the following equation.

property name = <*start_value*> * [1 + 2 * <*tolerance*>] ^ <*teds_value*>

The start value defines the value assigned to the property when the bits read from the TEDS are all zero and is used in calculating the final value assigned to the property for other values of the TEDS bits. This data type can only have positive values when <*start_value*> is positive and can only have negative values when <*start_value*> is negative. All "1" in RAW TEDS (binary) shall denote NaN.

This is equivalent to logarithmically mapping the positive integer values in the interval [0, (2^<*number_of_bits*> – 2)] to the real interval [<*start_value*>, maximum value], where the maximum value is realized when the <teds_value> is maximal.

Maximum value = <*start_value*> * {[1 + (2*<*tolerance*>)] ^ [(2^<*number_of_bits*>) – 2]}

Example:

```
%TF_KPq,"Quality factor @ f res (Q)", Cal, 7, ConRelRes, 0.3, 0.03,
"","V/(m/s_)"
```

The binary input for this might be:

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

1111110

which would be used to calculate:

TF_KPq = 0.3 * [1 + 2 * 0.03)] ^ 126 = 463.084535408

The data type in this example provides a logarithmic mapping of 126 values over the interval [0.3, 463.084535408].

7.4.5.3.5 Single

This data type is the IEEE 754-defined 4-byte single precision floating-point number.

Example:

```
%MaxPhysVal, “Max Val of the Physical Range”, ID, 32, Single, “”,
"V/(m/s2)”
```

The binary input for this might be:

1 10000001 10100000000000000000000

which would be used to calculate:

MaxPhysVal = –1 × 2 × (129–127) × 1.625 = –6.5

7.4.5.4 Enumerated types <enum_type>

Enumerated types provide an integer mapping to a set of discrete values. The enumeration is first defined in an `enumerate` command. The defined type is then used as the data type in a property command.

Syntax:

`Enumerate` <*defined_type*>, <*enum_list*>

The <*defined_type*> shall be an unquoted character string.

The <*enum_list*> is an ordered, comma-delimited list of quoted character strings or numerical values.

Reference to an item in an enumerated list may be by position as an integer or by the list item itself. An integer reference to an item in an enumerated list shall reference the first item in the list as 0. The number of bits necessary for an enumerated type is determined from the number of possible values. When using a constant value in an enumerated property (i.e., the value is not read from the TEDS), the constant value may be either the numerical value or the character value.

The `Enumerate` command shall not map any bits in the TEDS.

Example:

The following defines an enumerated data type called “mycolors.”

```
Enumerate mycolors, “blue”, “black”, “gray”, “green”
```

This enumeration assigns integer values of 0 to blue, 1 to black, 2 to gray, and 3 to green.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

This property command reads in the chosen color.

```
%MDEF_Color, “Choice of color”, USR, 2, mycolors, “”, “”
```

The binary input read from the TEDS for this command might be

```
01
```

which defines the chosen color to be black.

The equivalent property command for using constant assignment could be

```
%MDEF_Color, “Choice of color”, USR, 2, mycolors, “”, “” = black
```

7.4.6 Format parameter

The format parameter provides a recommended display format for the value of the property. Formats are always quoted strings. Formats may be described by constructing a string of specially defined characters. Table 10 lists the different format characters.

Table 10—Formatting symbols

Numeric formatting symbols	Description
. (period)	Decimal separator. The actual character displayed is dependent on the display program.
, (comma)	Thousand separator. The actual character displayed is dependent on the display program.
0	Digit placeholder. Display a digit or 0.
#	Digit placeholder. Display a digit or nothing.
$	Display the literal character $.
%	Percentage. The value is multiplied by 100 and a percent sign is appended.
E– or e–	Scientific notation with a minus sign (–) next to negative exponents and nothing next to positive exponents. This symbol shall be used with other symbols, as in 0.00E–00 or 0.00E00.
E+ or e+	Scientific notation with a minus sign next to negative exponents and a plus sign (+) next to positive exponents. This symbol shall be used with other symbols, as in 0.00E+00.
p	Displays a number using the SI prefixes identified in 7.4.6.1.
r	Relative (displays the number with a resolution corresponding to the resolution given by the Con-RelRes statement). This is generally not used by itself, but in conjunction with the “p” format. See 7.4.6.1.
rp	Relative with prefix. A combination of the preceding “r” and “p.” See 7.4.6.1.
String format	**Description**
S	Display as a standard string.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table 10—Formatting symbols *(continued)*

Date formats	Description
/	Date separator.
D	Day of the month in one or two numeric digits, as needed (1 to 31).
Dd	Day of the month in two numeric digits (01 to 31).
M	Month of the year in one or two numeric digits, as needed (1 to 12).
mm	Month of the year in two numeric digits (01 to 12).
mmm	First three letters of the month (Jan to Dec).
mmmm	Full name of the month (January to December).
yy	Last two digits of the year (01 to 99).
yyyy	Full year (0000 to 9999).
Enumerated format	**Description**
E or e	Displays the enumerated name rather than the enumerated index.

Examples:

Some examples of the numeric, date, string, and enumerated formats are given in Table 11.

Table 11—Format examples

Format	Example output
“0000.00”	0234.50
“0.00”	234.50
“####.##”	234.5
“#,##0.00”	2,345.60
“m/dd/yy”	3/26/98
“d-mmm”	26-Mar
“mmmm-yy”	January-98
“s”	This is a string.
“e”	Blue

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

7.4.6.1 Prefix and relative formats—p, r, and rp

These formats provide an easy way to indicate that the display should be scaled so as not to require lots of zeros. The “p” format indicates the use of prefixes to indicate the scale. The “r” format indicates that the resolution displayed should be based on the resolution of the number being displayed. This is especially

© IEEE 2004 – All rights reserved

useful when using the ConRel or ConRelRes data types since they are specifically designed to provide scaled numbers.

The “p” format is generally used with other numeric formats. That is, the “p” format is either used with format characters that indicate the number of digits to be displayed on either side of the decimal separator, or the “p” format is used in conjunction with the “r” format. When the “p” format is present, the appropriate prefix should be inserted into the display.

The “r” format is generally not used by itself. The “p” format display shall use the prefixes in Table 12. Note that these are consistent with the International System of Units (SI).

Table 12—Numeric prefixes

Factor	Name	Symbol	Factor	Name	Symbol
10^{24}	Yotta	Y	10^{-3}	milli	m
10^{21}	Zetta	Z	10^{-6}	micro	μ
10^{18}	Exa	E	10^{-9}	nano	n
10^{15}	Peta	P	10^{-12}	pico	p
10^{12}	Tera	T	10^{-15}	femto	f
10^{9}	Giga	G	10^{-18}	atto	a
10^{6}	Mega	M	10^{-21}	zepto	z
10^{3}	Kilo	k	10^{-24}	yocto	y

Examples:

Given the following template command:

```
%Sens@Ref, "Sensitivity@ref.", Cal, 16, ConRelRes, 5E-7, 0.00015,
"rp", “V/(m/s_)"
```

the values would be displayed as indicated in Table 13.

Table 13—Example relative and prefix formats (high resolution)

Actual value in the range		Will be stored in memory as	Resulting value	Displayed with format “rp”	Displayed with format “0.00p”
5,00225E–7	5,00375E–7	2	5,00300E–7	500,30n	500,30n
9,99508E–4	9,99808E–4	25339	9,99658E–4	999,7μ	999,66μ
9,99808E–4	1,00011E–3	25340	9,99957E–4	1000,0μ	999,96μ

A similar template command with lower resolution:

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
%Sens@Ref, "Sensitivity@ref.",  Cal, 14, ConRelRes, 5E-7, 0.015,
"rp",  "V/(m/s_)"
```

would display the values as indicated in Table 14.

Table 14—Example relative and prefix formats (low resolution)

Actual value in the range		Will be stored in memory as	Resulting value	Displayed with format "rp"	Displayed with format "0,00p"
5,22668E–7	5,38348E–7	2	5,30450E–7	530n	530,45n
9,55398E–3	9,84060E–3	334	9,69623E–3	9,7m	9,70m
9,84060E–3	1,01358E–2	335	9,98712E–3	10,0m	9,99m
3,38670E6	3,48830E6	1000	3,43712E6	3,44M	3,44M

7.4.7 Physical units

The `physical units` parameter associates a text string (the *unit_name*) with a definition of a physical dimension and units. The *unit_name* string is then used in the property commands to assign physical units to parameter values. The association of a *unit_name* with a definition of units is local to a template.

The physical dimension is defined by assigning an exponential value for each of the base SI units, including radian and steradian. The base SI units are assumed, but non-SI units shall be expressed through the use of offset and span scale factors. Finally, the units may be defined to be either the defined dimension, a ratio of values of that dimension, or a logarithm of that dimension.

This mechanism allows arbitrary units to be defined, and further completely defines the units such that they could be converted back to base SI units by application software.

7.4.7.1 Physical unit to character string association

The mechanism for associating character string representations of physical units with a definition of the units is provided through the `Physical_Unit` command. These associations are local to the current template and are not intended to indicate associations in other templates. The `Physical_Unit` command is used for application purposes only. This command shall not map any bits in the TEDS.

Syntax:

`Physical_Unit` <"*unit_name*">, <*unit_definition*>

The <"*unit_name*"> shall be a quoted character string which is to be associated with the units defined by the <*unit_definition*> field for the remainder of the template.

The <*unit_definition*> shall be a parenthesis-enclosed set of 12 elements, which define the units, as described in 7.4.7.2

7.4.7.2 Unit definition

The unit definition field consists of 12 elements. This field is written in the template in the following form:

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

(enumeration, exp1, exp2, exp3, exp4, exp5, exp6, exp7, exp8, exp9, scale, offset)

Table 15 describes how each of these fields shall be used. The exponents shall be decimal numbers in increments of 1/2 between –64 and 63 inclusive. (This is thus compatible with other IEEE 1451 standards.). The nine exponent fields define a unit U according to the following equation.

$U = radians^{exp1} \cdot steradians^{exp2} \cdot meters^{exp3} \cdot kilograms^{exp4} \cdot seconds^{exp5} \cdot amperes^{exp6} \cdot kelvin^{exp7} \cdot moles^{exp8} \cdot candelas^{exp9}$

The first field indicates how this dimension U shall be used in the definition of the unit. The various options are listed in Table 15. The unit is further defined through the use of scaling and offset factors (the last two fields) to allow for unit systems other than the SI system of units.

The U form (enumeration zero) simply uses the dimension U, modified only by the scaling and offset factors.

The U/U form (enumeration one) expresses ratios of units such as strain (meters per meter).

The $\log_{10}(U/U_0)$ forms (enumerations two and five) express logarithmic values of units, such as pH (log of concentration). The U_0 is assumed to have a value of 1.0 in the dimensions defined by U.

The $\log_{10}(U/U)$ form (enumerations three and six) allows for logarithmic ratios of values, such as a decade of frequency (log of the ratio of two frequencies).

Enumeration four is used for bit vectors (digital data), which have no units, as they do not directly represent physical quantities.

Table 15—Physical units data type structure

Element no.	Description
1 Enumeration Field	0: Unit is described as m·(U+b) where U is defined by the next 9 fields, m is a dimensionless scaling factor in field 11, and b is an offset in field 12 taken to have units of U.
	1: Unit is the ratio m·(U/U +b), where U is defined by the next 9 fields, m is a dimensionless scaling factor in field 11, and b is an offset in field 12 taken to have units of U/U.
	2: Unit is m·($\log_{10}$(U/U_0)+b), where U is defined by the next 9 fields, U_0 is a reference quantity having a magnitude of exactly 1.0 with units of U, and m and b are dimensionless scaling factors in fields 11 and 12.
	3: Unit is m·($\log_{10}$(U/U)+b), where the next 9 fields define U, and m and b are dimensionless scaling factors in fields 11 and 12.
	4: The data is a bit vector (digital data) and has no unit.
	5: Unit is m·($\log_{10}^{-1}$(U/U_0)+b), where U is defined by the next 9 fields, U_0 is a reference quantity having a magnitude of exactly 1.0 with units of U, and m and b are dimensionless scaling factors in fields 11 and 12.
	6: Unit is m·($\log_{10}^{-1}$(U/U)+b), where the next 9 fields define U, and m and b are dimensionless scaling factors in fields 11 and 12.
	7–255: Reserved
2	Exponent of radians

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table 15—Physical units data type structure *(continued)*

Element no.	Description
3	Exponent of steradians
4	Exponent of meters
5	Exponent of kilograms
6	Exponent of seconds
7	Exponent of amperes
8	Exponent of kelvins
9	Exponent of moles
10	Exponent of candelas
11	Scaling factor m
12	Offset factor b

Examples:

The command

```
Physical_Unit “Hz”, (0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 1, 0)
```

assigns the string “Hz” to a physical unit definition. The exponent fields define a base SI unit U as

U = radians^{0}·steradians^{0}·meters^{0}·kilograms^{0}·seconds^{-1}·amperes^{0}·kelvin^{0}·moles^{0}· candelas^{0}

which simplifies to U = seconds^{-1}.

The enumeration field of 0, the scaling factor 1, and the offset factor of 0 define the units as

“Hz” = 1.0·(U + 0 seconds^{-1})

Other examples are as follows:

```
Physical_Unit “mm”, (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0.001, 0)
// “mm” millimeter = 0.001 m
Physical_Unit “°C”, (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1.0, -273.15)
// “°C” Celsius is (kelvin - 273.15 K)
Physical_Unit “°F”, (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1.8, -
255.372222)
// “°F” Fahrenheit is 1.8 · (kelvin — 255.3722 kelvin)
Physical_Unit “ppm/°C”, (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1e-6, 0)
// “ppm/°C” 10-6 · (kelvin^-1)
Physical_Unit “microstrain”, (1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1e-6,
0)
// “microstrain” 10-6 · (meters/meters)
Physical_Unit “pH”, (2, 0, 0, -3, 0, 0, 0, 0, 1, 0, -1.0, -3)
// “pH” -1.0 · (log10[mole/meter^3] - 3) = -log10(10^-3 · mole/meter^3)
= -log10(mole/liter)
Physical_Unit “%/decade”, (6, 0, 0, 0, 0, -1, 0, 0, 0, 0, 100, 0)
// “%/decade” 100 / (log10[second^-1/second^-1] + 0) = 100 /
log10(Hz/Hz)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

7.4.8 Optional assignment

Instead of reading a property value from the TEDS, it is possible to assign a value directly in the template. This is done by appending an equals sign, "=", and a value after the property parameters. The syntax for this is shown in 7.4.1. The value shall be consistent with the specified data type. The value may be specified as any of the following:

- Decimal integer or number
- Binary number using the "0b" prefix
- Hexadecimal number using the "0x" prefix
- Quoted string

Regardless of the number of bits field, a property command using the optional assignment shall not read any bits from the TEDS.

Example:

This property command could be in a template.

```
%CalInitials, "person who calibrated", CAL, 21, ASCII, "s", "" =
"CHJ"
```

7.4.9 Extended Functionality using subproperties

Extended Functionality provides a mechanism to specify information about functionality of a transducer that extends beyond simple measurement or actuation. This is accomplished through the use of subproperties, which are modifiers to properties that define how to implement a given Control Function. Most often these functions manipulate switches in the transducer and enable or disable associated properties. Subproperties are syntactically indicated by square brackets, [ ], after the property tag. (See 7.4 for syntax.) A Control Function is defined by a set of subproperties and shall be initiated by the subproperty [Initialize] and shall include subproperty [CtrlFunctionMask]. Any property may be modified with subproperties. There can be more than one Control Function with the same property tag.

Modifying a property with a subproperty is semantically equivalent to creating another property that is uniquely identified by the combination of both the property and subproperty. The value of this modified property can thus be read from the TEDS or assigned directly in the template. Further, multiple occurrences are treated just as multiple occurrences of properties are treated (see 7.4.13).

7.4.9.1 Definitions

The following are definitions used in 7.4.9.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

7.4.9.1.1 Bit-wise operators to be used with arrays of datatype Bit_Digit

XOR	0	1	x
0	0	1	0
1	1	0	0
x	0	0	0

XNOR	0	1	x
0	1	0	1
1	0	1	1
X	1	1	1

SET	0	1	x
0	1	1	0
1	1	1	0
x	0	0	0

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

7.4.9.1.2 Function bits

Function bits are the bits in the special page (see Annex F and "1-Wire Master Device Configuration" [B1]) in a device with the Family Code "Generic Memory" and for which the function of these bits are defined in the template for the transducer.

7.4.9.1.3 Function Register (FR)

The FR is a binary register of length equal to the total number of bits associated with hardware switches or function bits defined by Family Code and ranked by the IEEE 1451.4 Transducer Node-List. The register shall contain the current state of all entities in the IEEE 1451.4 Transducer. The FR shall be of datatype `BitBin` valid for only "1" and "0." The left most `Bit_Digit` of the register shall correspond to the value of the first function entity (FR1) in the system. For example, an FR with a value of "1100" shall represent a system of four FR entities where FR1 and FR2 have a value of "1." The remaining FR entities (FR3 and FR4) have a value of "0."

© IEEE 2004 – All rights reserved

7.4.9.1.4 Control Property

A Control Property directly controls one or more FR entities in the underlying hardware. A Control Property has the subproperty [Function]. Control Properties are associated to FR entities by their property values. The property value is of datatype BitBin defining the FR setting that applies to a given function state. It can be a list of two values when the function is of [FunctionType] 0, defining a "checkmark." A property value specifies all the switches, also the ones not controlled. These are indicated by an "x."

For example, in a system with Control Property passive and Control Property values given by

State 1: "xx0x"

State 2: "0010"

FR values other than the ones defined by state 1 and state 2 shall not be settable by Control Property passive.

A Control Property is selected when a bitwise XNOR between its value and the FR results in "1" for all Bit_Digits. This implies that Function Property shall be uniquely defined by the FR with respect to the [FunctionType] (see Table 16).

When the user changes the setting of a Control Function, the FR is bitwise assigned the Control Property values except were these are "x" and the following step has to be taken for all the Control Functions defined later in the template:

1) A Control Property shall be disabled when a bitwise binary XOR between any previous selected Control Properties value and this Control Property value results in a "1" for any Bit_Digits. If the Control Function only has one state enabled, the Control Function shall be disabled as a whole.
2) A Control Property shall be set when a bitwise XNOR between its value and the FR results in "1" for all Bit_Digits; otherwise, it shall not be set.

If any changes have been applied to the FR, the underlying hardware has to be updated accordingly.

NOTE—For the special case of a function subproperty, the description field in the property command can be associated with the description field and the function value of another property command. As shown in the example, the description field can be replaced with a property tag and associated subproperty "Register mask." This indicates that the description field of the associated property concatenated with the proper formatted value read from the associated property command is to be used in the user display. The following property command example would lead to the display text shown in the example.

```
%Sens[function], %Sens@Ref[“100x”],  USR, 8, BitBin, “”,”” = “1000”
%Sens[function], %Sens@Ref[“010x”],  USR, 8, BitBin, “”,”” = “0100”

%Sens@Ref [“100x”], “Sensitivity @ F ref”, CAL, 16, ConRelRes, 100E-6,
0.0001, “rp”, “V/Pa”
%Sens@Ref [“010x”], “Sensitivity @ F ref”, CAL, 16, ConRelRes, 100E-6,
0.0001, “rp”, “V/Pa”
```

This will result in a Control Function state text such as the following (depending on the values in the TEDS):

o Sensitivity @ F ref 200mV/Pa
X Sensitivity @ F ref 20mV/Pa

7.4.9.1.5 Property associated with FR

A property can be dependent upon the current value of the FR. It is associated to this through a Register mask. The Register mask is of datatype BitBin with a length equal to the length of the FR. The Register mask specifies when this property shall be enabled. It can contain wildcards indicated by an "x." These properties are syntactically indicated by square brackets, [ ], after the property tag as subproperties (see 7.4.9). The property shall be enabled when a bitwise XNOR between the Register mask and the FR results in

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

"1" for all `Bit_Digits`, otherwise disabled. If a comma-separated list is applied, it shall be enabled if one Register mask in the list applies. Associated properties shall be updated when a change in the FR is detected by system specific means. When a property is disabled, it shall not be used in any way.

For example, a system with associated property

```
%gain["100x,0x00"], "Gain", Cal, 12, ConRes, -40, 0.025, "0.0p",
"dB"
```

will enable the associated property for FR values: 1000, 1001, 0000, 0100.

7.4.9.2 Subproperty list

The list of valid subproperties is provided in Table 16.

Table 16—Subproperties

Subproperty Accessor	Description
`Default`	The default value for the function. (Not the default value of the property.) This is either the value of the function upon start-up or, if the function must be initialized, the value to initialize the function to. The `Default` value is of datatype `BitBin`. Shall not be present if `%DefaultFR` is defined. If not present and default is not defined by `%DefaultFR`, then Default = 0.
`Initialize`	A single bit that indicates whether this function shall be initialized upon start-up. It also initiates a new Control Function. The `Initialize` value is of datatype `UNINT`. 0: Do not initialize 1: Initialize Shall be present.
`CtrlFunctionMask`	The mask on the FR indicating which bits of the FR are associated with the Control Function. The `CtrlFunctionMask` value is of datatype `BitBin`. The mask shall be a bit-wise SET of all the Control Property values defined in the Control Function. Shall be present.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table 16—Subproperties *(continued)*

Subproperty Accessor	Description
`ReadWrite`	Two bits indicating what level of programmability the function has. The ability to read means that the current setting can be electronically read from the transducer. The ability to write means that the setting can be changed electronically. The `ReadWrite` value is of datatype `UNINT`. 0: Read and Write 1: Can not read or write (physical switch only) 2: Read only 3: Write only If not present, then assume can read and write.
`FunctionType`	Indicates how many options can be, or shall be, selected at one time. (The widget for displaying these function types is left to the application program.) The `FunctionType` value is of datatype `UNINT`. 0: Can have either 0 or 1 settings set. 1: Shall have exactly 1 setting set. 2: Can have any number of settings set including none. 3: Can have any number of settings set but shall have at least one. If not present, then assume there shall be exactly 1 setting set.
`Function`	Controls the behavior of the Extended Functionality. A Control Property has the subproperty [`Function`]. Control Properties are associated to FR entities by their property values. The property value is of datatype `BitBin` defining the FR setting that applies to a given function state. For details, see 7.4.9.1.4.
`"Register mask"`	This is the one subproperty where the subproperty name, “Register mask,” is not actually used in a template. Instead, a Register mask, or a list of register masks is provided in the form of a value of datatype BitBin. The property shall be enabled when a bitwise XNOR between the Registermask and the FR results in “1” for all Bit_Digits, otherwise disabled. If a list is applied, it shall be enabled if one Register mask in the list applies.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Example:

The following is an extensive example of the ability to define Extended Functionality. It continues through the end of this subclause.

Consider the system described in Figure 8. It has two stages: a filter with gain and a multiplexer. The following states grouped in three functions are defined at design time:

The gain:

Gain 0dB (default value)

© IEEE 2004 – All rights reserved

Gain 10 db (amp 1: +10db, amp2: +0dB)

Gain 20 dB (amp 1: +10db, amp2: +10dB)

The filter:

Filter OFF (default value)

Filter ON (only valid for gain>0db)

The multiplexer:

Passive-off (default value)

Passive-on

In the case of +20 dB gain, a bandwidth limiting occurs; this results in a low-pass filter.

The amplifier and the filter are still in valid states when the device is in passive mode.

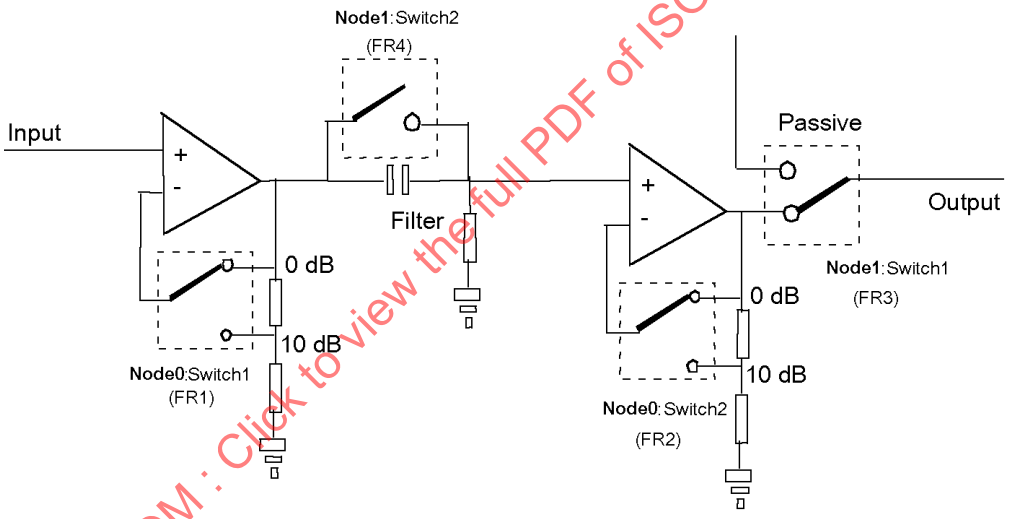


Figure 8—Filter with gain and multiplexer system

A possible node configuration to control the amplifier with filter in Figure 8 could be described by Figure 9. A Switch is defined as a hardware device that can have values 0 and 1. Switches are functional components of a hardware node such as DS2406/DS2407 and are fully described by the IEEE 1451.4 XML device definition files. PIO-A and PIO-B in a DS2406/2407 are examples of switches. FR is defined as an abstraction of a hardware switch. The TDL deals only with FR entities.

NOTE—Node0 includes a Node-List in its memory map containing the nodes belonging to the amplifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

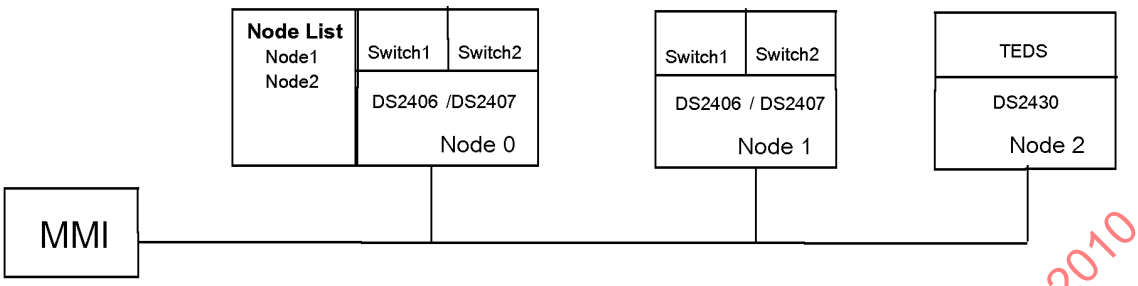


Figure 9—Filter with gain and multiplexer node configuration

A gain function defines FR1, FR2 as the FRs associated with the amplifier gains of 0 dB, 10 dB and 20 dB.

A filter function defines FR4 as the FR associated with the filter.

A passive function associated with FR3 defines if the multiplexer is in passive on or off state. The multiplexer control is associated to the passive property defined in the following template.

A TDL template describing the amplifier in Figure 8 could be

```
// function control definitions
%passive[Initialize], “Always initialize this”, ID, 1, UnInt, “”,
“” = 1
%passive[CtrlFunctionMask], “Control Function Mask”,ID, 8, BitBin,
“”, “” = “1111”
%passive[ReadWrite], “Write only”, ID, 2, UnInt, “”, “” = 3
%passive[Default], “Default not passive”, ID, 8, BitBin, “”, “” =
“xx0x”
%passive[FunctionType], “passive control type”, ID, 2, unint,
“”,“” = 0//checkmark
%passive[Function], "Passive mode", USR, 10, BitBin, “”,“” =
“xx0x,0010”

%gain[Initialize], “Always initialize this”, ID, 1, UnInt, “”, “”
= 1
%gain[CtrlFunctionMask], “Control Function Mask”,ID, 8, BitBin,
“”, “” = “1101”
%gain[ReadWrite], “Write only”, ID, 2, UnInt, “”, “” = 3
%gain[Default], “Default gain: 0dB”, ID, 8, BitBin, “”,“” = “00x0”
%gain[FunctionType], “Gain control type”, ID, 2, unint, “”, “” =
1//One Exactly

%gain[Function], “This is 20dB”, USR, 8, BitBin, “”,“” = “11xx”
//State1
%gain[Function], "This is 10dB", USR, 8, BitBin, “”,“” = “01xx”
//State2
%gain[Function], "This is  0dB", USR, 8, BitBin, “”,“” = “00x0”
//State3

%filter[Initialize], “Always initialize this”, ID, 1, UnInt, “”,
“” = 1
%filter[CtrlFunctionMask], “Control Function Mask”,ID, 8, BitBin,
“”, “” = “0001”
%filter[ReadWrite], “Write only”, ID, 2, UnInt, “”, “” = 3
%filter[Default], “Default disabled”, ID, 8, BitBin, “”, “” =
“xxx0”
%filter[FunctionType], “Control type”, ID, 2, unint, “”,“” =
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
0//checkmark
%filter[Function], "HP filter", USR, 10, BitBin, "","" =
“xxx0,xxx1”
%Passive, "Supports multiplexer via no gain selected", Cal, 1,
UNINT, "",""
%Gain[“1x0x”], "This is stage 1. 10dB", Cal, 12, ConRes, -40,
0.025, "0.000p", "dB"
%Gain[“0x0x”], "This is stage 1. 0dB", Cal, 12, ConRes, -40,
0.025, "0.000p", "dB" = 0
%Gain[“x10x”], "This is stage 2. 10dB", Cal, 12, ConRes, -40,
0.025, "0.000p", "dB"
%Gain[“x00x”], "This is stage 2. 0dB", Cal, 12, ConRes, -40,
0.025, "0.000p", "dB" = 0
%TF_HP_S[“xx00”], "High Pass filter off", Cal, 1, unint,
"0.000p", "Hz" = 0
%TF_HP_S[“xx01”], "High Pass filter on (only valid for
gain>0dB)", Cal, 12, ConRelRes, 0.01, 0.001,"0.000p", "Hz"
%TF_SP[“110x”], "Bandwidth limiting at +20dB", Cal, 5,
ConRelRes, 10, 0.01,"0.000p", "Hz"
```

Alternatively to the default subproperty, the default can be defined by the %DefaultFR property in which case the whole FR setup must be defined. Defining this in the device is more compact than using the subproperty because that it can be done without the “wildcard.”

For example

```
%DefaultFR, "Default Function setting", Cal, 4, UNINT, "",""
```

End Example.

7.4.10 Some special properties

These properties require extra information to use them properly.

7.4.10.1 Manufacturer-defined properties, %MDEF

Manufacturers or other users may define their own properties by using the prefix “%MDEF.” That is, a manufacturer may create a property by appending an unquoted character string to the prefix “%MDEF.” This allows manufacturer specific extensions to the property list. These properties will not be universally recognized by template parsers, and thus should only be used when truly necessary. Template parsers shall be able to parse these properties and read the indicated number of bits from the TEDS. If the parser does not recognize the property, it may discard or display the value as it so chooses.

Example:

A property might be defined as such

```
%MDEF_myprop, “Special manufacturer property”, USR, 6, UNINT, “”, “”
```

7.4.10.2 Appended TEDS properties

An Appended TEDS is a TEDS not located on the transducer. It provides a mechanism for defining properties without memory size constraints.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

7.4.10.2.1 %Appended_TEDS

This property indicates if an Appended TEDS exists for this transducer. The TEDS shall be in the Embedded Template format (see 7.4.11).

The following is the file naming convention for the Appended TEDS. The use of "+" here indicates string concatenation.

```
"IEEE1451_4_" + Manufacturer Code + "_" + Model Number + "_" + Version
Letter + "_" + Serial Number + ".ted"
```

If the Appended TEDS describes the product data, or data that is not specific to a particular serial number, then the serial number in the filenames is substituted with "PD."

```
"IEEE1451_4_" + Manufacturer Code + "_" + Model Number + "_" + Version
Letter + "_" + "PD.ted"
```

Example:

This property command could be in a TEDS.

```
%Appended_TEDS, "Appended TEDS", USR, 1, YesNo, "", ""
```

If the bit read indicates that an Appended TEDS exists, then the associated filename might be:

```
"IEEE1451_4_17_4394_A_0_3235.ted"
```

where

```
ManufacturerID = 17
Model Number   = 4394
Version Letter = a
Version Number = 0
Serial Number  = 3235
```

If Appended TEDS contains general product information, then the filename would be:

```
"IEEE1451_4_17_4394_A_0_PD.ted"
```

7.4.10.2.2 %Appended_TEDS_location

This property provides a location for the Appended TEDS if it exists. Most likely this would read in an URL or a directory path.

Example:

This property command could be in a TEDS.

```
%Appended_TEDS_location,  "Appended  TEDS  location",  USR,  5,
String5, "", ""
```

This might read from the TEDS something like:

```
www.ieee1451.org
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

7.4.11 Embedded Template format

An Embedded Template is a self-describing TEDS that include the template description.

That is, the TEDS contains legal TDL commands, where all properties are assigned values explicitly. This provides a self-contained portion of the TEDS not requiring a predefined template.

Each line in this format is carriage return line feed terminated.

The following rules shall be adhered to for data in this section of the TEDS:

1) Every property or command requiring data shall use the optional assignment structure (since there is no TEDS to be read).
2) The following commands shall not be used:
   a) Template and EndTemplate
   b) TDL_Version_Number
   c) UGID
   d) Comments
   e) Abstract
   f) SelectCase and Case
   g) StructArray
   h) Align
3) The following properties shall not be used:
   a) %user
   b) %xml
   c) %EmbTpl

7.4.11.1 %EMBTPL

The `%EMBTPL` property allows a section of the TEDS to be in Embedded Template format. In the syntax of the `%EMBTPL` command, if the <*number_of_bits*> for the <*data_type*> is 0, then the rest of the TEDS is to be read as ASCII TDL commands. Otherwise, the <*number_of_bits*> can indicate the number of bits allocated for the Embedded Template. The Embedded Template shall be zero terminated if not all allocated memory is used.

Example:

This example indicates that the remaining portion of the TEDS is in the Embedded Template format.

```
%EMBTPL, "TDL Commands", Usr, 0, ASCII, “”, “”
```

7.4.12 Free-form TEDS properties

Several properties allow a portion of the TEDS to be in a form not strictly defined by commands in a template. These are provided for flexibility, but should be used with caution since not all parsers will be able to support or interpret data provided in this fashion. Further, these forms of storing data in TEDS generally use much more memory.

WARNING
The data in these free-form areas are user defined and are utilized at the user’s own risk!

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

7.4.12.1 %user

The %user property allows a section of the TEDS to be in a user-defined format. If the bit length of the data type is 0, then the rest of the TEDS is in the user-defined format. Otherwise, the bit length indicates the total length of the user-defined field.

Example:

This example indicates that the remaining portion of the TEDS is in a user-defined format of 7-bit ASCII characters.

```
%user, "User data (ASCII 7-bit)",  Usr, 0, ASCII, “”, “”
```

7.4.12.2 %XML

The %XML property allows a section of the TEDS to be in ASCII-based XML form. Note that use of this requires an external XML DTD or Schema. In the syntax of the %XML command, if the <*number_of_bits*> for the <*data_type*> is 0, then the rest of the TEDS is to be read as ASCII XML. Otherwise, the <*number_of_bits*> can indicate the actual total number of bits to be read. Care must be taken in this latter case as the size of an XML file is only determined after completely writing it.

Example:

This example indicates that the remaining portion of the TEDS is in a user-defined XML format.

```
%XML, "XML Commands",  Usr, 0, ASCII, “”, “”
```

7.4.13 Multiple occurrences of the same property

When a property is included in a template more than once, then it shall be assumed that the property requires an array of data. Thus, values of multiple occurrences of the same property are stored sequentially in an array. This is true of properties modified by subproperties as well. That is, a modified property is considered a unique property unto itself and may require an array for storage of its associated values. It is also important that order be maintained. That is, the array should maintain the same order of the properties as they are found in the template. If a stronger association of properties is required, it is recommended that they be grouped using the StructArray command (see 7.3.4).

Note that identically tagged properties may occur in different, but concatenated, templates. Such properties shall be treated separately and associated with the specific template.

7.4.14 Handling of unrecognized properties

For a variety of reasons, a TDL parser may not recognize a property. Some reasons include:

1) New properties have been added since the parser was developed.
2) The property is manufacturer defined (`%MDEF`).
3) The parser does not support all properties.

When this happens, if the TDL parser is serving another application, then the parser should transfer the property tag and the associated value read from the TEDS to that application. Any use of that information is for the application to determine. A likely scenario is that the application simply displays that information for the user’s benefit.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

8. Mixed Mode Transducer Interface (MMI) specification

8.1 Introduction

The IEEE 1451.4 MMI ensures the robust transfer of a transducer signal and digital TEDS data between a transducer and an NCAP or data acquisition system.

8.1.1 Definition of the MMI

The IEEE 1451.4 MMI shall be the connection, for both analog signals and digital TEDS data, between a transducer conforming to this standard and an NCAP or data acquisition system. Protocols, timing, and electrical specifications in this standard shall define two classes of MMI, allowing analog and digital signals either to sequentially share a single connection or to be available simultaneously on separate connections.

For this standard, IEEE Std 1451.4-2004, a connection shall be defined as two conductors.

The TEDS memory resides in a two-leaded device, also referred to as a node, adhering to a serial, multidrop Data Transmission Protocol described in 5.5 and 8.5, which allows power and data to share one lead, while the other lead is used as return. Nodes shall use open-drain line drivers and be powered parasitically from the data line. Successful implementation of the MMI requires that users of this standard observe this protocol, which is explained fully in 5.5, 8.3, and 8.5. The Data Transmission Protocol used in this standard has been adopted from a proprietary protocol described in "DS2430A 256-bit 1-Wire EEPROM" [B3], "1-Wire Master Device Configuration" [B1], Smiczek et al. [B12], Annex E, and Annex G.

8.1.2 Operation of the MMI

A Class 1 MMI shall define the transmission of either a transducer signal or digital TEDS data, sequentially, on a single connection. Class 1 shall allow a transducer signal and power, or digital TEDS data and power, to make alternate use of the same connection. In a Class 1 MMI, digital logic levels shall be defined as 0 V = logic state 0, and –5 V = logic state 1. In general, the Class 1 MMI uses inverted logic and has a shared data connection.

A Class 2 MMI shall use separate connections for transducer signals and digital TEDS data. Class 2 shall allow simultaneous access to a transducer signal and digital TEDS data, and Class 2 connections may share a common return. In a Class 2 MMI, digital logic levels shall be defined as 0 V = logic 0, and +5 V = logic 1. In general, the Class 2 MMI uses positive logic and has a separate data connection.

8.1.2.1 Class 1

A Class 1 MMI shall sequentially transfer either a transducer signal or digital TEDS data, utilizing one Signal wire and one Return, and may also be referred to as a 2-conductor bus interface. An interface similarly implemented in either the signal circuit of a constant-current or constant voltage (current loop) 2-conductor transducer, or in the positive (+) power conductor of a transducer having a separate power conductor, or having a separate data line and sharing a return conductor, shall be known as a Class 1 MMI. Typical realizations of the Class 1 MMI are described in Figure 10, Figure 11, and Figure 12.

The block diagram, Figure 10, illustrates a typical constant-current powered sensor containing an internal amplifier. The analog signal voltage exists on the same wire carrying power current to the amplifier. The memory device contains the TEDS. Two current-steering diodes, or controlled analog switches, allow sequential access to either the amplifier or the TEDS memory, by reversing the polarity of the power current or by selecting the alternate switch, respectively. Analog switches allow the diode voltage drop to be avoided. The positive current source, in the data acquisition system, supplies power current to the amplifier, through the signal connection and the upper diode, when the switch is in the analog position. Logic power

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

flows through the lower diode, to operate the memory device, when the control switch is in the digital position, supplied by the current-limited logic supply (see 8.5.3).

Because the memory device requires that the logic 0 state be asserted with certain constraints on fall time, which cannot be met with the steering diode isolating the memory from the data line driver, a terminating pulldown resistor (R_t) shall be included in the circuit, when diode switching is utilized. The resistor discharges the capacitance of the memory circuit and wiring, insuring that the logic 0 voltage is met within the proper timing requirements. The nature of the data protocol and its ability to be used with inverted power allow analog and digital functions to share a conductor via a change of polarity.

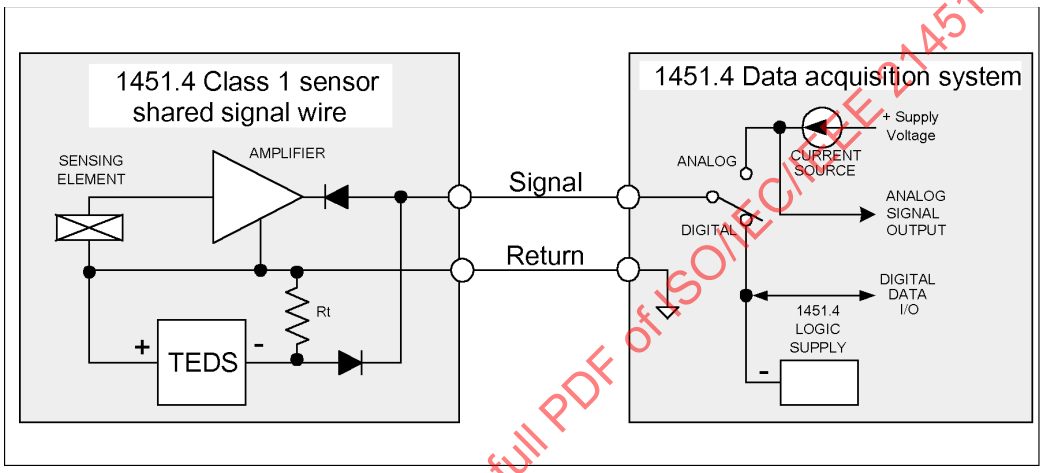


Figure 10—IEEE 1451.4 Class 1 MMI, shared signal wire

A Class 1 MMI implemented in the positive power-supply connection of a transducer with separate power and signal connections is illustrated in Figure 11.

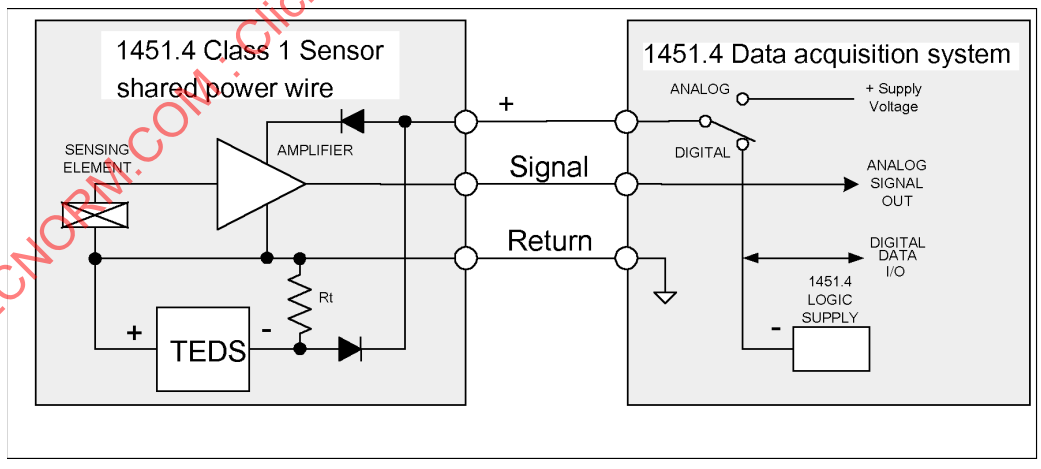


Figure 11—IEEE 1451.4 Class 1 MMI, shared power wire

Figure 12 illustrates a transducer configuration with a shared analog and digital return, typically a ground return connection, or shield. Precautions must be taken to prevent mutual interference between analog signals and digital TEDS data due to voltage drop on the shared return, so Analog Mode disables the digital data with a switch to minimize noise.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

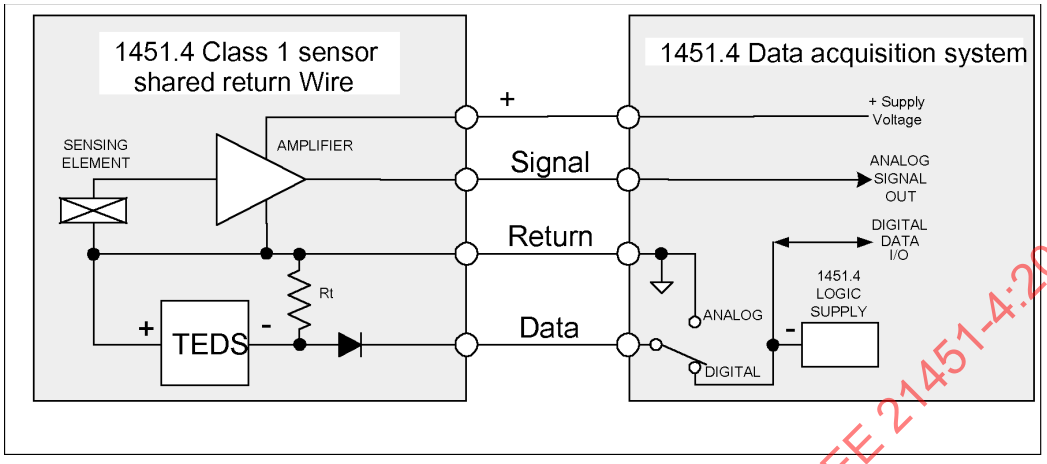


Figure 12—IEEE 1451.4 Class 1 MMI, shared return wire

8.1.2.2 Class 2

The IEEE 1451.4 Class 2 interface shall use separate connections to transfer transducer signals and digital TEDS data. With separate connections, signal and data access may occur simultaneously. The Class 2 digital data connection shall use positive logic levels, defined as 0 V = logic 0 and +5 V = logic 1, and may use a switching diode and terminating resistor for polarity protection. The separate connections of Class 2 allow data transmission with transducers not adapted to a shared, two-wire connection.

All logic timing for the Class 2 digital interface shall be as described in 8.5 and shall be identical to the timing of Class 1.

Typical IEEE 1451.4 Class 2 interfaces are illustrated in Figure 13 and Figure 14. Figure 13 depicts TEDS used in a 4–20 mA control loop application with a current-to-pressure pneumatic valve. Figure 14 illustrates the application of TEDS to a resistive bridge sensor.

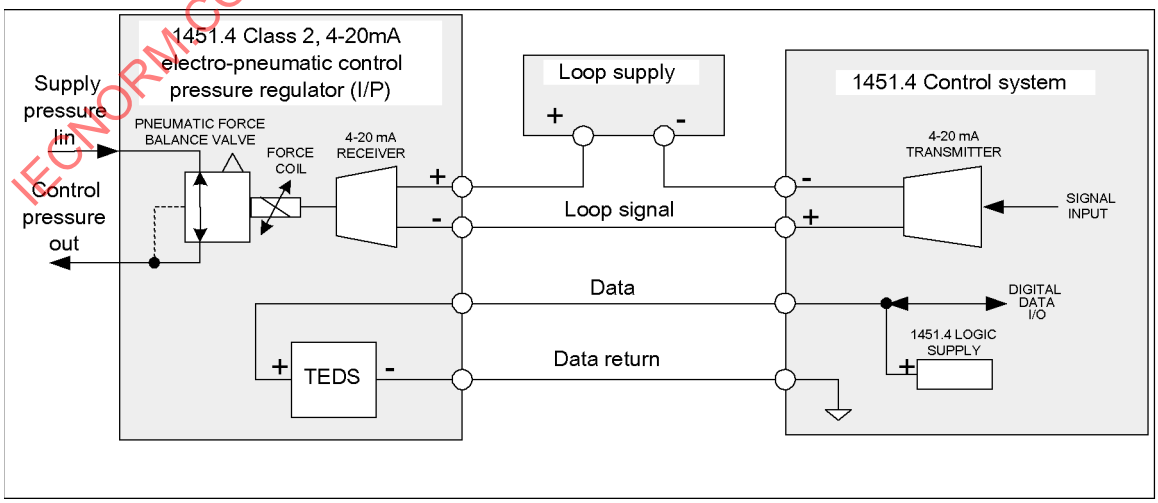


Figure 13—IEEE 1451.4 Class 2 MMI, 4-20 mA actuator

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

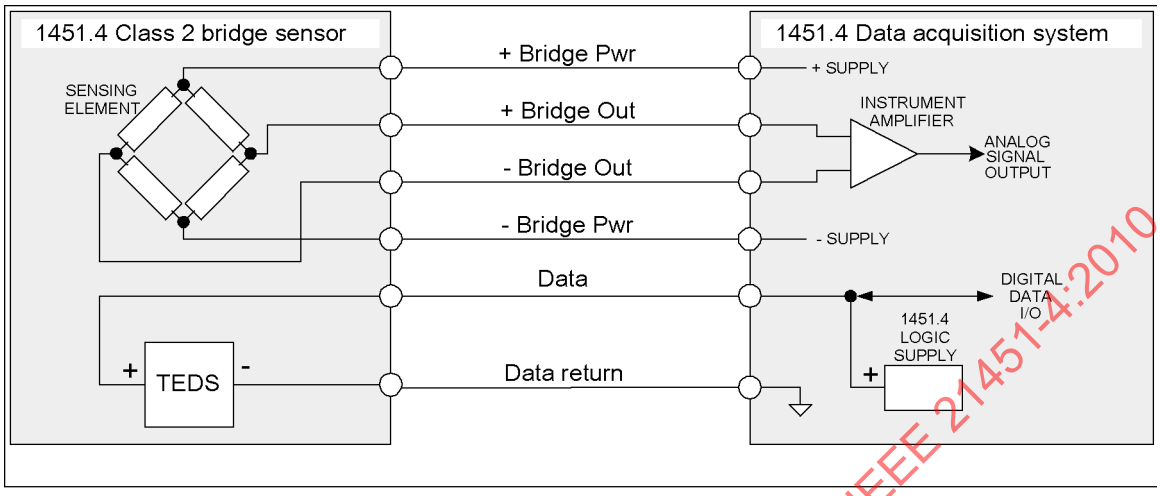


Figure 14—IEEE 1451.4 Class 2 MMI, resistive bridge sensor

8.2 Analog Mode

8.2.1 Class 1

The operating mode of an IEEE 1451.4 Class 1 transducer shall be controlled by the polarity of the power-supply voltage or current delivered to the transducer. The Class 1 transducer shall be in the Analog Mode for voltages positive with respect to the return on the signal wire of a constant-current powered transducer, or on a separate supply wire.

Figure 10 illustrates a Class 1 constant-current sensor containing an internal amplifier. The analog signal voltage occupies the same wire carrying constant-current power to the amplifier. Two current-steering diodes allow sequential access to either the amplifier or the TEDS memory by reversing the polarity of the power current. The positive current source in the 1451.4 data acquisition system supplies positive constant-current power to the amplifier through the signal connection and the upper diode, when the switch is in the analog position.

The Class 1 Voltage Powered Transducer illustrated in Figure 11 similarly switches to Analog Mode when the data acquisition system switches to positive supply power.

The Class 1 System shown in Figure 12 maintains a full-time connection to the analog amplifier. The digital TEDS data line is connected to allow digital access, or disabled for analog signal acquisition, to minimize noise current in the shared return wire.

8.2.2 Class 2

A Class 2 interface shall use separate connections to transfer transducer signals and digital TEDS data. With separate connections, signal and data access may occur simultaneously. Because of the separation of analog and digital functions, the analog operation of the transducer has no effect upon the operation of the digital communication and subsequently shall not be defined in this standard.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

8.3 Digital Mode

8.3.1 Class 1

The operating mode of an IEEE 1451.4 Class 1 transducer shall be controlled by the polarity of the power-supply voltage or current delivered to the transducer. For negative voltages between 0 V and not exceeding –6 V (typically –5 V) with respect to the return applied to the signal of a constant-current powered transducer, or to a separate positive supply connection, an IEEE 1451.4 Class 1 transducer shall operate in the Digital Mode. The logic supply shall be current limited (see 8.5.3). Logic levels shall be defined as 0 V = logic state 0 and –5 V = logic state 1.

IEEE 1451.4 Nodes shall be accessible as two-leaded devices and shall adhere to a Data Transmission Protocol, allowing power and data to co-exist on one lead, while the other lead is used as return. The nodes shall use open-drain output devices and use parasitic power from the data line. Implementation of the 1451.4 MMI requires that users of this standard shall observe this protocol, which is explained in 5.5 and 8.5. The transmission protocol is a proprietary technique that has been adapted for use with this standard. The memory device requires that the logic 0 state be asserted with certain constraints on fall time. This timing cannot be met with the steering diode isolating the memory from the data line driver. Thus, a terminating pulldown resistor (R_t) shall be included in the circuit when diode current steering is used. The resistor discharges the capacitance of the memory circuit and wiring, ensuring that the logic 0 voltage is met within the proper timing requirements.

8.3.2 Class 2

The IEEE 1451.4 Class 2 MMI shall use a digital TEDS data connection separate from the analog connections. Logic levels shall be defined as 0 V = logic 0 and +5 V = logic 1. The logic supply shall be current limited (see 8.5.3). Timing and data access protocol for Class 2 are the same as Class 1 and are defined in 5.5 and 8.5. If a diode is used in the data connection of the Class 2 MMI for polarity protection, a terminating resistor shall also be used (see 8.3.1).

8.4 Line definitions

8.4.1 Class 1

Digital communication with an IEEE 1451.4 Class 1 transducer shall be on a single connection, shared with either power or analog signal.

Class 1 connections are summarized in Table 17.

Digital communication requires that the data line be driven to –5V relative to the Return by a current-limited source, as defined in 8.5.3.

8.4.2 Class 2

IEEE 1451.4, Class 2, transducers shall use a separate connection for digital communication. Digital and analog functions may take place simultaneously. See Table 18.

Digital communication requires that the data line be driven to +5V relative to the Return by a current-limited source, as defined in 8.5.3. Analog operation may continue uninterrupted by digital operation.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table 17—Line definitions for Class 1 MMI

Mode		Physical wire name			
		+ Power	Signal	Data	Return
Shared signal wire	**Analog Mode**	Not applicable	Analog signal	Not applicable	Return
	Digital Mode	Not applicable	(–) Data	Not applicable	Return
Shared power wire	**Analog Mode**	(+) Power	Analog signal	Not applicable	Return
	Digital Mode	(–) Data	Undefined	Not applicable	Return
Shared return wire	**Analog Mode**	(+) Power	Analog signal	Not used	Return
	Digital Mode	(+) Power	Analog signal	(–) Data	Return

Table 18—Line definitions for Class 2 MMI

Physical wire name		
Data	Data return	"n" Analog
(+) Data	Return	Analog signals

8.5 MMI digital Data Transmission Protocol

The following Data Transmission Protocol shall be used in the implementation of IEEE 1451.4 Mixed Mode Transducers (MMXdcr) and data acquisition systems. This protocol, allowing power to be supplied to—and data transfer to or from—one, or more, IEEE 1451.4 devices, shall be governed by the following rules:

1) The Protocol shall be a master-slave, multidrop, serial data protocol. A single master device (generally the data acquisition system) shall supply power and initiate each transaction, with each node (slave device), according to a defined transaction timing sequence, on a signal wire and return. Nodes shall use open-drain output line drivers and shall use parasitic power from the data line (see 8.5.3).
2) Each node shall contain a 64-bit URN, consisting of an 8-bit Family Code; 48-bit serial number and 8-bit CRC code, identifiable by the master, used to control individual access among multiple nodes, on a common bus. Details are contained in 5.4.
3) A hierarchy of commands, determined by the Family Code and issued by the master, shall control each Transaction Sequence (see 8.5.1) between the master and any node. The commands shall begin with an Initialization Sequence (see 8.5.1.1), followed by the Signaling Sequence (see 8.5.1.2), consisting of a ROM Function Command (see 8.5.1), then a Memory Function Command (see 8.5.1), each made up of Write and Read Time Slots (see 8.5.1.3), after which the node issues a Presence Pulse, and returns to the Reset State, awaiting the next Transaction.
4) A defined set of interface Signaling Sequences (see 8.5.1.2), based upon timing, shall be asserted by the master, including power/recovery, initialization, read and write, to transfer data between devices.
5) Data shall be written and read lsb first.
6) For IEEE 1451.4 Class 1 MMI systems, data shall be written and read in inverted format, with logic 1 typically at –5 V and logic 0 typically at 0 V. Exact limits on voltages are defined in 8.3.1. For IEEE 1451.4 Class 2 MMI systems, data shall be written and read in positive format, with logic 1 typically at +5 V and logic 0 typically at 0 V. Timing is identical in both Class 1 and Class 2.
7) Signal voltages more positive than 0 V shall be analog signals, in Class 1 MMIs.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Nodes may contain any of several digital and analog functions. Digital functions include random access memory (RAM), EEPROM, OTP, and 64 bits of permanently programmed ROM. Analog functions such as analog switches, data transmitter/receivers, couplers, ADC, DAC, and temperature transducers may also be included.

Each node shall have a 64-bit, preprogrammed, permanent ROM, containing the unique electronic registration number, which includes a Family Code and CRC code. This unique serial number is the basis upon which addressable, digital communication takes place, within the multidrop bus architecture. The Family Code defines the command set, to be used by the master in all digital data transactions with the node. The use of the unique electronic registration number is detailed in 5.4.

8.5.1 Transaction Sequence

The following Transaction Sequence shall control transactions in the IEEE 1451.4 Digital Mode:

1) *Initialization:* Each transaction shall begin with an Initialization Sequence. The master issues a reset pulse, after which the node(s) responds with a presence pulse. A presence pulse allows the master to determine that one, or more, nodes are present on the bus.
2) *ROM Function Command:* Following Initialization, the master shall issue one of the four ROM Function Commands, by using the Write and Read Time Slot sequences. Each ROM Command requires that the master follow a specific pattern of Write and Read sequences, to properly steer the node through its functions. Search ROM allows the master to determine the serial number of each node present on the bus. Match ROM allows the master to address one of a number of nodes on the bus, after having identified them with the Search ROM Command. Read ROM allows the master to read the contents of the 64-bit ROM, but if more than one node is present, data collisions will occur, and a CRC error will alert the master to the presence of multiple nodes. Skip ROM is a shortcut past the addressing functions, for use only when the presence of a single node is assured. If used with multiple nodes, the Skip ROM Command will cause corrupted data, so its use is discouraged. Other ROM Function Commands are available in various devices, but these are the basic node-addressing commands. A current data sheet on the specific node device is the most reliable source for the commands valid for that node. The illustrative data in “DS2430A 256-bit 1-Wire EEPROM” [B3] will allow the reader to gain some preliminary understanding of the logic processes involved in the ROM Commands.
3) *Memory Function Command and Data Transfer:* Having successfully navigated a ROM Command, the node shall enter the Memory Function Commands state. The master shall then issue the appropriate command and read or write data, with the proper series of Write and Read Time Slot sequences. Once again, a variety of commands are available, depending on the Family Code of the specific node device being addressed. Specific device data sheets are the most reliable source for commands available in a node device, but “DS2430A 256-bit 1-Wire EEPROM” [B3], “1-Wire Master Device Configuration” [B1], and Annex E will serve as a tutorial on the operation of example devices.

8.5.1.1 Initialization Sequence

NOTE—The following timing diagrams adhere to the timing of those presented in “DS2430A 256-bit 1-Wire EEPROM” [B3], but the logic polarity has been inverted to reflect the Data Transmission Protocol of the Class 1 MMI defined in this standard. The Class 2 MMI adheres to the same timing, but uses the positive logic levels presented in “DS2430A 256-bit 1-Wire EEPROM” [B3]. Signal names follow the convention of HIGH levels, and max values are those that are more positive than LOW levels and min values. Thus, tHI0 is a positive-going pulse (–5V to 0V) initiating a write cycle, but the HIGH level of the pulse represents the ZERO logic state. Similarly, Vil MAX is the least negative voltage level meeting the requirements for the negative voltage representing device power and the ONE logic state. Simply stated, 0 V = logic state 0 and –5 V = logic state 1, in these diagrams.

The bus master (data acquisition system) shall initiate each digital data transaction by issuing a reset pulse, driving the bus to its idle state of –5 V, to begin an Initialization Sequence (see Figure 15). To guarantee that

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

all nodes on the bus will be reset, the pulse shall be a minimum of 480 µs in duration. Within 15–60 µs following a reset pulse, the nodes shall assert a presence pulse, pulling the bus up to 0 V, for a duration between 15 µs and 60 µs.

From the idle state at –5 V, the master shall assert the 480 µs reset pulse, and the node shall return a presence pulse, 15–60 µs later. Whenever a transaction is suspended temporarily, the bus shall be left in the –5 V idle state, in order to return to the transaction. If the bus is left in the 0 V state longer than 120 µs, the node shall begin a reset sequence, and the transaction shall be terminated.

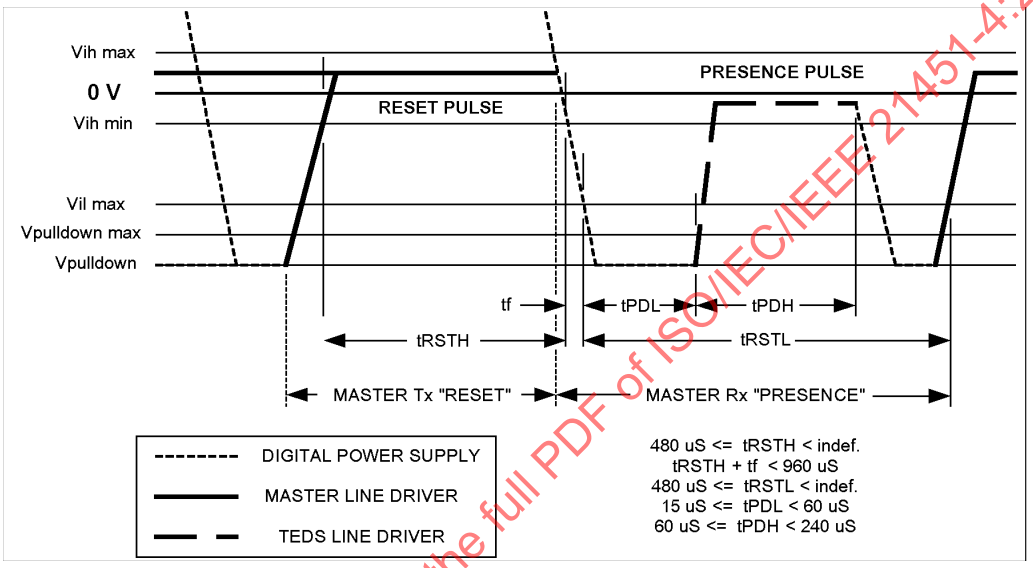


Figure 15—Initialization Sequence

8.5.1.2 Signaling Sequence

Following the Initialization Sequence, the master shall begin signaling the node, using one of several ROM Command sequences, followed by the desired Memory Command. The master shall use the Write One and Write Zero Time Slot sequences to write data to the node, or the Read Time Slot sequence, to read back data.

Write and Read Time Slots begin identically, when the master asserts the tHI start pulse from the –5 V idle state to 0 V for a duration of 1–15 µs. A write or read operation can be identified only within the context of the current command and following the exact data exchange sequence of that command.

For example, following the Initialization Sequence, a Read ROM Command (33h) is constructed of Write 1 and Write 0 time slots:

W1, W1, W0, W0, W1, W1, W0, W0 (33h, lsb first)

Following would be 64 Read Time Slots, to recover the Family Code, serial number, and CRC code. This sequence would place the node at the entry point to the Memory Commands, and if the master found the CRC to be in order, a Memory Command could be started. A bad CRC would indicate that more than one device had responded to the Read ROM Command, and the correct reaction would be for the master to assert a new Reset Sequence and proceed with a Search ROM Command (F0h), to determine the identities of the nodes on the bus.

These operations are explained in detail by the flowcharts contained in “DS2430A 256-bit 1-Wire EEPROM” [B3].

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

8.5.1.3 Time slots

The master shall initiate the Write One Time Slot (see Figure 16) by asserting a write pulse, tHI1, 0 V (logic 0), for a duration from 1–15 µs, and then releasing the bus to be pulled down to the –5 V (logic 1) level. When 60 µs has elapsed from the rising edge of the write pulse, the node shall have strobed the data in, and the next time slot may be begun, following the 1 µs charge recovery time. Because the bus is in the –5 V idle state, observing the 120 µs slot time is important only to the data rate. Remaining idle longer simply slows down the data transfer.

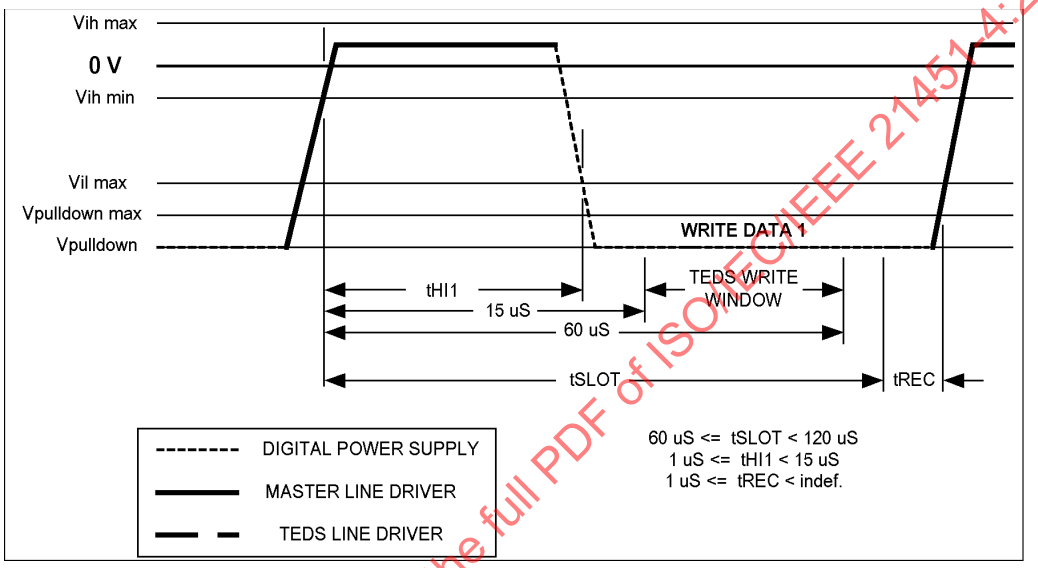


Figure 16—Write One Time Slot

The master shall initiate the Write Zero Time Slot (see Figure 17) by asserting a write pulse, tHI0, 0 V (logic 0), for a duration from 1–15 µs, and then holding the bus at the 0 V (logic 0) level. When 60 µs has elapsed from the rising edge of the write pulse, the node shall have strobed the data in, and the next slot may be begun, following the 1 µs charge recovery time. Because the bus is in the 0 V idle state, observing the 120 µs slot time is important to preventing an unintentional reset sequence from beginning. Returning to the –5 V idle state as soon as possible is important for the recovery time.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

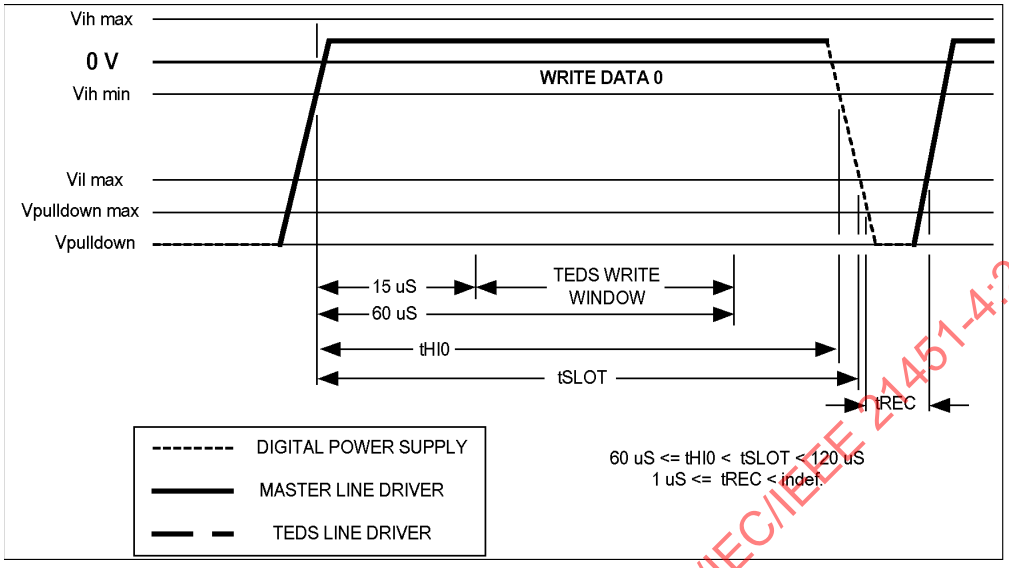


Figure 17—Write Zero Time Slot

The master shall initiate the Read Time Slot (Figure 18) by asserting a read pulse, tHIR, 0 V (logic 0), for a duration from 1 µs to maximum 15 µs, and then shall release the bus, to either be pulled down to the –5 V (logic 1) level or brought to the 0 V (logic 0) level by the node data transmitter. Within the window after the read pulse and until 15 µs has elapsed from the rising edge of the read pulse, the node shall have strobed valid data out. The node then releases the bus, which is pulled down to the –5 V (logic 1) level. The next slot may begin, following the optional 45 µs release time plus the 1 µs charge recovery time. Because the bus is in the –5 V idle state, observing the 120 µs slot time is important only to the data rate. Remaining idle longer simply slows down the transfer.

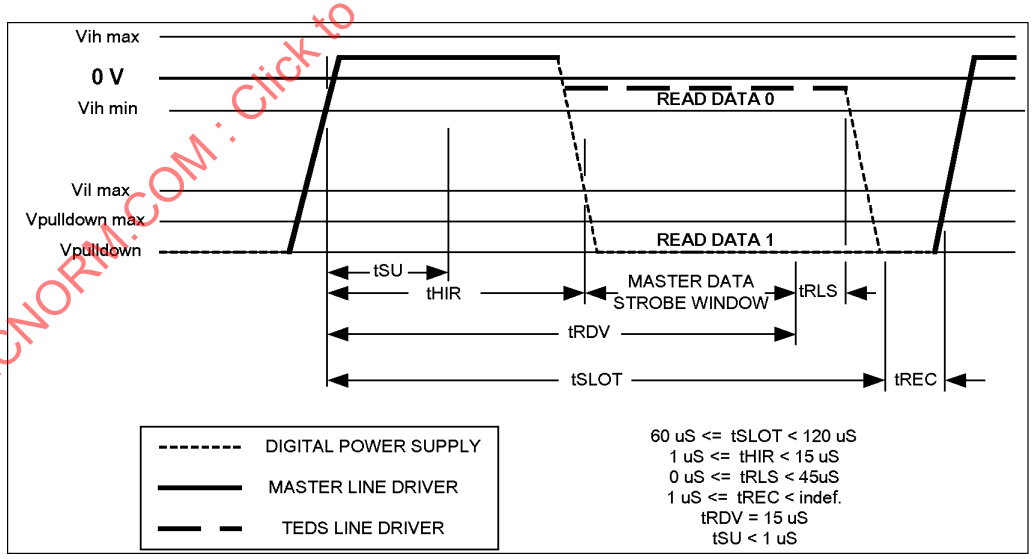


Figure 18—Read Time Slot

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

8.5.2 Electrical characteristics

Table 19 contains detailed information on the signal voltages, which shall be observed in the implementation of the IEEE 1451.4 Class 1 MMI. Signals follow the convention of HIGH levels and max values being more positive than LOW levels and min values. Voltages for the IEEE 1451.4 Class 2 Interface shall have the same magnitude, but be of positive polarity. Timing details contained in Table 20 shall apply to both Class 1 and Class 2 Interfaces.

Table 19—DC electrical characteristics

(–40 °C to +85 °C; Vpulldown = –3.4 V to - 6.0 V)

Parameter	Symbol[a]	Min	Typ	Max	Units
Analog Mode (Class 1)		0	2 to 30		V
Logic 1[b]	Vil	–6.0	–5	–3.4	V
Logic 0[b]	Vih	–0.8		0	V
Output[c] Logic 0 @ –4mA	Voh	–1		0	V
Output[c] Logic 1		–6.0[d]		Vpulldown	V
Supply current limit[b] (during read)	IL	–1.5		–4.0	mA
Operating charge[c]	QOP			30	nC
Programming current[c]	IP			600	µA
Capacitance data pin[c]	CD			800	pF

[a]Refer to logic diagrams.
[b]Master line driver.
[c]TEDS line driver.
[d]Voltages more negative might damage the digital devices.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

8.5.3 IEEE 1451.4 logic power supply

Nodes used with the IEEE 1451.4 MMI, shall use open-drain output line drivers and shall use parasitic power from the data line. Power for the nodes shall be supplied during read by a current-limited supply, which shall supply 5 V, typically (6 V maximum), at 1.5 mA, in the Logic 1 state. In the Logic 0 state, current shall be limited to 4.0 mA, maximum. For the Class 1 MMI, all voltages shall be negative. For the Class 2 Interface, all voltages shall be positive.

© IEEE 2004 – All rights reserved

Table 20—AC electrical characteristics

(–40 °C to +85 °C; Vpull-up/down = –3.4 V to –6.0 V)

Parameter	Symbol	Min	Max	Units
Time slot	tSLOT	60	120	µs
Write 1 HIGH time	tHI1	1	15	µs
Write 0 HIGH time	tHI0	60	120	µs
Read HIGH time	tHIR	1	15	µs
Read Data Valid	tRDV	Exactly 15		µs
Release time	tRLS	0	45	µs
Release time @ Vpulldown = –4.6V to –6.6V	tRLS	7	30	µs
Read data setup	tSU		1	µs
Recovery time	tREC	1		µs
Reset time high	tRSTH	480	960[a]	µs
Reset time low	tRSTL	480	Indefinite	µs
Presence detect high	tPDH	60	240	µs
Presence detect low	tPDL	15	60	µs

[a]tRSTH may be extended indefinitely, but if >960 µs, it might mask devices attempting to assert an interrupt signal on the bus.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

9. Transducer Block specification

IEEE Std 1451.1-1999 provides the NCAP information model applicable to all standards in the IEEE 1451 family. That information model specifies a comprehensive object model in which the `IEEE1451_Dot4TransducerBlock` class is introduced without supporting definition. This strategy reserved space within the IEEE 1451.1 class hierarchy and deferred definition to this standard. Annex L specifies an IEEE 1451.1 adapter that allows the IEEE 1451_Dot4TransducerBlock class and its associated object model introduced in this chapter to be used transparently in an IEEE 1451.1 environment.

This clause defines the IEEE 1451.4 Transducer Block. Throughout the remainder of this clause, the term Transducer Block (two words, each capitalized) shall denote the collective logic required to manage the transducer bus and all external components. In contrast, the term `TransducerBlock` (one word, note typeface) shall denote an object instantiated from the `IEEE1451_Dot4TransducerBlock` class. One or more `TransducerBlocks` may be instantiated in an NCAP or under other application software.

The Transducer Block shall execute on an NCAP, except on distributed settings where independent components shall execute on their target environment (e.g., a computer-based NCAP connected to an instrument acting as Transducer Interface instance) as depicted in Figure 19. It encapsulates all management responsibility for the MMI and the associated transducer bus.

© IEEE 2004 – All rights reserved

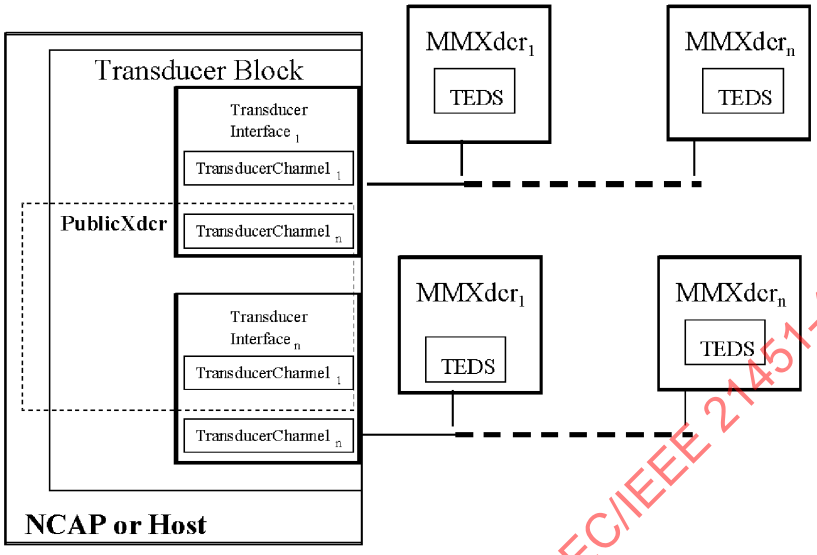


Figure 19—TransducerBlock context diagram

The modular architecture of the Transducer Block can be extended or abbreviated to support deployment in execution environments other than an IEEE 1451.1-compliant system. The object model described shall be used if an IEEE 1451.1 NCAP is used within the constraints defined by the adapter defined in Annex L. If the NCAP or host does not implement IEEE 1451.1, the object model may be used but is optional. A host is a system capable of supporting the IEEE1451_Dot4TransducerBlock class.

The UML was used extensively in developing this clause. See Booch et al. [B2], Fowler [B4], Mellor and Balcer [B10], and Pritchard [B11] for more information about UML.

9.1 Overview

The Transducer Block is a software component that provides an interface between `Transducer Channels` and application software. It includes methods for the following:

a) Configuring and managing the MMI

b) Configuring and managing IEEE1451.4 Transducers (IEEE1451_Dot4MMXdcr) and their associated TransducerChannels (IEEE1451_Dot4XdcrChannels)

c) Extracting and encoding TEDS data

The Transducer Block provides the required functionality for any IEEE 1451 Transducer Block by exposing Transducer Channels as network accessible objects. The Transducer Block also supports information services unique to this standard (e.g., Appended TEDS support.).

The Transducer Block specification is composed of

— Transducer Block Object Model (**TBOM**): Describes the associations and ownership of objects and components.

— Common Object Interface (**COI**): Defines a collection of objects that allow components to communicate as discrete components within the TBOM.

— TEDS Service: Supports TEDS access.

An overview of component software in transducer applications can be found in Lopez-Reyna and Zemel [B9].

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.1.1 Transducer Block Object Model (TBOM)

The TBOM is the collection of objects that form the core functionality within the Transducer Block. A class diagram of the TBOM is provided in Figure 20. For brevity, class names within the diagrams are presented using a shorthand notation in which the "IEEE1451_Dot4" prefix has been omitted. The full class names and a brief description of the object represented are provided as Table 21. All classes are described in detail in 9.2. An IEEE1451.4 TBOM schema describing the information structure can be found at Annex K.

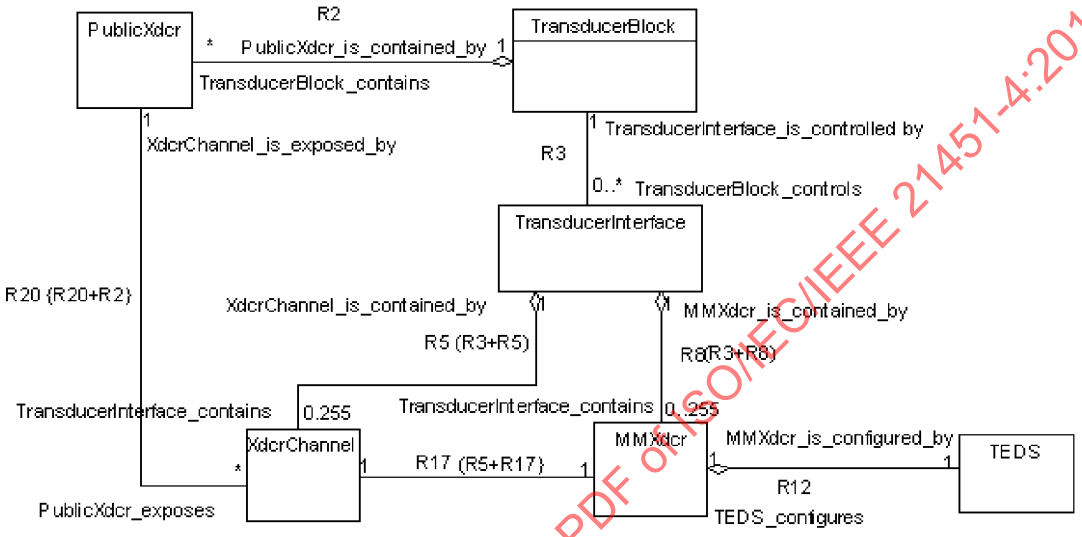


Figure 20—TBOM class diagram

Table 21—TBOM class summary

Class Name	Description
IEEE1451_Dot4TransducerBlock	Transducer Block
IEEE1451_Dot4TransducerInterface	General abstraction of the MMI
IEEE1451_Dot4MMXdcr	Mixed-Mode Transducer
IEEE1451_Dot4TEDS	Transducer Electronic Data Sheet
IEEE1451_Dot4BTEDS	Binary TEDS structure
IEEE1451_Dot4Template	Template
IEEE1451_Dot4XdcrChannel	Transducer Channel
IEEE1451_Dot4Node	Addressable Dot4 device
IEEE1451_Dot4Property	TDL defined property
IEEE1451_Dot4TEDSService	TEDS management interface
IEEE1451_Dot4PublicXdcr	Public transducer

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.1.2 COI

The COI is an optional high-level object-oriented messaging service, defined as an abstraction layer that may be used in other standards in the IEEE1451 family. The COI allows TBOM elements to communicate through a standard interface within the Transducer Block. The COI serves as the messaging and event notification mechanism for TBOM components whether they are local or distributed.

The COI is comprised of classes summarized in Table 22. A full description of each class is provided in 9.3.

Table 22—COI class summary

Class Name	Description
IEEE1451_Dot4CommandSource	Generates command messages
IEEE1451_Dot4CommandSink	Receives commands messages
IEEE1451_Dot4EventSource	Generates events
IEEE1451_Dot4EventSink	Receives events

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

9.1.3 TEDS Service

The TEDS Service interface accommodates the environmental or physical packing constraints that prevent the manufacturer from embedding information or capability. The TEDS Service provides support for Appended TEDS, TEDS encoding, TEDS decoding, and a Template Database.

From a logical perspective, application software shall find the BTEDS component of the TEDS in the IEEE 1451.4 Transducer. The BTEDS is a device-independent binary encoded TEDS as defined in 9.2.7 and Annex L. This paradigm is fundamental to all standards within the 1451 family. The internalization of the TEDS is considered to be one of the key attributes that make a transducer "smart." Logically, the TEDS is embedded in the MMXdcr.

This standard defines the Appended TEDS for cases where the physical package does not contain the TEDS. The manufacturer shall supply the TEDS in a machine-readable format for deployment on the end user's system (BTEDS). To maintain the integrity of the logical perspective, it is the responsibility of the Transducer Block to satisfy requests for all TEDS, whether embedded or virtual, in a manner transparent to the application. This responsibility is encapsulated in the TEDS Service interface.

The TEDS Service provides support even when the requested binary encoded part of the TEDS `(BTEDS)` is not local to the MXdcr. This service relies on deployment-specific configuration data to locate and retrieve the TEDS from the NCAP's or host's file system.

Additionally, the TEDS Service provides operations that allow the user to manage templates and encode and decode TEDS.

9.1.4 Relationship semantics

The TBOM UML specification makes extensive use of relationship semantics and restrictions. UML associations are described by enumerated relationships R0 through R31 in Table 23 and complemented with XML schema in Annex L.

© IEEE 2004 – All rights reserved

Table 23—TBOM relationship semantics

Name	Type	Description
R0	reserved	`Reserved`
R1	reserved	`Reserved`
R2	containment	TransducerBlock contains PublicXdcr
R3	control	TransducerBlock controls `TransducerInterface`
R4	service	TEDS Service provides service to Transducer Block
R5	containment	`TransducerInterface` contains XdcrChannel
R6	reserved	`Reserved`
R7	activation	EventSink activates TransducerBlock
R8	containment	`TransducerInterface` contains MMXdcr
R9	reserved	`Reserved`
R10	containment	MMXdcr contains Node
R11	containment	MMXdcr contains Property
R12	configuration	TEDS configures MMXdcr
R13	reserved	`Reserved`
R14	reserved	`Reserved`
R15	reserved	`Reserved`
R16	reserved	`Reserved`
R17	exposure	XdcrChannel exposes MMXdcr
R18	containment	TEDS contains Template
R19	containment	TEDS contains BTEDS
R20	exposure	PublicXdcr exposes XdcrChannel
R21	containment	Node contains BTEDS
R22	ownership	`TransducerInterface` owns CommandSink
R23	reserved	Reserved
R24	reserved	Reserved
R25	ownership	TransducerBlock owns CommandSource
R26	controls	CommandSource controls CommandSink
R27	reserved	`Reserved`
R28	reserved	`Reserved`
R29	reserved	`Reserved`
R30	activation	TransducerInterface activates TriggerSource
R31	invocation	EventSource invokes EventSink

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.1.5 Relationship semantics definitions

Each UML association is qualified with extended semantics action words defined in Table 24.

Table 24—Relationship semantics definitions

Relationship	Semantics
activation	Object A activates Object B. Associated with Event Sources this relationship defines that Object A activates an Event in Object B. See 9.3.2.
control	Object A controls Object B. Object A is able to issue commands to Object B but not the opposite. See 9.3.1.1.
containment	Object A composes part of Object B. Object B is expected to control the life cycle of Object A.
configuration	Object A configures Object B. Object A contains information that allows Object B to set-up and operate.
exposure	Object A exposes Object B. Object A aggregates Object B and shares its ownership with other objects classes as defined by the TBOM.
ownership (its)	Object A its Object B. Object A is expected to control the life cycle of Object A.
reserved	Relationship has been reserve for future use by the IEEE 1451.4 group.
Service	Object A serves Object B. Object A provides a service interface to Object B. See 9.4.
Invoke	Object A invokes Object B. Object A invokes an Event in Object B. See 9.3.3.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

9.1.6 Data Model Semantic Notation

The following notation is provided to simplify the symbolic representation of elements defined in Annex L:

Root_Element:First_subelement:Second_subelement….:N_subelement

Root_Element shall represent a base data model element or type (for example, IEEE1451_Dot4PublicXdcrType).

First_subelement shall represent a subelement on a given data model (for example, ExposedParameter).

Second_subelement shall represented a subelement of the First_subelement element (for example, MetaData).

Then the notation IEEE1451_Dot4PublicXdcrType:Parameter:MetaData element shall refer to the MetaData element defined in the Parameter element of the IEEE1451_Dot4PublicXdcrType.

9.2 TBOM specification

The TBOM forms the functional foundation for the Transducer Block. Each class is described in the following subclauses.

© IEEE 2004 – All rights reserved

9.2.1 IEEE1451_Dot4TransducerBlock

The IEEE1451_Dot4TransducerBlock provides the interface between IEEE1451_Dot4TransducerChannels and application software. There can be more than one instance of the IEEE1451_Dot4TransducerBlock class on a given host or NCAP. This object controls zero or more instances of the IEEE1451_Dot4TransducerInterface class. The IEEE1451_Dot4TEDSService provides TEDS support. The IEEE1451_Dot4TransducerBlock relationships shall be restricted by the class diagram in Figure 21. The IEEE1451_Dot4TransducerBlock:TransducerBlockID element shall always have a value of 0.

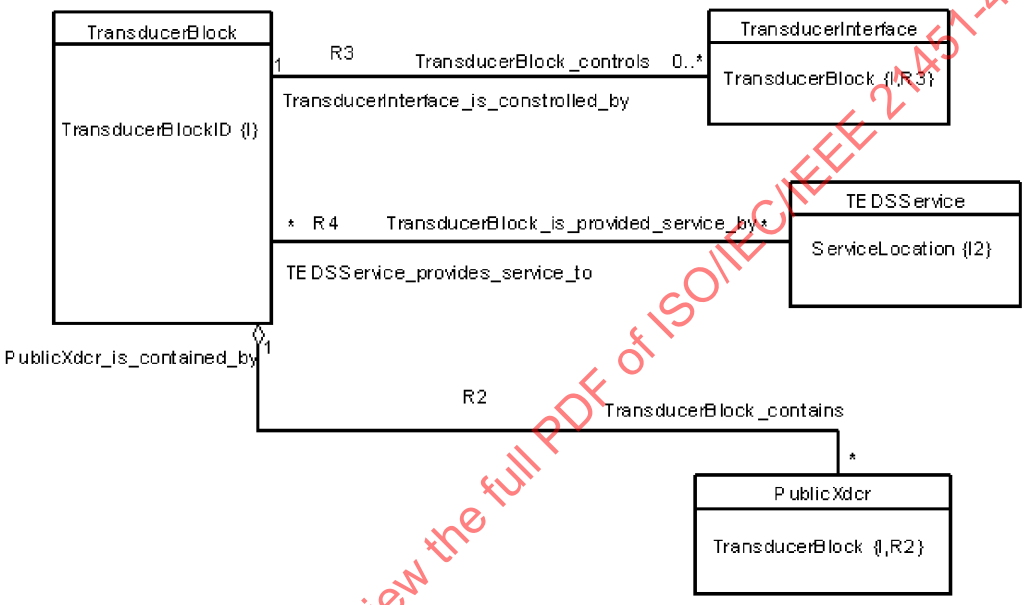


Figure 21—IEEE1451_Dot4TransducerBlock associations

9.2.2 IEEE1451_Dot4TransducerInterface

The IEEE1451_Dot4TransducerBlock controls the IEEE1451_Dot4TransducerInterface object. The IEEE1451_Dot4TransducerInterface object shall control and contain all IEEE1451_Dot4MMXdcr objects representing IEEE 1451.4 Transducers physically attached to the MMI's transducer bus. An IEEE1451_Dot4TransducerInterface is capable of sourcing events as described in the class diagram in Figure 32.

The IEEE1451_Dot4TransducerInterface shall contain all IEEE1451_Dot4XdcrChannel objects exposing IEEE1451_Dot4MMXdcr within the same IEEE1451_Dot4TransducerInterface. Each IEEE1451_Dot4TransducerInterface holds zero or more instances of IEEE1451_Dot4MMXdcr and the IEEE1451_DotXdcrChannel objects. The IEEE1451_Dot4TransducerInterface associations are restricted by the class diagram in Figure 22. There are no TEDS associated with IEEE1451_Dot4TransducerInterface object.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

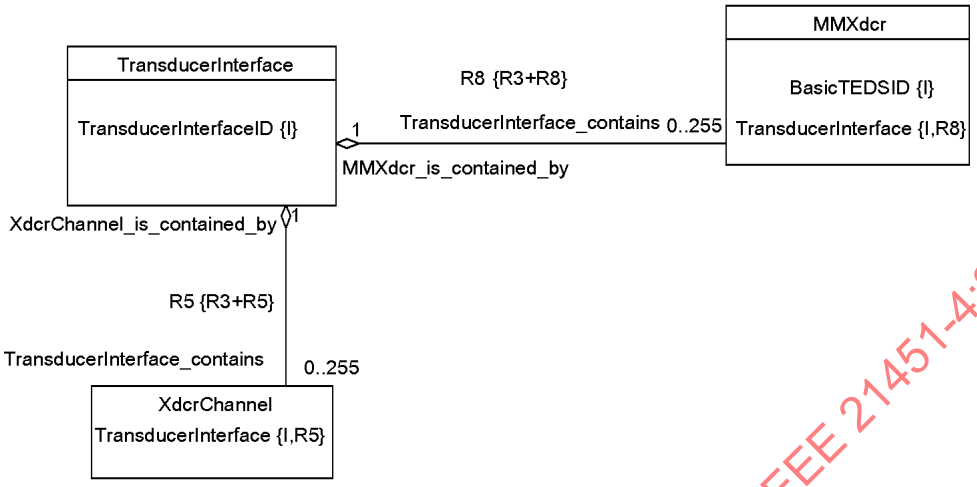


Figure 22—IEEE1451_Dot4TransducerInterface associations

9.2.3 IEEE1451_Dot4MMXdcr

The `IEEE1451_Dot4MMXdcr` objects represent IEEE 1451.4 Transducers and maintains an association (R10) to all nodes related to a MMI transducer bus. A Node-List as defined by 5.4 and Annex L is represented by relationship R10 and R21. Each MMXdcr holds zero or more instances of IEEE1451_Dot4`Property` and one or more instances of `IEEE1451_Dot4Node`. `IEEE1451_Dot4TEDS` objects associated with `IEEE1451_Dot4MMXdcr` shall be restricted by the relationships defined by the class diagram in Figure 23 and Figure 24.

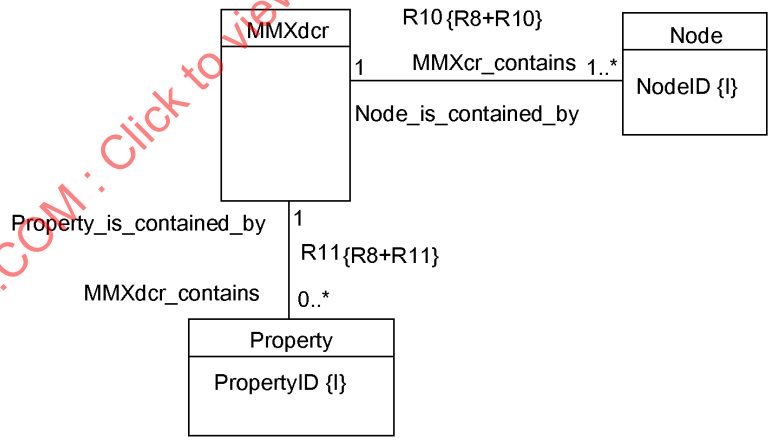


Figure 23—IEEE1451_Dot4MMXdcr associations

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

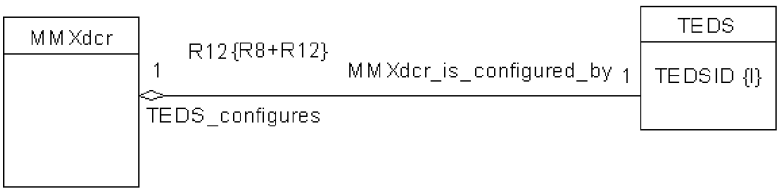


Figure 24— IEEE1451_Dot4TEDS association for IEEE1451_Dot4MMXdcr

9.2.4 IEEE1451_Dot4XdcrChannel

The `IEEE1451_Dot4XcdrChannel` refers to a MMXdcr and all electronics in the path that connect the transducer to the communication functions represented by the `IEEE1451_Dot4MMXdcr` class. The `IEEE1451_Dot4XdcrChannel` shall expose `IEEE1451_Dot4MMXdcr` transducers and their associated metadata (`IEEE1451_Dot4Property`). `IEEE1451_Dot4XcdrChannel` associations are depicted in the class diagram in Figure 25. There are no `IEEE1451_Dot4TEDS` directly associated with the `IEEE1451_Dot4XdcrChannel`.

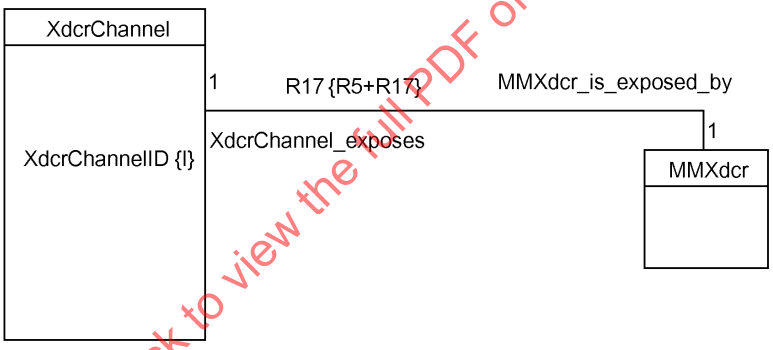


Figure 25—IEEE1451_Dot4XdcrChannel associations

The IEEE1451_Dot4XdcrChannel XdcrChannelType element defined in Annex L shall contain information on whether the `IEEE1451_Dot4XcdrChannel` is exposing a sensor or an actuator. The value of the IEEE1451_Dot4XdcrChannelType element shall be defined by the `IEEE1451_Dot4MMXdcr` metadata containing an `IEEE1451_Dot4Property` with element name value of ELECSIGTYPE. If no `IEEE1451_Dot4Property` with element name value of ELECSIGTYPE can be found the IEEE1451_Dot4XdcrChannelType element shall have a value of "Generic." An actuator as used in this clause shall cause a physical or embedded output action to occur. A sensor as used in this clause shall measure some physical parameter. The ELECSIGTYPE property is defined in B.4.1.

`IEEE1451_Dot4XdcrChannel` objects shall set their Active element value to "false" if their associated `IEEE1451_Dot4Xdcr` object is removed from the system. The IEEE1451_Dot4XdcrChannel is said to be inactive. Inactive `IEEE1451_Dot4XdcrChannel` objects shall not be unbounded until the `IEEE1451_Dot4TransducerBlock` is reset. An associated `IEEE1451_Dot4Xdcr` object is defined as an `IEEE1451_Dot4MMXdcr` object whose MMXdcrID reference element value is listed in the `IEEE1451_Dot4XdcrChannel` AssociatedMMXdcr element.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.2.5 IEEE1451_Dot4Property

The `IEEE1451_Dot4Property` class shall represent all TDL properties defined in the TEDS template. The `IEEE1451_Dot4Property` data model is defined in Annex L. Mapping of TDL properties data types to IEEE1451_Dot4ValueTypes elements shall follow the mapping described in Table 25.

Table 25—TDL datatype to IEEE1451_Dot4ValueType element

<Property datatype>	Corresponding IEEE1451_Dot4ValueType
Date	Value = xs:date
UnInt	Value = xs:decimal
Chr5	Value = xs:string
ASCII	Value = xs:string
Unicode	Value = xs:string
String5	Value = xs:string
String7	Value = xs:string
String16	Value = xs:string
<float_type>	
ConRelRes	Value = xs:decimal, MaxValue = xs:decimal, MinValue = xs:decimal, QuantizationStep = xs:decimal
ConRes	Value = xs:decimal, MaxValue = xs:decimal, MinValue = xs:decimal, QuantizationStep = xs:decimal
Single	Value = xs:decimal, MaxValue = xs:decimal, MinValue = xs:decimal, QuantizationStep = xs:decimal
<enum_type>	
(Defined enumeration type)	Value = xs:string Value shall have a value equal to any Enumerant element defined in the IEEE1451_Dot4Property:EnumeratList sequence element.
<Extended Functionality>	
Bitbin	Value = IEEE1451_Dot4BitBinType

`IEEE1451_Dot4Property` objects whose Exposed element is set to one or "true" shall be defined as exposed. `IEEE1451_Dot4TransducerBlock` operations and subclass specific means shall determine which IEEE1451_Dot4Property objects shall be exposed.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

IEEE1451_Dot4Property objects whose data model contains the optional Extended_FN_Metadata element shall be defined as "Extended Functionality properties." All other properties shall be termed "information properties."

Information properties shall contain the most up-to-date value extracted from the TEDS. Properties that are associated with an Extended Functionality Control Property shall contain the most up-to-date value defined by the Control Property.

Extended Functionality IEEE1451_Dot4Property objects control a particular configuration setting in an IEEE 1451.4 Transducer (i.e., gain, filter settings, etc.). IEEE Std 1451.4-2004 defines two types of Extended Functionality properties: Control Properties that directly control the underlying hardware by setting the value of its IEEE1451_Dot4Property:Value element and associated properties whose IEEE1451_Dot4Property:Value element is a function of the IEEE 1451.4 Transducer FR value. Both types of Extended Functionality properties shall have an ExtendedFN_MetaData element representing all the property's functional states as defined by the Function subproperty (see 7.4.9.2). ExtendedFn_MetaData:PropertyType element shall indicate if the Property is representing a Control Property or an associated property.

Correspondingly, subclass specific means shall determine if the FunctionalState is disabled by matching its PropertyValue against the IEEE1451 Dot4 Transducer FR as defined in 7.4.9.1.3.

9.2.6 IEEE1451_Dot4TEDS

The IEEE1451_Dot4TEDS class shall represent all TEDS defined by IEEE Std 1451.4-2004. The class diagram in Figure 26 shall define the relationships for the IEEE1451_Dot4TEDS class. IEEE1451_Dot4TEDS shall contain an IEEE1451_Dot4TEDSType:BasicTEDSID element that uniquely associates the TEDS to an IEEE1451_Dot4MMXdcr. The value of the IEEE1451_Dot4TEDSTYPE:BasicTEDSID element shall be encoded as the Basic TEDS field defined in 7.1.4.

The IEEE1451_Dot4TEDS is composed of an IEEE1451_Dot4BTEDS class that represents a device-independent raw binary array containing the TEDS encoded information as defined in 9.2.7. IEEE1451_Dot4Templates shall represent templates as defined in 9.2.8.

9.2.7 IEEE1451_Dot4BTEDS

The IEEE1451_Dot4BTEDS class shall represent a device-independent binary encoded TEDS and shall be restricted by relationship R19. The IEEE1451_Dot4BTEDSType shall be a data structure represented as an XML schema xs:hexBinary data type defined in Annex L.

The IEEE1451_Dot4BTEDS element of the IEEE1451_Dot4TEDS element shall be constructed by concatenating the MemoryBanks elements in the order defined by the IEEE1451.4 XML device description instance files and removing all CRCs (see 6.4). IEEE1451_Dot4BTEDS data structures can be encoded into more than one memory IEEE1451_Dot4Node of type memory or Combined memory and functional as defined by relationship R21. Multiple IEEE1451_Dot4BTEDS can share one IEEE1451_Dot4Node of type memory or Combined memory and functional as defined by relationship R21. The IEEE1451_Dot4Node type is specified in 9.2.9.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

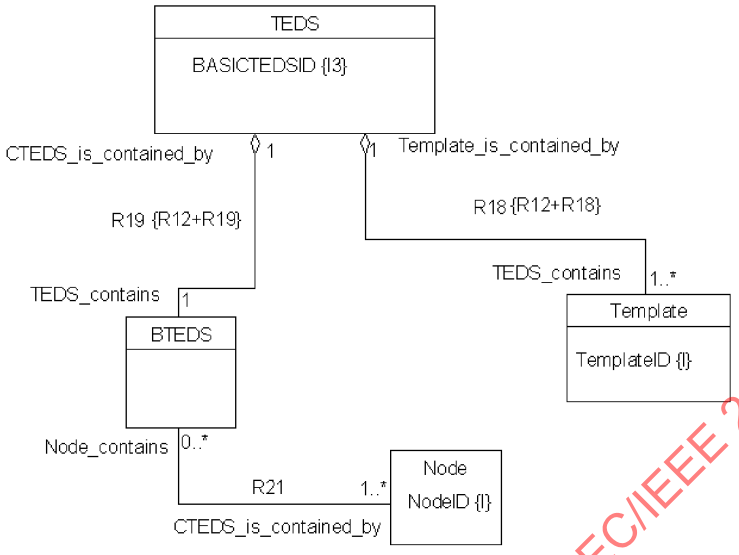


Figure 26—TEDS structural class diagram

9.2.8 IEEE1451_Dot4Template

The `IEEE1451_Dot4Template` shall represent the IEEE1451.4 template instances defined in A.1 and Annex L and restricted by relationship R18. Each template shall contain a unique TemplateID defined in A.1 and Annex L.

9.2.9 IEEE1451_Dot4Node

The `IEEE1451_Dot4Node` represents an abstract model for memory and functional devices. Each node shall be associated with an IEEE1451.4 XML device description schema instance that provides a mechanism for describing the functionality and structure of `IEEE1451_Dot4Node`. Devices compliant with the schema shall be able to encode and decode TEDS information transparently. IEEE1451.4-compliant devices shall not be required to either parse or recognize the device description XML schema.

Three types of nodes are recognized:

1) *Memory only* shall be devices that only define the MemoryBank element of the XML device description schema defined in Annex F. MemoryBanks shall be organized in the order the information will be encoded.
2) *Functional only* shall be devices that define the SwitchChannel and or TemperatureChannel elements of the XML device description schema in Annex F.
3) *Combined Memory and Functional* shall be devices that define the MemoryBank and the SwitchChannel and or the TemperatureChannel elements of the XML device description schema in Annex F.

The IEEE1451.4 XML device description file for frequently used devices can be found in “1-Wire Master Device Configuration” [B1].

9.2.10 IEEE1451_Dot4PublicXdcr

The `IEEE1451_Dot4PublicXdcr` shall represent a public transducer that exposes `IEEE1451_Dot4XdcrChannel objects`. The IEEE1451_Dot4PublicXdcr abstracts and groups the functionality and metadata of one or more IEEE1451_Dot4XdcrChannel exposed by the

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

IEEE1451_Dot4TransducerBlock. The `IEEE1451_Dot4PublicXdcr` data model is defined in Annex L. The `IEEE1451_Dot4PublicXdcr` object shall be restricted as defined by the class diagram in Figure 27. Rules to group `IEEE1451_Dot4XdcrChannels` into `IEEE1451_Dot4PublicXdcrs` are presented in 9.2.10.1.

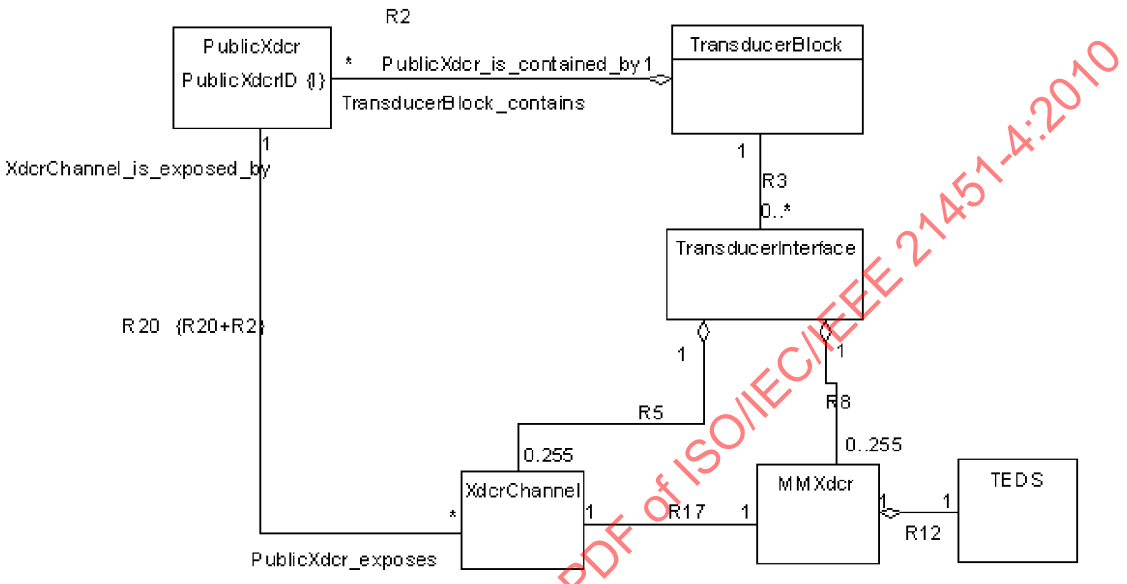


Figure 27—IEEE1451_Dot4PublicXdcr associations

`IEEE1451_Dot4PublicXdcr` objects shall set their Active element value to "false" (IEEE1451_Dot4PublicXdcr is inactive) if one or more of its associated `IEEE1451_Dot4XdcrChannel` objects are inactive. Inactive `IEEE1451_Dot4PublicXdcr` objects shall not be unbounded until the `IEEE1451_Dot4TransducerBlock` is reset. An associated IEEE1451_Dot4XdcrChannel objects is defined as an `IEEE1451_Dot4XdcrChannel` object whose XdcrChannelID reference element value is listed in the IEEE1451_Dot4PublicXdcr objects AssociatedXdcrChannels element.

Each exposed IEEE1451_Dot4Property object in an IEEE1451_Dot4XdcrChannels XdcrChannelPropertyList element shall be represented by at least one Parameter element on its associated IEEE1451_Dot4PublicXdcr object. IEEE1451_Dot4PublicXdcrs whose IEEE1451_Dot4XdcrChannels are associated to one or more IEEE1451_Dot4PublicXdcrs shall maintain a static data model of their Parameter elements until an IEEE1451_Dot4TransducerBlock interface operation or event updates the Parameter elements of the IEEE1451_Dot4PublicXdcr.

If the IEEE1451_Dot4PublicXdcr:Parameter Scalar, Vector, ScalarSeries or VectorSeries elements are accessed by read and update or write and update operations defined in the IEEE1451_Dot4TransducerBlock interface then the Parameter element data model shall be updated by subclass specific means; otherwise, the data model shall remain static. Parameters with Vector or VectorSeries elements shall refresh all associated IEEE1451_Dot4XdcrChannel objects by subclass specific means when accessed by a Read and Update or Write and Update operation.

9.2.10.1 IEEE1451_Dot4PublicXdcr IEEE1451_Dot4XdcrChannel grouping rules

The `IEEE1451_Dot4XdcrChannel` associated with an `IEEE1451_Dot4TransducerBlock` shall be exposed and grouped by an `IEEE1451_Dot4PublicXdcr` in accordance with the following restrictions and mapping rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Rule 1: Restrictions based on `IEEE1451_Dot4XdcrChannel` type exposed by the `IEEE1451_Dot4PublicXdcr`

The ParameterInterpretation element of the `IEEE1451_Dot4PublicXdcr` data model shall have a value of PI_SENSOR if all the exposed `IEEE1451_Dot4XdcrChannels'` IEEE1451_Dot4XdcrChannelType elements have values of Voltage_Sensor, Current_Sensor, Bridge_Sensor, Resistance_Sensor, LVDT_Sensor, or Pulse_Sensor as defined in Annex L.

The ParameterInterpretation element of the `IEEE1451_Dot4PublicXdcr` data model shall have a value of PI_ACTUATOR if all the exposed `IEEE1451_Dot4XdcrChannels'` IEEE1451_Dot4XdcrChannelType elements have values of Voltage_Actuator, Current_Actuator, or Pulse_Actuator as defined in Annex L.

The ParameterInterpretation element of the `IEEE1451_Dot4PublicXdcr` data model shall have a value of PI_USER_DEFINED if the exposed `IEEE1451_Dot4XdcrChannels'` IEEE1451_Dot4XdcrChannelType elements have different values or if any of them has a value of Generic as defined in Annex L.

Rule 2: Restrictions based on grouping information involving one or more `IEEE1451_Dot4XdcrChannel` instances

Grouping Information for `IEEE1451_Dot4XdcrChannels` shall be derived from the `IEEE1451_Dot4MMXdcr` BasicTEDSID attribute and by `IEEE1451_Dot4Property` instances with grouping information in each exposed IEEE1451_Dot4XdcrChannel.

Case 1: `IEEE1451_Dot4PublicXdcr` **exposing a single** `IEEE1451_Dot4XdcrChannel`

Unless indicated by Case 2 or 3 if no grouping information is present, each `IEEE1451_Dot4XdcrChannel` shall be exposed by an `IEEE1451_Dot4PublicXdcr` with a Type element set equal to SCALAR. Exposed IEEE1451_Dot4Propertys' shall be represented by a Parameter element with a Scalar or ScalarSeries element.

Case 2: `IEEE1451_Dot4PublicXdcr` **instances grouping** `IEEE1451_Dot4XdcrChannel` instances exposing `IEEE1451_Dot4MMXdcr` **associated with different** `IEEE1451_Dot4TransducerInterface instances.`

`IEEE1451_Dot4PublicXdcr` instances exposing `IEEE1451_Dot4XdcrChannel` instances containing `IEEE1451_Dot4MMXdcr` with the same BasicTEDSID element and grouping properties on their TEDS physically connected to different `IEEE1451_Dot4TransducerInterface instances` but within the same NCAP or host shall constitute a group. `IEEE1451_Dot4XdcrChannel` groups shall be exposed through the same `IEEE1451_Dot4PublicXdcr` object. If no grouping properties can be found, then no `IEEE1451_Dot4XdcrChannel` grouping information shall be assumed. Grouping Properties are defined in B.2. The IEEE1451_Dot4PublicXdcrType AssociatedXdcrChannels element shall contain a sequence of XdcrChannelID representing the IEEE1451_Dot4XdcrChannel elements exposed by the IEEE1451_Dot4PublicXdcr in the order defined by the TDL MemberIndex property tag of each IEEE1451_Dot4XdcrChannel s' XdcrChannelPropertyList element. See B.2.3.

The IEEE1451_Dot4PublicXdcr:CoordinateSystem element shall be set equal to the enumerated value of the property tag CoordinateSystem of the first exposed IEEE1451_Dot4XdcrChannel object. IEEE1451_Dot4PublicXdcr:CoordinateSystem element shall only be valid for IEEE1451_Dot4PublicXdcr with Type element equal to Vector.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Exposed `IEEE1451_Dot4Property` objects associated to an `IEEE1451_Dot4XdcrChannel` covered by Case 2 shall be represented by an `IEEE1451_Dot4PublicXdcr` **with** Type element equal to VECTOR. PublicXdcr with type equal to VECTOR will be referred hereafter as "Vector PublicXdcrs." **Common IEEE1451_Dot4Property objects** exposed by vector PublicXdcrs shall be represented by a Parameter element with a Vector or VectorSeries element. An **IEEE1451_Dot4Property** shall be defined as common if the exposed IEEE1451_Dot4Property is defined and exposed in the PropertyList element of all IEEE1451_Dot4Xdcrchannel objects being exposed by the IEEE1451_Dot4PublicXdcr object. IEEE1451_Dot4Propertys that are not common but exposed by a vector PublicXdcr shall be represented by a Parameter element with a Scalar or ScalarSeries element.

Case 3: IEEE1451_Dot4PublicXdcr instances grouping IEEE1451_Dot4XdcrChannel instances exposing IEEE1451_Dot4MMXdcr associated with the same IEEE1451_Dot4TransducerInterface instance.

IEEE1451_Dot4XdcrChannel exposing IEEE1451_Do4MMXdcr associated with the same IEEE1451_Dot4TransducerInterface shall contain the Attached Transducer Properties defined in B.6.1 and shall be exposed by an IEEE1451_Dot4PublicXdcr:Paramter with a Scalar or ScalarSeries element.

Rule 3:Mapping of exposed IEEE1451_Dot4Property to IEEE1451_Dot4PublicXdcr Parameter element

IEEE1451_Dot4PublicXdcr:Parameter elements shall represent exposed Properties in each XdcrChannel associated to the PublicXdcr. A PublicXdcr Parameter element can expose two types of IEEE1451_Dot4Property objects:

1) Information properties: Properties that contain information encoded in the TEDS.
2) Extended Functionality properties: Properties that are defined in the TEDS and control a particular configuration setting in an IEEE 1451.4 Transducer (i.e., gain, filter settings, etc.).

9.2.10.1.1 Parameter element representation of information properties

Information properties shall contain the most up-to-date value extracted from the TEDS. Properties that are associated with an Extended Functionality Control Property shall contain the most up-to-date value defined by the Control Property. See 7.4.9, associated Control Properties.

1. Information Properties that contain one IEEE1451_Dot4Property:Value element

Case 1: IEEE1451_Dot4PublicXdcr with Type element equal to SCALAR or with no grouping information.

Exposed IEEE1451_Dot4Property objects shall be represented by PublicXdcr:Parameter elements with a PublicXdcrParameter:Scalar element. The IEEE1451_Dot4PublicXdcr:Parameter:Scalar:Value element shall be set equal to the exposed IEEE1451_Dot4Propertys' IEEE1451_Dot4Property:Value element.

The Parameter element's metadata shall be set according to the following restrictions:

MetaData:ParameterName element shall be set equal to the exposed IEEE1451_Dot4Property object Name element.

MetaData:ParameterInterpretation element shall be set to PI_USER_DEFINED.

MetaData:Buffering element shall be set to PB_NORMAL; otherwise, it may be set by subclass specific means.

MetaData:Active element shall be set to true if the exposed IEEE1451_DotProperty's Disabled element has a value of false. Otherwise, it shall have a value of false.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

MetaData:Permission element shall be set to a value of "Read+Write" if the exposed IEEE1451_Dot4Property AccessLevel element has a value of "USR" or "MANF." Otherwise, it shall have a value of "Read."

MetaData:DescriptionList:Description The MetaData:DescriptionList:element sequence shall contain one MetaData:DescriptionList:Description element with value equal to the exposed IEEE1451_Dot4Propertys' IEEE1451_Dot4Property:Description value.

MetaData:QuantizationStepList:QuantizationStep The MetaData:QuantizationStepList element sequence shall contain one MetaData:QuantizationStepList:QuantizationStep element with value equal to the exposed IEEE1451_Dot4Propertys' IEEE1451_Dot4Property:Quantizationstep element value.

MetaData:UnitsList:Units The MetaData:unitsList element shall contain one MetaData:UnitsList:Units element with value equal to the exposed IEEE1451_Dot4Property:Units element value.

MetaData:UpperLimitList:UpperLimit The MetaData:UpperLimitList element shall contain one MetaData:UpperLimitList:UpperLimit element with value equal to the exposed IEEE1451_Dot4Property:MaxValue element. If the exposed Property has a Value element of type xs:string, then MetaData:Range:LowerLimit shall contain the maximum number of characters that can be stored in the exposed IEEE1451_Dot4Property:Value element.

MetaData:LowerLimitList:LowerLimit The MetaData:LowerLimitList element shall contain one MetaData:LowerLimitList:LowerLimit element with value equal to the exposed IEEE1451_Dot4Property:MinValue element. If the exposed Property has a Value element of type xs:string, then MetaData:Range:LowerLimit shall contain the minimum number of characters than can be stored in the exposed IEEE1451_Dot4Property:Value element.

Case 2 and 3: IEEE1451_Dot4PublicXdcr with Type element value equal to VECTOR.

Common exposed IEEE1451_Dot4Propertys' Property:Value elements shall be represented by a corresponding sequence of Parameter:Vector:Value elements in a PublicXdcr. The order of the Parameter:Vector sequence Parameter:Vector:Value elements shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

For example, given an IEEE1451_Dot4PublicXdcr associated to two IEEE1451_Dot4XdcrChannels with XdcrChannelID values of XdcrChannelID1 and XdcrChannelID2, respectively, and defined in the PublicXdcr AssociatedXdcrChannels element in the same order, then the value of the first Parameter:Vector:Value element shall be equal to the value of the common IEEE1451_Dot4Property object's IEEE1451_Dot4Property:Value element associated with the IEEE1451_Dot4XdcrChannel with XdcrChannelID equal to XdcrChannelID1. The second Parameter:Vector:Value element shall be equal to the value of the common IEEE1451_Dot4Property object's IEEE1451_Dot4PropertyValue element associated with the IEEE1451_Dot4XdcrChannel with XdcrChannelID equal to XdcrChannelID2.

NOTE—A single Parameter element in a vector IEEE1451_Dot4PublicXdcr can represent a number of common IEEE1451_Dot4Property elements equal to the value of the IEEE1451_Dot4PublicXdcr:Dimension element.

For each common exposed IEEE1451_Dot4Property, there shall be a corresponding Parameter:MetaData element defined by the following restrictions:

MetaData:ParameterName element shall be set equal to the exposed common IEEE1451_Dot4Property object IEEE1451_Dot4Property:Name element.

MetaData:ParameterInterpretation element shall be set to PI_USER_DEFINED.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

MetaData:Buffering element shall be set to PB_NORMAL.

MetaData:Active element shall be set to true if all the IEEE1451_Dot4Property represented by Parameter element have Disabled elements with a value of false. Otherwise, it shall have a value of false.

MetaData:Permission element shall be set to a value of "Read+Write" if the any of the IEEE1451_Dot4Property represented by the Parameter element has an AccessLevel element with a value of "USR" or "MANF." Otherwise, it shall have a value of "Read."

MetaData:Dimension element shall be set equal to the number of XdrcChannelID elements defined in the IEEE1451_Dot4PublicXdcr:AssociatedXdcrChannel sequence.

MetaData:CoordinateSystem element shall be set equal to the value of the common exposed IEEE1451_Dot4Property:CoordinateSystem element whose exposing XdcrChannel XdcrChannelID element is defined first in the PublicXdcr:AssociatedXdcrChannel sequence element.

MetaData:QuantizationStepList:QuantizationStep: For each exposed IEEE1451_Dot4Property:QuantizationStep element, there shall be a corresponding MetaData:QuantizationList:QuantizationStep element of equal value. The order of the MetaData:QuantizationList:QuantizationStep elements in the MetaData:UpperLimitList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

MetaData:UpperLimitList:UpperLimit: For each exposed IEEE1451_Dot4Property:MaxValue element, there shall be a corresponding MetaData:UpperLimitList:UpperLimit element of equal value. The order of the MetaData:UpperLimitList:UpperLimit elements in the MetaData:UpperLimitList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

Range:LowerLimitList:LowerLimit: For each exposed IEEE1451_Dot4Property:MinValue element, there shall be a corresponding MetaData:LowerLimitList:LowerLimit element of equal value. The order of the MetaData:LowerLimitList:LowerLimit elements in the MetaData:LowerLimitList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

MetaData:UnitList:units: For each exposed IEEE1451_Dot4Property:Units element, there shall be a corresponding MetaData:unitsList:units element of equal value. The order of the MetaData:UnitList:units elements in the MetaData:unitList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

MetaData:DescriptionList:Description: For each exposed IEEE1451_Dot4Property:Description element, there shall be a corresponding MetaData:DescriptionList:Description element of equal value. The order of the MetaData:DescriptionList:Description elements in the MetaData:Description sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

MetaData:DescriptionList:Formatting: For each exposed IEEE1451_Dot4Property:Formatting element, there shall be a corresponding MetaData:FormattingList:Formatting element of equal value. The order of the MetaData: FormattingList: Formatting elements in the MetaData:Description sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

2. Information Properties that contain more than one IEEE1451_Dot4Property:Value:ValueArrayElement element

Case 1: IEEE1451_Dot4PublicXdcr with Type element equal to SCALAR or with no grouping information.

Exposed IEEE1451_Dot4Property objects with more than one IEEE1451_Dot4Property:Value:ValueArrayElement element shall be represented by PublicXdcr:Parameter elements with a PublicXdcr:Parameter:ScalarSeries element. The IEEE1451_Dot4PublicXdcr:Parameter:ScalarSeries:Scalar:Value element shall be set equal to the exposed IEEE1451_Dot4Propertys' IEEE1451_Dot4Property:Value element. The Parameter:ScalarSeries:Scalar:Value:ArrayLength shall have a value equal to the total number of Parameter:ScalarSeries:Scalar:Value:ArrayElement elements.

The Parameter element's metadata shall be set according to the following restrictions:

MetaData:ParameterName element shall be set equal to the exposed IEEE1451_Dot4Property object Name element.

MetaData:ParameterInterpretation element shall be set to PI_USER_DEFINED.

MetaData:Buffering element shall be set to PB_NORMAL.

MetaData:Active element shall be set to true if the exposed IEEE1451_DotProperty's Disabled element has a value of false. Otherwise, it shall have a value of false.

MetaData:Permission element shall be set to a value of "Read+Write" if the exposed IEEE1451_Dot4Property AccessLevel element has a value of "USR" or "MANF." Otherwise, it shall have a value of "Read."

MetaData:QuantizationStepList:QuantizationStep The MetaData:QuantizationStepList element shall contain one MetaData:QuantizationStepList:QuantizationStep element with value equal to the exposed IEEE1451_Dot4Propertys' IEEE1451_Dot4Property:Quantizationstep element value.

MetaData:UnitsList:Units The MetaData:unitsList element shall contain one MetaData:UnitsList:Units element with value equal to the exposed IEEE1451_Dot4:Units element value.

MetaData:UpperLimitList:UpperLimit The MetaData:UpperLimitList element shall contain one MetaData:UpperLimitList:UpperLimit element with value equal to the exposed IEEE1451_Dot4Property:MaxValue element. If the exposed Property has a Value element of type xs:string then MetaData:Range:LowerLimit shall contain the maximum number of characters that can be stored in the exposed IEEE1451_Dot4Property:Value element.

MetaData:LowerLimitList:LowerLimit The MetaData:LowerLimitList element shall contain one MetaData:LowerLimitList:LowerLimit element with value equal to the exposed IEEE1451_Dot4Property:MinValue element. If the exposed Property has a Value element of type xs:string then MetaData:Range:LowerLimit shall contain the minimum number of characters that can be stored in the exposed IEEE1451_Dot4Property:Value element.

Case 2 and 3: IEEE1451_Dot4PublicXdcr with Type element value equal to VECTOR exposing Information Properties that contain more than one IEEE1451_Dot4Property:Value:ValueArrayElement element.

Common exposed IEEE1451_Dot4Property with more than one IEEE1451_Dot4Property:Value:ValueArrayElement element shall be represented by

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

IEEE1451_Dot4PublicXdcr with a Parameter:VectorSeries element. Each common exposed IEEE1451_Dot4Propertys' Property:Value elements shall be represented by a corresponding sequence of Parameter:VectorSeries:Vector:Member:Value elements in a PublicXdcr. The order of the Parameter:VectorSeries:Vector sequence Parameter:VectorSereis:Vector:Value elements shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

For example, given an IEEE1451_Dot4PublicXdcr associated to two IEEE1451_Dot4XdcrChannels with XdcrChannelID values of XdcrChannelID1 and XdcrChannelID2, respectively, and defined in the PublicXdcr AssociatedXdcrChannels element in the same order, then the value of the first Parameter:VectorSeries:Vector:Value element shall be equal to the value of the common IEEE1451_Dot4Property object's IEEE1451_Dot4Property:Value element associated with the IEEE1451_Dot4XdcrChannel with XdcrChannelID equal to XdcrChannelID1. The second Parameter:VectorSeries:Vector:Member:value element shall be equal to the value of the common IEEE1451_Dot4Property object's IEEE1451_Dot4PropertyValue element associated with the IEEE1451_Dot4XdcrChannel with XdcrChannelID equal to XdcrChannelID2. All exposed IEEE1451_Dot4Property objects shall have an equal number of Value:ValueArrayElements elements.

NOTE—A single Parameter element in a vector IEEE1451_Dot4PublicXdcr can represent a number of common IEEE1451_Dot4Property elements equal to the value of the IEEE1451_Dot4PublicXdcr:Dimension element.

For each common exposed IEEE1451_Dot4Property, there shall be a corresponding Parameter:MetaData element defined by the following restrictions:

MetaData:ParameterName element shall be set equal to the exposed common IEEE1451_Dot4Property object IEEE1451_Dot4Property:Name element.

MetaData:ParameterInterpretation element shall be set to PI_USER_DEFINED.

MetaData:Buffering element shall be set to PB_NORMAL.

MetaData:Active element shall be set to true if all the IEEE1451_Dot4Property represented by Parameter element have Disabled elements with a value of false. Otherwise, it shall have a value of false.

MetaData:Permission element shall be set to a value of "Read+Write" if the any of the IEEE1451_Dot4Property represented by the Parameter element has an AccessLevel element with a value of "USR" or "MANF." Otherwise, it shall have a value of "Read."

MetaData:Dimension element shall be set equal to the number of XdrcChannelID elements defined in the IEEE1451_Dot4PublicXdcr:AssociatedXdcrChannel sequence.

MetaData:CoordinateSystem element shall be set equal to the value of the common exposed IEEE1451_Dot4Property:CoordinateSystem element whose exposing XdcrChannel XdcrChannelID element is defined first in the PublicXdcr:AssociatedXdcrChannel sequence element.

MetaData:QuantizationStepList:QuantizationStep: For each exposed IEEE1451_Dot4Property:QuantizationStep element, there shall be a corresponding MetaData:QuantizationList:QuantizationStep element of equal value. The order of the MetaData:QuantizationList:QuantizationStep elements in the MetaData:UpperLimitList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

MetaData:UpperLimitList:UpperLimit: For each exposed IEEE1451_Dot4Property:MaxValue element, there shall be a corresponding MetaData:UpperLimitList:UpperLimit element of equal value. The order of the MetaData:UpperLimitList:UpperLimit elements in the MetaData:UpperLimitList sequence element

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

Range:LowerLimitList:LowerLimit: For each exposed IEEE1451_Dot4Property:MinValue element, there shall be a corresponding MetaData:LowerLimitList:LowerLimit element of equal value. The order of the MetaData:LowerLimitList:LowerLimit elements in the MetaData:LowerLimitList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

MetaData:UnitList:units: For each exposed IEEE1451_Dot4Property:Units element, there shall be a corresponding MetaData:unitsList:units element of equal value. The order of the MetaData:UnitList:units elements in the MetaData:unitList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

MetaData:DescriptionList:Description: For each exposed IEEE1451_Dot4Property:Description element, there shall be a corresponding MetaData:DescriptionList:Description element of equal value. The order of the MetaData:DescriptionList:Description elements in the MetaData:Description sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

MetaData:DescriptionList:Formatting: For each exposed IEEE1451_Dot4Property:Formatting element, there shall be a corresponding MetaData:FormattingList:Formatting element of equal value. The order of the MetaData: FormattingList: Formatting elements in the MetaData:Description sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

9.2.10.1.2 Exposing Extended Functionality Control Properties

Extended Functionality IEEE1451_Dot4Property objects control a particular configuration setting in an IEEE 1451.4 Transducer (i.e., gain, filter settings, etc.). IEEE Std 1451.4-2004 defines two types of Extended Functionality properties: Control Properties that directly control the underlying hardware by setting the value of its IEEE1451_Dot4Property:Value element and associated properties whose IEEE1451_Dot4Property:Value element is a function of the IEEE 1451.4 Transducer FR value. Both types of Extended Functionality properties shall be represented by an IEEE1451_Dot4PublicXdcr:Parameter element with Parameter:MetaData:ExtenededFNList:ExtendedFN element with value equal to the exposed IEEE1451_Dot4Property:ExtendedFN _MetaData element value.

Case 1: Extended Functionality IEEE1451_Dot4Property objects exposed by an IEEE1451_Dot4PublicXdcr with Type element equal to SCALAR or with no grouping information.

Exposed IEEE1451_Dot4Property objects shall be represented by PublicXdcr:Parameter elements with a PublicXdcr:Parameter:Scalar element. Upon reception of a RESET_EVENT or a REFRESH_EVENT by the IEEE1415_Dot4TransducerBlock, the IEEE1451_Dot4PublicXdcr:Parameter:Scalar:Value element shall be set equal to the value of the IEEE1451_Dot4PublicXdcr:Parameter:MetaData:ExtendFnList:ExtendedFn:FunctionalStateList:FunctionState whose Default element has a value of true by subclass specific means. Otherwise, the Parameter element representing the exposed IEEE1451_Property shall have a Parameter:Scalar:Value equal to the exposed IEEE1451_Dot4Propertys' IEEE1451_Dot4Property:Value element it is representing. The Parameter:Scalar:Value element of the Parameter representing the exposed Property shall be settable to any of the Parameter:MetaData:ExtendedFnList:ExtendedFn:FunctionalStateList:FunctionState PropertyValue elements whose Active element has a value of true.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

The Parameter element representing the exposed IEEE1451_Property shall have a Parameter:Scalar:Value element settable to any of the Parameter:MetaData:ExtendedFnList:ExtendedFn:FunctionalStateList:FunctionState:PropertyValue element values defined in the Parameter:MetaData:ExtendedFnList:ExtFn:FunctionalStateList sequence. If the exposed IEEE1451_Dot4Property:ExtendedFn:PropertyType is equal to associated property, the Property:value shall be set by subclass specific means to match the value defined by the current value of the IEEE1451 Transducer Functional Register.

The Parameter element's metadata shall be set according to the following restrictions:

MetaData:ParameterName element shall be set equal to the exposed IEEE1451_Dot4Property object Name element.

MetaData:ParameterInterpretation element shall be set to PI_CONTROL_SETPOINT.

MetaData:Active element shall be set to true if the exposed IEEE1451_DotProperty's Disabled element has a value of false. Otherwise, it shall have a value of false.

MetaData:Permission element shall be set to a value of "Read+Write" if the exposed IEEE1451_Dot4Property AccessLevel element has a value of "USR" or "MANF." Otherwise, it shall have a value of "Read." If the exposed Extended Functionality property is an associated Control Property, then the Parameter element representing it shall have a MetaData:Permission element with a value of "Read."

MetaData:DescriptionList:Description The MetaData:DescriptionList:element sequence shall have one MetaData:DescriptionList:Description element whose value is equal to the Parameter:MetaData:ExtendedFnList:ExtendedFn:FunctionalStateList:FunctionState Description element whose PropertyValue element matches the Parameter:Scalar:Value element.

MetaData:UnitsList:Units The MetaData:UnitsList element shall contain one MetaData:UnitsList:Units element with value equal to the exposed IEEE1451_Dot4:Units element value.

Case 2 and 3: Extended Functionality IEEE1451_Dot4Property objects exposed by an IEEE1451_Dot4PublicXdcr with Type element equal to VECTOR.

Common exposed Extended Functionality IEEE1451_Dot4Propertys' Property:Value elements shall be represented by a corresponding sequence of Parameter:Vector:Value elements in a PublicXdcr. The order of the Parameter:Vector sequence Parameter:Vector:Value elements shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

For example, given an IEEE1451_Dot4PublicXdcr associated to two IEEE1451_Dot4XdcrChannels with XdcrChannelID values of XdcrChannelID1 and XdcrChannelID2, respectively, and defined in the PublicXdcr AssociatedXdcrChannels element in the same order, then the value of the first Parameter:Vector:Value element shall be equal to the value of the common IEEE1451_Dot4Property object's IEEE1451_Dot4Property:Value element associated with the IEEE1451_Dot4XdcrChannel with XdcrChannelID equal to XdcrChannelID1. The second Parameter:Vector:Value element shall be equal to the value of the common IEEE1451_Dot4Property object's IEEE1451_Dot4PropertyValue element associated with the IEEE1451_Dot4XdcrChannel with XdcrChannelID equal to XdcrChannelID2.

Upon reception of a RESET_EVENT or a RefreshEvent by the IEEE1415_Dot4TransducerBlock, each Parameter:Vector:Value element shall be set equal to the exposed IEEE1451_Dot4Property PropertyValue element defined in the IEEE1451_Dot4Property:ExtFn_MetaData:FunctionalStateList:FunctionalState whose Default element has a value of true. Each Parameter:Vector:Value element of the Parameter representing an exposed common IEEE1451_Dot4Property shall be settable to any of the

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Parameter:MetaData:ExtendedFnList:ExtendedFn:FunctionalStateList FunctionState:PropertyValue elements whose Active element has a value of true.

NOTE—A single Parameter element in a vector IEEE1451_Dot4PublicXdcr can represent a number of common IEEE1451_Dot4Property elements equal to the value of the IEEE1451_Dot4PublicXdcr:Dimension element.

For each common exposed IEEE1451_Dot4Property, there shall be a corresponding Parameter:MetaData element defined by the following restrictions:

MetaData:ParameterName element shall be set equal to the exposed common IEEE1451_Dot4Property object IEEE1451_Dot4Property:Name element.

MetaData:ParameterInterpretation element shall be set to PI_CONTROL_SETPOINT.

MetaData:Buffering element shall be set to PB_NORMAL.

MetaData:Active element shall be set to true if all the IEEE1451_Dot4Property represented by Parameter element have Disabled elements with a value of false. Otherwise, it shall have a value of false.

MetaData:Permission element shall be set to a value of "Read+Write" if the any of the IEEE1451_Dot4Property represented by the Parameter element has an AccessLevel element with a value of "USR" or "MANF." Otherwise, it shall have a value of "Read." If the exposed common Extended Functionality property is an associated Control Property then the parameter element representing it shall have a MetaData:Permission element with a value of "Read."

MetaData:Dimension element shall be set equal to the number of XdrcChannelID elements defined in the IEEE1451_Dot4PublicXdcr:AssociatedXdcrChannel sequence.

MetaData:CoordinateSystem element shall be set equal to the value of the common exposed IEEE1451_Dot4Property:CoordinateSystem element whose exposing XdcrChannel XdcrChannelID element is defined first in the PublicXdcr:AssociatedXdcrChannel sequence element.

MetaData:UnitList:units: For each exposed IEEE1451_Dot4Property:Units element, there shall be a corresponding MetaData:unitsList:units element of equal value. The order of the MetaData:UnitList:units elements in the MetaData:unitList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

MetaData:DescriptionList:Description: For each exposed IEEE1451_Dot4Property:Description element, there shall be a corresponding MetaData:DescriptionList:Description element of equal value. The order of the MetaData:DescriptionList:Description elements in the MetaData:DescriptionList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence. The value of the MetaData:DescriptionList:Description element for each Parameter:Description element representing an exposed property shall be equal to the Parameter:MetaData:ExtendedFnList:ExtendedFn:FunctionalStateList:FunctionState Description element whose PropertyValue element matches the Parameter:Vector:Value element.

MetaData:DescriptionList:Formatting: For each exposed IEEE1451_Dot4Property:Formatting element, there shall be a corresponding MetaData:FormattingList:Formatting element of equal value. The order of the MetaData: FormattingList: Formatting elements in the MetaData:FormattingList sequence element shall be determined by the order in which the XdcrChannel:XdcrChannelID exposing the common IEEE1451_Dot4Property is defined in the PublicXdcr:AssociatedXdcr element sequence.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.3 Common Object Interface (COI) specification

The COI defines an optional abstraction layer that provides control and event generation capabilities to TBOM components. The COI consists of two mechanisms:

1) A messaging mechanism that simplifies the integration and control of MMI hardware from different manufacturers into the TransducerBlock.
2) An event generation and handling mechanism that provides support for IEEE 1451.4 systems with hotswap, refresh, and error notification capabilities.

The IEEE1451_Dot4CommandSource, IEEE1451_Dot4CommandSink, IEEE1451_Dot4EventSource, and IEEE1451_Dot4EventSink classes shall add the COI functionality to TBOM classes. A summary of TBOM classes that are COI capable can be found in Table 26. Associated data models MessageType and EventType define the semantics and constructs of messages and events. Definitions can be found in Annex L.

Table 26—COI capable classes

Object	IEEE1451_Dot4CommandSource	IEEE1451_Dot4CommandSink	IEEE1451_Dot4EventSource	IEEE1451_Dot4EventSink
IEEE1451_Dot4TransducerBlock	X	—	—	X
IEEE1451_Dot4TransducerInterface	—	X	X	—

9.3.1 IEEE1451_Dot4CommandSource and IEEE1451_Dot4CommandSink

The IEEE1451_Dot4CommandSource and IEEE1451_Dot4CommandSink classes shall provide the necessary mechanisms to generate, validate, and process messages among TBOM components.

9.3.1.1 IEEE1451_Dot4CommandSource

The `IEEE1451_Dot4CommandSource` shall generate commands restricted by relationships defined by the class diagrams in Figure 28 and Figure 30.

The `IEEE1451_Dot4CommandSource` shall observe the behavior defined by the behavioral diagram in Figure 29. The `IEEE1451_Dot4CommandSource` object shall be responsible for generating request messages compliant with semantics defined by the IEEE1451_Dot4MessageType data model in Annex L and 9.3.1.3. The specific mechanisms to transmit messages are outside the scope of this standard.

The `IEEE1451_Dot4CommandSource` local operations InvokeOperationByName and InvokeOperationByID shall provide a mechanism to invoke remote operations on an object associated with an `IEEE1451_Dot4CommandSink` object (i.e., `IEEE1451_Dot4TransducerInterface`). Table 27 defines a list of valid operations. Local `IEEE1451_Dot4CommandSource` operations shall only be honored when the `IEEE1451_Dot4CommandSource` object is in the Idle state (i.e., IEEE1451_Dot4CommandSource:State element has a value of IDLE). If the IEEE1451_Dot4CommandSource is not in Idle state, then the local operation OpReturnCode major field shall have a value of MJ_BUSY. When a local operation is invoked, the `IEEE1451_Dot4CommandSource` object state machine shall undergo the GenerateCommand transition

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

from its Idle state to the GenerateExecuteCommand state. In the GenereateExecuteCommand state, the `IEEE1451_Dot4CommandSource` object shall generate a well-formed request message and await synchronously for a response message containing the results of the remote operation as defined by the IEEE1451_Dot4MessageType data model in Annex L.

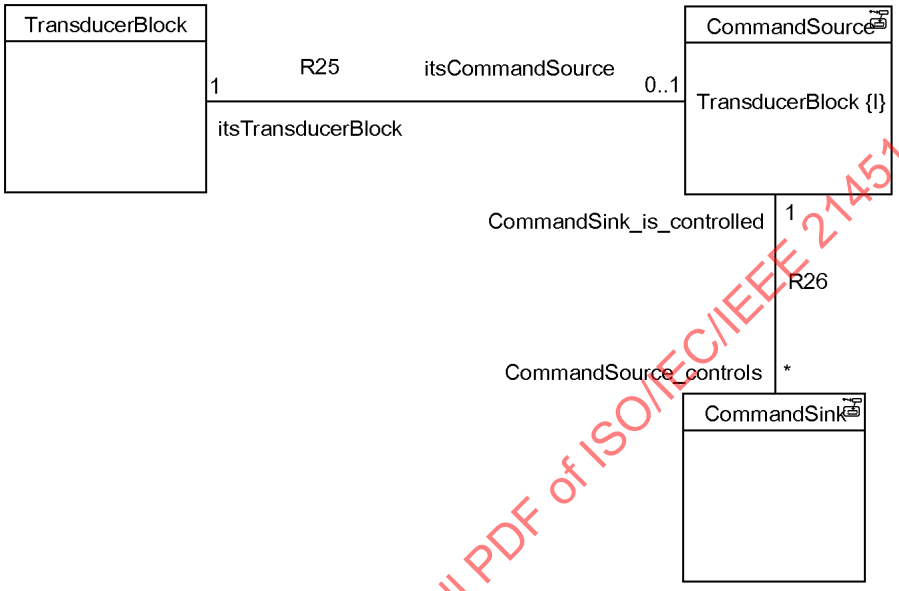


Figure 28—IEEE1451_Dot4CommandSource associations

Once a valid response message has been received, the `IEEE1451_Dot4CommandSource` object shall transition to the Idle state. The OpReturnCode of the invoking operation shall contain a MJ_COMPLETE value in its major field. If the `IEEE1451_Dot4CommandSource` object times out or is otherwise unable to receive a valid response message, it shall transition to ERROR_STATE and shall remain there until its reset by subclass specific means. The OpReturnCode of the invoking local operation shall contain a valid error in its major field as defined in IEEE 1451.1-1999, Table 10.

Table 27—Valid message operation ID

Message operations	Operation ID
GetTransducerInterfaceStatus	0
SetTransducerInterfaceStatus	1
GetTransducerInterfaceEventMask	2
SetTransducerInterfaceEventMask	3
GetTransducerInterfaceMode	4
SetTransducerInterfaceMode	5
LockTransducerInterfaceMode	6
UnLockTransducerInterfaceMode	7
GetTransducerInterfaceClass	8

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table 27—Valid message operation ID *(continued)*

Message operations	Operation ID
ResetTransducerInterface	9
RefreshTransducerInterface	10
SelfTestTransducerInterface	11
GetNumberOfMMXdcrsInTransducerInterface	12
GetMMXdcrListInTransducerInterface	13
GetNumberOfNodesInTransducerInterface	14
GetNodeListInTransducerInterface	15
TunnelTransparentProtocol	16
TunnelProtocol	17

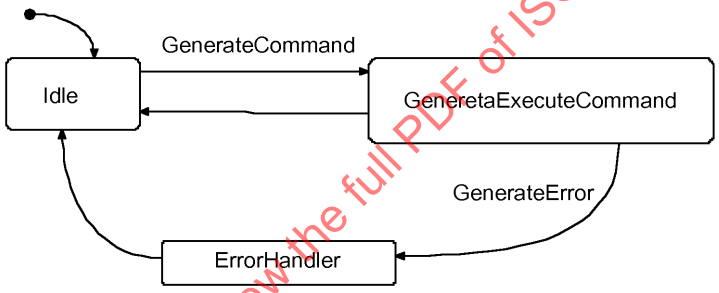


Figure 29—IEEE1451_Dot4CommandSource behavioral model

9.3.1.2 IEEE1451_Dot4CommandSink

The `IEEE1451_Dot4CommandSink` shall process all request messages addressed to its associated object (i.e., TransducerInterface) and be restricted by the relationships defined in the class diagrams in Figure 28 and Figure 30.

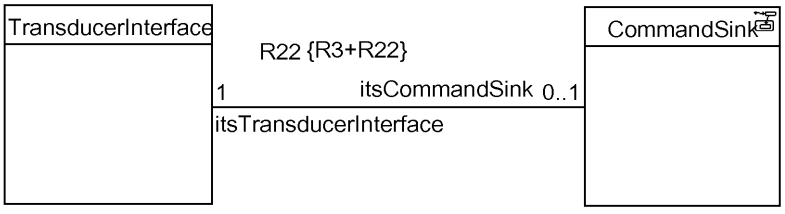


Figure 30—IEEE1451_Dot4CommandSink associations

The `IEEE1451_Dot4CommandSink` shall observe behavior defined by Figure 31. The `IEEE1451_Dot4CommandSink` object is responsible for generating a response message with semantics defined by the IEEE1451_Dot4MessageType data model in Annex L and 9.3.1.3. The specific mechanisms to transmit messages are outside the scope of this standard.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

The IEEE1451_Dot4CommandSink has no local operations defined. Request messages shall only be honored when the IEEE1451_Dot4CommandSink object state machine is in the Idle state (i.e., IEEE1451_Dot4CommandSink:State element has a value of IDLE). Once a request message is received, the IEEE1451_Dot4CommandSink shall undergo the ExecuteOperation transition from its Idle state to the ExecutingOperation state, process the request message, and invoke the corresponding operation on its associated object by subclass specific means.

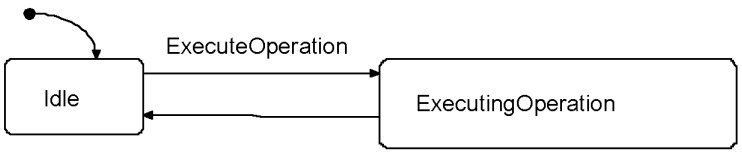


Figure 31—IEEE1451_Dot4CommandSink behavioral model

The IEEE1451_Dot4CommandSink shall remain in the ExecutingOperation state until the associated object completes the request operation and the IEEE1451_Dot4CommandSink generates a response message with its major and minor OpReturnCode elements set equal to those of the requested operation. The IEEE1451_Dot4CommandSink shall then transition to the Idle state. If the associated object or the IEEE1451_Dot4CommandSink is unable to honor the requested operation, then the IEEE1451_Dot4CommandSink shall generate an appropriate response message containing an OpReturnCode describing the error. Additionally, the IEEE1451_Dot4TransducerInterface:Status:Invalid_Command element of the IEEE1451_Dot4TransducerInterface associated to the IEEE1451_Dot4CommandSink shall be set to a value of true. It shall then transition to the Idle state.

9.3.1.3 IEEE1451_Dot4MessageType element semantics

The IEEE1451_Dot4MessageType represents the data model for messages transmitted between IEEE1451_Dot4CommandSource and IEEE1451_Dot4CommandSink instances. The Message semantics are defined in Annex L. The IEEE1451_Dot4MessageType contains two root elements, the Header and Body elements.

9.3.1.3.1 Header element

The Header element contains three subelements: A Header:Source element containing the identifier attribute of the object owning the IEEE1451_Dot4CommandSource (i.e., the TransducerBlockID); a Header:Target element containing the identifier of the object owning the IEEE1451_Dot4CommandSink that the message is addressed to (i.e., the MMIID of the TransducerInterface); and a MessageID element that is assigned by subclass specific means to uniquely identify the transaction and that shall be the same for the request and response message pair.

9.3.1.3.2 Body element

The Body element shall contain an OperationRequest element if the message originated from an IEEE1451_Dot4CommandSource (i.e., request message). The OperationRequest element shall contain either an OperationName or OperationID with a valid operation code defined in Table 26 and an ArgumentList element containing a list of input arguments corresponding to the operation. The operation shall have the same argument names as those defined in 9.5.

The Body attribute shall contain a Body:OperationType element with a value of “RESPONSE” if the message originated from an IEEE1451_Dot4CommandSink or a value of “REQUEST” if the message originated from an IEEE1451_Dot4CommandSource. The Body:Operation element shall contain either

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

an OperationName or OperationID element with a valid operation code defined in Table 27: A MessageID element that should be the same as the MessageID from the OperationRequest message; an OpReturnCode element detailing the result of the invoked operation; and an ArgumentList containing a list of the input/output arguments of the operation. The operations defined in Table 27 shall have the same argument names as those defined in 9.5.

9.3.1.4 IEEE1451_Dot4CommandSource and IEEE1451_Dot4CommandSink local operations

9.3.1.4.1 IEEE1451_Dot4CommandSource local operations

Table 28—CommandSource local operations

Operation	Requirement
InvokeOperationByID	(m)
InvokeOperationByName	(m)
Reset	(m)

9.3.1.4.1.1 Operation InvokeOperationByID

Description: Operation InvokeOperationByID shall invoke an operation on an object associated with an `IEEE1451_Dot4CommandSink` object (i.e., an IEEE1451_Dot4`TransducerInterface` object associated with an `IEEE1451_Dot4CommandSink` object). Valid operations are defined in Table 27.

```
OpReturnCode InvokeOperationByID {
[in] xs:hexBinary MMIID,
[in] xs:unsignedInt OperationID,
[in] IEEE1451_Dot4:IEEE1451_Dot4AnyTypeList ArgumentListIn
[out] IEEE1451_Dot4:AnyTypeList ArgumentListOut}
```

9.3.1.4.1.1.1 Argument MMIID

Argument MMIID shall uniquely identify the TransducerInterface object being addressed by the operation.

9.3.1.4.1.1.2 Argument OperationID

Argument OperationID shall be a valid OperationID defined in Table 27.

9.3.1.4.1.1.3 Argument ArgumentListIn

ArgumentListIn shall contain a list of valid input arguments for the operation identified by OperationID. OperationID shall have the same argument names as those defined in 9.5 and Annex L.

9.3.1.4.1.1.4 Argument ArgumentListOut

ArgumentListOut shall contain a list of valid output arguments for the operation identified by OpeationID. OperationID shall have the same argument names as those defined in 9.5 and Annex L.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.3.1.4.1.2 Operation InvokeOperationByName

Description: Operation InvokeOperationByName shall invoke an operation on an object associated with an `IEEE1451_Dot4CommandSink` object (i.e., an IEEE1451_Dot4`TransducerInterface` object associated with an `IEEE1451_Dot4CommandSink` object). Valid operations are defined in Table 27.

```
OpReturnCode InvokeOperationByName {
[in] xs:hexBinary MMIID,
[in] xs:string OperationName,
[in] IEEE1451_Dot4:IEEE1451_Dot4AnyTypeList ArgumentListIn
[out] IEEE1451_Dot4:IEEE1451_Dot4AnyTypeList ArgumentListOut}
```

9.3.1.4.1.2.1 Argument MMIID

Argument MMIID shall uniquely identify the TransducerInterface object being addressed by the operation.

9.3.1.4.1.2.2 Argument OperationName

Argument OperationName shall be a valid operation name defined in Table 27.

9.3.1.4.1.2.3 Argument ArgumentListIn

ArgumentListIn shall contain a list of valid input arguments as to the operation identified by OperationName. OperationName shall have the same argument names as those defined in 9.5 and Annex L.

9.3.1.4.1.2.4 Argument ArgumentListOut

ArgumentListOut shall contain a list of valid output arguments for the operation identified by OperationName. OperationName shall have the same argument names as those defined in 9.5 and Annex L.

9.3.1.4.1.3 Operation Reset

Description: Operation Reset shall reset the `IEEE1451_Dot4CommandSink` to its Idle state. All pending operations shall be disposed.

```
OpReturnCode InvokeOperationByName {}
```

9.3.1.4.2 IEEE1451_Dot4CommandSink local operations

No local operation is defined.

9.3.2 IEEE1451_Dot4EventSource and IEEE1451_Dot4EventSink

The `IEEE1451_Dot4EventSource` and `IEEE1451_Dot4EventSink` classes shall provide the necessary mechanisms to generate, validate, and process events within IEEE 1451.4 TBOM classes. The IEEE1451.4 TBOM supports four types of events: HOTSWAP_EVENT, RESET_EVENT, REFRESH_EVENT, and ERROR_EVENT.

9.3.2.1 IEEE1451_Dot4EventSource

The `IEEE1451_Dot4EventSource` shall generate events restricted by relationships defined by the class diagram in Figure 32. Events shall be asynchronous and shall be broadcasted to all `IEEE1451_Dot4EventSink` objects associated with the IEEE1451_Dot4TransducerBlock. Events shall follow the semantics defined by the IEEE1451_Dot4EventType data model in Annex L.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

The IEEE1451_Dot4EventSource local operation GenerateEvent shall provide a mechanism to generate and broadcast events to all IEEE1451_Dot4EventSink instances associated with the IEEE1451_Dot4TransducerBlock. The mechanics of broadcasting and delivering events are outside the scope of this standard.

The IEEE1451_Dot4EventSource shall observe behavior defined by the behavioral diagram in Figure 33. When local operation GenerateEvent is invoked, the IEEE1451_Dot4EventSource shall undergo the EventGeneration transition from its Idle state to its EventGeneration state. Once the message has been broadcasted, the IEEE1451_DotEventSource shall return to its Idle state. If local operation GenerateEvent is successful, its OpReturnCode major field shall have a value of MJ_COMPLETE. If local operation GenerateEvent is unable to generate or broadcast the event, then its OpReturnCode shall have a value of MJ_NOP_OPERATION and MJ_COMMUNCIATION_ERROR, respectively.

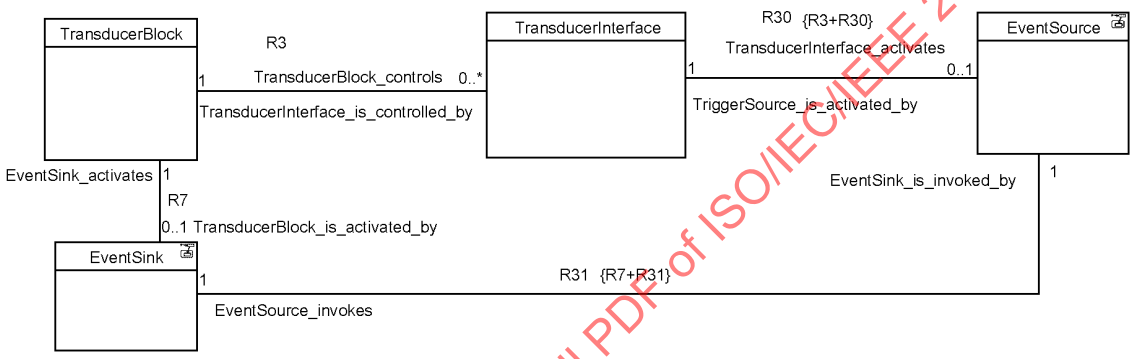


Figure 32—IEEE1451_Dot4EventSource associations

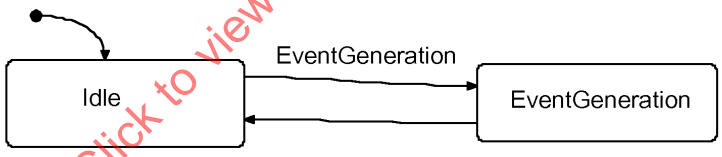


Figure 33—IEEE1451_Dot4EventSource behavior diagram

9.3.3 IEEE1451_Dot4EventSink

Class IEEE1451_Dot4EventSink shall receive and process events restricted by the relationships defined in the class diagram in Figure 34.

The IEEE1451_Dot4EventSink shall observe the behavior defined by the behavioral diagram in Figure 35. Events shall be honored only when the IEEE1451_Dot4EventSink is in the Idle state. When an event is received, the IEEE1451_Dot4EventSink state machine shall undergo the ResvEvent transition to the EventReception state and notify its associated object by subclass specific means. In case the event received by the IEEE1451_Dot4EventSink can not be validated, then it shall recover by class specific mean and return to the IDLE state. The IEEE1451_Dot4EventSink has no local operations.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

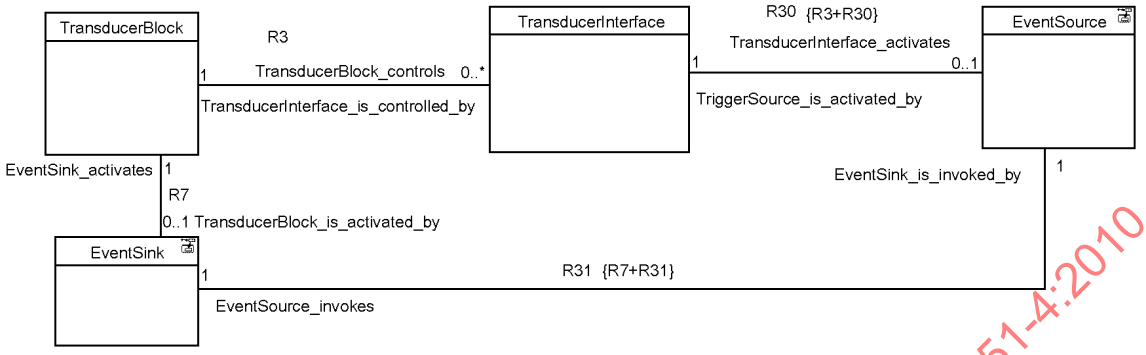


Figure 34—IEEE1451_Dot4EventSink associations

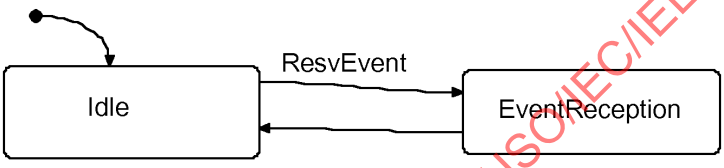


Figure 35—IEEE1451_Dot4EventSink behavioral diagram

9.3.4 IEEE 1451.4 supported events

The IEEE1451.4 TBOM supports four types of events: HOTSWAP_EVENT, RESET_EVENT REFRESH_EVENT, and ERROR_EVENT. Events shall be asynchronous and broadcasted to all `IEEE1451_Dot4EventSink` instances associated with TBOM components.

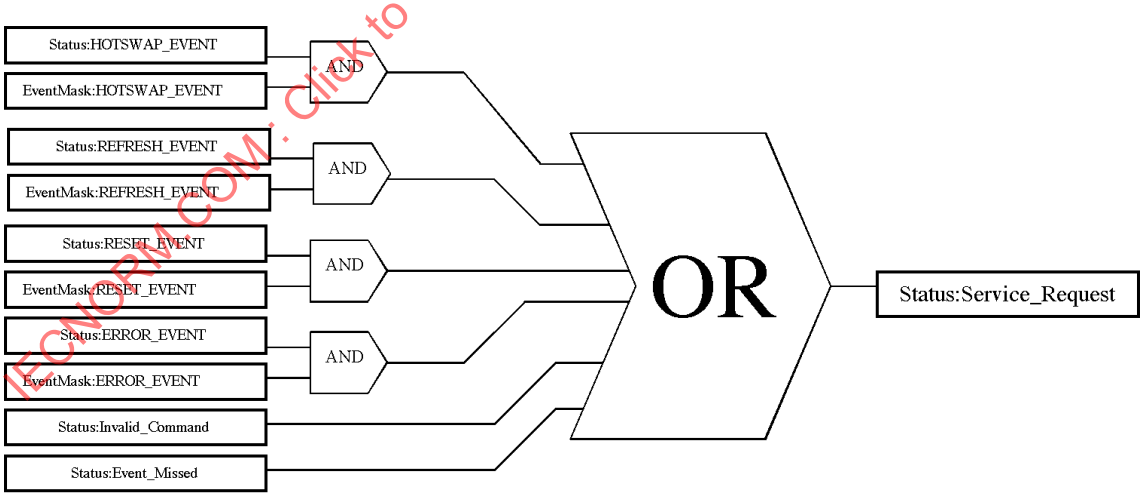


Figure 36—IEEE1451_Dot4TransducerInterface:Status element update operation

9.3.4.1 HOTSWAP_EVENT

HOTSWAP_EVENT shall be raised when an IEEE1451.4 Transducer is connected or removed from the MMI. The electrical means for detecting a HOTSWAP are outside the scope of this standard and left open to the manufacturer. The EventSource associated to an `IEEE1451_Dot4TransducerInterface` shall

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

generate the HOTSWAP_EVENT. Upon reception of a HOTSWAP_EVENT by an `IEEE1451_Dot4TransducerBlock`, it shall update the content of its associated IEEE1451_Dot4PublicXdcr `objects` and its IEEE1451_Dot4TransducerBlock:Status element to reflect the changes.

The steps to generate and detect a HOTSWAP_EVENT are as follows:

1) A 1451.4 Transducer is physically connected or removed from an MMI.
2) The MMI detects that one or more 1451.4 Transducers have been connected or removed.
3) The IEEE1451_Dot4TransducerInterface object associated to the MMI
   a) Sets its IEEEE1451_Dot4TransducerInterface:Status:HOTSWAP_EVENT element to true.
   b) Sets its IEEE1451_Dot4TransducerInterface:Status:Event_Missed element to true if the IEEE1451_Dot4TransducerInterface:Status:Service_Request element has a value of true. Otherwise, it set its Status:Event_Missed to false.
   c) Updates its IEEE1451_Dot4TransdcuerInterface:Status:Service_Request element performing the logical operation defined in Figure 36.
4) The `IEEE1451_Dot4TransducerInterface` object starts discovery on the physical MMI, and updates its associated `IEEE1451_Dot4MMXdcrs` and `IEEE1451_Dot4MMXdcrChannel` objects by subclass specific means.
   a) If an `IEEE1451_Dot4MMXdcr` is no longer associated to a physical 1451.4 Transducer (i.e., the physical transducer has been removed), then the `IEEE1451_Dot4MMXdcr` and `IEEE1451_Dot4XcdrChannel` objects data models Active element associated with the IEEE 1451.4 Transducer shall be set to false and shall not be removed from the `system`. The `IEEE1451_Dot4MMXdcr` and `IEEE1451_Dot4XcdrChannel` shall only be removed from the TBOM when the `IEEE1451_Dot4TransducerBlock` is reset.
   b) If a new IEEE 1451.4 Transducer has been added, then
      i) The IEEE1451_Dot4XdcrChannel BasicTEDSID element representing the newly added IEEE1451.4 Transducer shall be added to the IEEE1451_TransducerInterface AssociatedMMXdcr element sequence.
      ii) The IEEE1451_Dot4XdcrChannel objects XdcrChannelID element representing the newly added IEEE1451_Dot4MMXdcr shall be added to IEEE1451_TransducerInterface: AssociatedXdcrChannels element sequence.
5) The `IEEE1451_Dot4TransducerInterface` shall broadcast a HOTSWAP_EVENT addressed to all IEEE1451_Dot4EventSink objects in the TBOM.
6) The `IEEE1451_Dot4TransducerBlock`, upon reception of the HOTSWAP_EVENT, shall
   a) Update its associated `IEEE1451_Dot4PublicXdcr` by subclass specific means (excluding any Parameter elements).
   b) Update its Status element:
      i) The value of each element in the IEEE1451_Dot4TransducerBlock:Status:StatusFlags sequence (excluding the BUSY element) shall be the OR of all the corresponding IEEE1451_Dot4TransducerInterface:Status elements in all associated IEEE1451_Dot4TransducerInterface objects. For example, the IEEE1451_Dot4TransducerBlock:Status:StatusFlags:Service_Request element shall be the OR value of each IEEE1451_Dot4TransducerInterface:Status:Service_Request elements in all associated IEEE1451_Dot4TransducerInterface objects.
      ii) The MMIID of the IEEE1451_Dot4TransducerInterface object where the EVENT originated shall be added to the IEEE1451_Dot4TransducerBlock:Status:MMIIDList sequence. If the IEEE1451_Dot4TransducerBlock:Status:MMIIDList sequence contains an element with the same value as the MMIID that originated the event, then no MMIID element shall be added.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

c) If an IEEE1451_Dot4XcdrChannel associated with an IEEE1451_Dot4PublicXdcr has an Active element with a value of false, then

i) The IEEE1451_Dot4PublicXdcr Active element shall be set to false (i.e., it shall be inactive)

ii) The IEEE1451_Dot4PublicXdcr IEEE1451_Dot4PublicXdcrID shall be kept on the IEEE1451_Dot4TransducerBlock AssociatedPublicXdcrs sequence until the IEEE1451_Dot4TransducerBlock is reset.

9.3.4.2 RESET_EVENT

RESET_EVENT shall be raised by TransducerInterface as the result of a local or IEEE1451_Dot4TransducerBlock interface operation. The EventSource associated to the IEEE1451_Dot4TransducerInterface shall generate the RESET_EVENT. Upon reception of a RESET_EVENT by an IEEE1451_Dot4TransducerBlock, it shall update the content of its associated IEEE1451_Dot4PublicXdcr objects and its IEEE1451_Dot4TransducerBlock:Status element to reflect the changes.

The steps to generate and detect a RESET_EVENT are as follows:

1) The IEEE1451_Dot4TransducerInterface object locally initiates a reset operation on its associated physical MMI, and updates its associated IEEE1451_Dot4MMXdcrs and IEEE1451_Dot4MMXdcrChannel objects by subclass specific means.
2) The IEEE1451_Dot4TransducerInterface object
   a) Sets its IEEEE1451_Dot4TransducerInterface:Status:RESET_EVENT element to true.
   b) Sets its IEEE1451_Dot4TransducerInterface:Status:Event_Missed element to true if the IEEE1451_Dot4TransducerInterface:Status:Service_Request element has a value of true. Otherwise, it sets its Status:Event_Missed to false.
   c) Updates its IEEE1451_Dot4TransdcuerInterface:Status:Service_Request element performing the logical operation defined in Figure 36.
3) The IEEE1451_Dot4TransducerInterface object starts discovery on the physical MMI, and updates its associated IEEE1451_Dot4MMXdcrs and IEEE1451_Dot4MMXdcrChannel objects by subclass specific means.
   a) If an IEEE1451_Dot4MMXdcr is no longer associated to a physical 1451.4 Transducer (i.e., the physical transducer has been removed), then the IEEE1451_Dot4MMXdcr and IEEE1451_Dot4XcdrChannel objects data models shall be removed from the IEEE1451_Dot4TransducerInterface AssociatedXdcrChannels and AssociatedMMXdcrs element sequences.
   b) If a new IEEE 1451.4 Transducer has been added, then
      i) The IEEE1451_Dot4XdcrChannel BasicTEDSID element representing the newly added IEEE1451.4 Transducer shall be added to the IEEE1451_TransducerInterface AssociatedMMXdcr element sequence.
      ii) The IEEE1451_Dot4XdcrChannel objects XdcrChannelID element representing the newly added IEEE1451_Dot4MMXdcr shall be added to IEEE1451_TransducerInterface: AssociatedXdcrChannels element sequence.
4) The IEEE1451_Dot4TransducerInterface shall broadcast a RESET_EVENT addressed to all IEEE1451_Dot4EventSink objects in the TBOM.
5) The IEEE1451_Dot4TransducerBlock upon reception of a RESET_EVENT shall
   a) Update its associated IEEE1451_Dot4PublicXdcr by subclass specific means (excluding any Parameter elements).
   b) Update its Status element:

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

i) The value of each element in the IEEE1451_Dot4TransducerBlock:Status:StatusFlags sequence (excluding the BUSY element) shall be the OR of all the corresponding IEEE1451_Dot4TransducerInterface:Status elements in all associated IEEE1451_Dot4TransducerInterface objects.

ii) The MMIID of the IEEE1451_Dot4TransducerInterface object where the EVENT originated shall be added to the IEEE1451_Dot4TransducerBlock:Status:MMIIDList sequence. If the IEEE1451_Dot4TransducerBlock:Status:MMIIDList sequence contains an element with the same value as the MMIID that originated the event, then no MMIID element shall be added.

6) If an IEEE1451_Dot4XcdrChannel associated with an IEEE1451_Dot4PublicXdcr has been removed

a) The IEEE1451_Dot4PublicXdcr IEEE1451_Dot4PublicXdcrID shall be kept on the IEEE1451_Dot4TransducerBlock AssociatedPublicXdcrs sequence until the IEEE1451_Dot4TransducerBlock is reset.

9.3.4.3 REFRESH_EVENT

REFRESH_EVENT shall be raised by TransducerInterface as the result of a locally initiated refresh operation (for example, the refresh button was pressed on an instrument). The EventSource associated to the `IEEE1451_Dot4TransducerInterface` shall generate the REFRESH_EVENT. Upon reception of a REFRESH_EVENT by an `IEEE1451_Dot4TransducerBlock,` it shall update the content of its associated `IEEE1451_Dot4PublicXdcr objects` and its IEEE1451_Dot4TransducerBlock:Status element to reflect the changes.

The steps to generate and detect a REFRESH_EVENT are as follows:

1) The `IEEE1451_Dot4TransducerInterface` object locally initiates a refresh operation on its associated physical MMI, and updates its associated `IEEE1451_Dot4MMXdcrs` and `IEEE1451_Dot4MMXdcrChannel` objects by subclass specific means.

2) The IEEE1451_Dot4TransducerInterface object

a) Sets its IEEEE1451_Dot4TransducerInterface:Status:REFRESH_EVENT element to true.

b) Sets its IEEE1451_Dot4TransducerInterface:Status:Event_Missed element to true if the IEEE1451_Dot4TransducerInterface:Status:Service_Request element has a value of true. Otherwise, it set its Status:Event_Missed to false.

c) Updates its IEEE1451_Dot4TransdcuerInterface:Status:Service_Request element performing the logical operation defined in Figure 36.

3) The IEEE1451_Dot4TransducerInterface object starts discovery on the physical MMI, and updates its associated IEEE1451_Dot4MMXdcrs and IEEE1451_Dot4MMXdcrChannel objects by subclass specific means.

a) If an IEEE1451_Dot4MMXdcr is no longer associated to a physical 1451.4 Transducer (i.e., the physical transducer has been removed), then the IEEE1451_Dot4MMXdcr and IEEE1451_Dot4XcdrChannel objects data models Active element associated with the IEEE 1451.4 Transducer shall be set to false and shall not be removed from the system. The IEEE1451_Dot4MMXdcr and IEEE1451_Dot4XcdrChannel shall only be removed from the TBOM when the IEEE1451_Dot4TransducerBlock is reset.

b) If a new IEEE 1451.4 Transducer has been added, then

i) The IEEE1451_Dot4XdcrChannel BasicTEDSID element representing the newly added IEEE1451.4 Transducer shall be added to the IEEE1451_TransducerInterface AssociatedMMXdcr element sequence.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

ii) The IEEE1451_Dot4XdcrChannel objects XdcrChannelID element representing the newly added IEEE1451_Dot4MMXdcr shall be added to IEEE1451_TransducerInterface: AssociatedXdcrChannels element sequence.

4) The `IEEE1451_Dot4TransducerInterface` shall broadcast a REFRESH_EVENT addressed to all IEEE1451_Dot4EventSink objects in the TBOM.

5) The IEEE1451_Dot4TransducerBlock upon reception of a REFRESH_EVENT shall

a) Update its associated IEEE1451_Dot4PublicXdcr by subclass specific means (excluding any Parameter elements).

b) Update its Status element:

i) The value of each element in the IEEE1451_Dot4TransducerBlock:Status:StatusFlags sequence (excluding the BUSY element) shall be the OR of all the corresponding IEEE1451_Dot4TransducerInterface:Status elements in all associated IEEE1451_Dot4TransducerInterface objects.

ii) The MMIID of the IEEE1451_Dot4TransducerInterface object where the EVENT originated shall be added to the IEEE1451_Dot4TransducerBlock:Status:MMIIDList sequence. If the IEEE1451_Dot4TransducerBlock:Status:MMIIDList sequence contains an element with the same value as the MMIID that originated the event, then no MMIID element shall be added.

6) If an IEEE1451_Dot4XcdrChannel associated with an IEEE1451_Dot4PublicXdcr has an Active element with a value of false, then

a) The IEEE1451_Dot4PublicXdcr Active element shall be set to false (i.e., it shall be inactive)

b) The IEEE1451_Dot4PublicXdcr IEEE1451_Dot4PublicXdcrID shall be kept on the IEEE1451_Dot4TransducerBlock AssociatedPublicXdcrs sequence until the IEEE1451_Dot4TransducerBlock is reset.

9.3.4.4 ERROR_EVENT

EVENT_ERROR shall be generated when any event generation capable TBOM component detects a local error by subclass specific means (i.e., `IEEE1451_Dot4TransducerInterface`). For example, the `IEEE1451_Dot4TransducerInterface` associated with an MMI detects that the MMI is experiencing a short circuit. ERROR_EVENT events shall be process by subclass specific means.

The steps to detect and generate and ERROR_EVENT are as follows:

1) The TBOM component detects a local error by subclass specific means.

2) The `IEEE1451_Dot4EventSource` broadcasts the ERROR_EVENT to all `IEEE1451_Dot4EventSinks` in the TBOM.

9.3.5 IEEE1451_Dot4EventType semantics

The IEEE1451_Dot4EventType shall represent the data model for events sent by the IEEE1451_Dot4EventSource and received by the IEEE1451_Dot4CommandSink. The event semantics are defined in Annex L. The IEEE1451_ Dot4EventType shall contain two root elements, the Header and Body elements.

9.3.5.1 Header element

The Header element contains a Source element with the identifier of the object owning the `IEEE1451_Dot4EventSource` (i.e., the `IEEE1451_Dot4TransducerInterface` MMIID identifier).

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.3.5.2 Body element

The Body attribute shall contain three elements as described by Annex L. EventName element shall describe the type of event as defined in Annex L. EventTypeID and EventDescriptionElement shall correspond to the description and EventTypeID defined in Table 29 and Annex L.

Table 29—EventTypeID description

Event Name	EventTypeID	Description
NULL_EVENT	0	No event has been detected.
HOTSWAP_EVENT	1	A HotSwap event has been detected.
REFRESH_EVENT	2	A Refresh event has been detected.
RESET_EVENT	3	A Reset event has been detected.
ERROR_EVENT	4	MMI short circuit (the MMI electrical level is at ground when it should be at a defined voltage).
	5	MMI protocol time-out (a command was issued on the MMI and could not be completed).
	6	MMI has detected more than one 1451.4 Transducer with the same BASICTEDSID.
	7	MMI can not find all nodes in a Node-List of a 1451.4 Transducer.
	8	MMI has detected a CRC error in 1451.4 Transducers.
	9	MMI unable to initialize bus.
	10–255	Reserved.
	256–4095	Open to industry.
	4096–16383	User defined.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

9.3.6 IEEE1451_Dot4EventSource and IEEE1451_Dot4EventSink local operations

9.3.6.1 IEEE1451_Dot4EventSource local operations

See Table 30.

Table 30—EventSource local operation

Operation	Requirement
GenerateEvent	(m)

9.3.6.1.1 Operation GenerateEvent

Description: GenerateEvent shall generate an event of valid EventName type (HOTSWAP_EVENT, REFRESH_EVENT, ERROR_EVENT) with EventTypeID defined in Table 29.

© IEEE 2004 – All rights reserved

```
OpReturnCode GenerateEvent{
[in] xs:hexBinary MMIID,
[in] xs:string EventName
[in] xs:unsignedInt EventTypeID,
}
```

9.3.6.1.1.1 Argument MMIID

Argument MMIID shall uniquely identify the TransducerInterface generating the event.

9.3.6.1.1.2 Argument EventName

Argument EventName shall be a valid operation EventName defined in Annex L.

9.3.6.1.1.3 Argument EventTypeID

Argument EventTypeID shall contain a valid EventTypeID defined in Table 29 and Annex L.

9.3.6.2 IEEE1451_Dot4EventSink local operations

No local operation is defined.

9.4 TEDS Service

The TEDS Service implements complete transactional services supporting reading and writing of information located in the TEDS. The TEDS Service encapsulates the responsibility for reading and writing all TEDS, whether the TEDS is embedded in the IEEE 1451.4 Transducer or implemented as a virtual resource located within the end user's system.

There are no requirements in this standard to support a write transaction to an Appended TEDS. The TEDS service relies on deployment-specific configuration data to locate and retrieve the Appended TEDS from the end user's system. Table 31 provides the interface requirements to implement a TEDS service (m = mandatory, o = optional).

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table 31—TEDS service operations

Operation	Requirement
ParseTEDS	(m)
EncodeTEDS	(o)
GetTemplate	(o)

9.4.1 Operation ParseTEDS specification

Description: Operation ParseTEDS provides a mechanism to decode the binary encoded TEDS (BTEDS) in IEEE 1451.4 Transducers as defined in 6.5.

```
OpReturnCode ParseTEDS  {
      [in] xs:anyURI ServiceLocation,
      [in] IEEE1451_Dot4:IEEE1451_Dot4BTEDSType BTEDS,
      [out] IEEE1451_Dot4:IEEE1451_Dot4MMXdcrType MMXdcr
}
```


© IEEE 2004 – All rights reserved

9.4.1.1 Argument ServiceLocation

Argument ServiceLocation shall be the URI location of the ParseTEDS service.

9.4.1.2 Argument BTEDS

Argument BTEDS shall be a data structure containing the IEEE1451_Dot4BTEDS to be decoded. IEEE1451_Dot4BTEDS are defined in 9.2.7 and Annex L. In case the BTEDS represents an IEEE 1451.4 Transducer with Appended TEDS, the ServiceLocation shall be able to return an appropriately decoded MMXdcrType structure.

9.4.1.3 Argument MMXdcr

Argument MMXdcr shall be the data structure containing the decoded MMXdcr.

9.4.2 Operation EncodeTEDS specification

Description: Operation EncodeTEDS provides a mechanism to generate binary encoded TEDS (BTEDS) from an MMXdcr object. The resulting BTEDS can be encoded into IEEE 1451.4 Transducers as defined in 6.5.

```
OpReturnCode EncodeTEDS {
      [in] xs:anyURI ServiceLocation,
      [out] IEEE1451_Dot4:IEEE1451_Dot4BTEDSType BTEDS,
      [in] IEEE1451_Dot4:IEEE1451_Dot4MMXdcrType MMXdcr
}
```

9.4.2.1 Argument ServiceLocation

Argument ServiceLocation shall be the URI location of the EncodeTEDS service.

9.4.2.2 Argument BTEDS

Argument BTEDS shall be a data structure containing the encoded BTEDS.

9.4.2.3 Argument MMXdcr

Argument MMXdcr shall be the data structure containing the MMXdcr object to be encoded.

9.4.3 Operation GetTemplate specification

Description: Operation GetTemplate provides a mechanism to retrieve templates from a specified location defined by a user-defined URI.

```
OpReturnCode GetTemplate      {
      [in] xs:anyURI ServiceLocation,
      [in] xs:unsignedShort ManufacturerID,
      [in] xs:unsignedShort TemplateID,
      [out] IEEE1451_Dot4:IEEE1451_Dot4TemplateType Template
}
```

9.4.3.1 Argument ServiceLocation

Argument ServiceLocation shall be the URI location of the GetTemplate service.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.4.3.2 Argument ManufacturerID

Argument ManufacturerID shall be a valid Manufacturer ID as defined by 5.1.1.

9.4.3.3 Argument TemplateID

Argument TemplateID shall be a valid template identifier as defined by 6.5.1.

9.4.3.4 Argument Template

Argument Template shall be a valid IEEE1451_Dot4Template data structure as defined by Annex L.

9.5 IEEE 1451.4 Transducer Block general interface

The Transducer Block general interface allows access to the functionality provided by the IEEE1451.4 TBOM. A host or NCAP using the general interface shall be exposed only to the IEEE1451_Dot4TransducerBlock general interface. The IEEE1451_Dot4TransducerBlock general interface encapsulates the functionality of all components defined by the IEEE1451.4 TBOM. Interface definitions for each component defined in the IEEE1451.4 TBOM are provided for manufacturers of Transducer Blocks and system components (e.g., class libraries, instruments, data acquisition systems). Operations are defined as mandatory (m) and optional (o). Mandatory operations shall implement the full functionality specified. Optional operations shall be full or interface only implementation of the interface specification. Operations shall return an OpReturnCode defined upon completion as shown in IEEE1451.1-1999, Table 9. Interface only implementations shall not cause any change of state of the object and shall return immediately with major return code of: MJ_NOP_OPERATION (see 7.2.1.2, IEEE Std 1451.1-1999). The general interface data types are defined in Annex L.

9.5.1 IEEE1451_Dot4TransducerBlock

Operations defined in Table 27 provide methods to manage and identify Transducer Block objects. See Table 32.

Table 32—IEEE1451.4 Transducer Block managing operations

Operation Name	Requirement
GetTBlockName	(m)
SetTBlockName	(o)
GetTBlockManufacturerID	(m)
GetTBlockModelNumber	(m)
GetTBlockVersion	(m)
GetTBlockStatus	(m)
GetTBlockTBOM	(m)
TBlockReset	(m)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.1 Operation GetTBlockName specification

Description: Operation GetTBlockName provides a mechanism to read a system or user TransducerBlock label. The labels format and semantics are outside the scope of this standard.

```
OpReturnCode GetTBlockName    {
      [out] xs:string TBlockName
}
```

9.5.1.1.1 Argument TBlockName

Argument TBlockName value shall be a system or user-defined label.

9.5.1.2 Operation SetTBlockName specification

Description: Operation SetTBlockName provides a mechanism to assign a system or user-defined `TransducerBlock` name label. The name label format and semantics are outside the scope of this standard.

```
OpReturnCode SetTBlockName {
      [out] xs:string TBlockName
}
```

9.5.1.2.1 Argument TBlockName

Argument TBlockName value shall be a system or user-defined label describing the function preformed by the `TransducerBlock` instance.

9.5.1.3 Operation GetTBlockManufacturerID

Description: Operation GetTBlockManufacturerID provides a mechanism to read the `TransducerBlock` manufacturer code interpretation in a string format as defined in Appendix K.

```
OpReturnCode GetTBlockManufacturerID {
[out] xs:string ManufacturerID
}
```

9.5.1.3.1 Argument ManufacturerID

Argument BlockName shall be TransducerBlock manufacturer name.

9.5.1.4 Operation GetTBlockModelNumber

Description: Operation GetTBlockModelNumber provides a mechanism to read the `TransducerBlock`'s manufacturer-defined model number. The model number label format and semantics are outside the scope of this standard.

```
OpReturnCode GetTBlockModelNumber {
[out] xs:string ManufacturerModelNumber
}
```

9.5.1.4.1 Argument ManufacturerModelNumber

Argument ManufacturerModelNumber shall be assigned by the Transducer Block manufacturer to identify the `TransducerBlock` model.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.5 Operation GetTBlockVersion

Description: Operation GetTBlockVersion provides a mechanism to read the Transducer Block's manufacturer-defined version. The version label format and semantics are outside the scope of this standard.

```
OpReturnCode GetTBlockVersion {
[out] xs:string TBlockVersion
}
```

9.5.1.5.1 Argument TBlockVersion

Argument TBlockVersion shall be assigned by the TrasnducerBlock manufacturer to identify the `TransducerBlock` version.

9.5.1.6 Operation GetTBlockStatus

Description: Operation GetBlockStatus provides a mechanism to obtain a read-only data structure containing the Transducer Block's status data model as defined in Annex L. The IEEE1451_Dot4BlockStatusType contains information about the `TransducerBlock's` associated `EventSink` and `CommandSource` objects.

```
OpReturnCode GetTBlockStatus  {
[out] IEEE1451_Dot4:IEEE1451_Dot4TBlockStatusType TBlockStatus
}
```

9.5.1.6.1 Argument TBlockStatus

Argument TBlockStatus shall return the IEEE1451_Dot4TransducerBlock:Status element as defined in Annex L.

9.5.1.7 Operation GetTBlockTBOM

Description: Operation GetTBlockTBOMshall provide a mechanism to obtain a read-only data structure containing the TBOM data model as defined in Annex L.

```
OpReturnCode GetTBlockTBOM    {
[out] IEEE1451_Dot4:IEEE1451_Dot4TransducerBlockType TBOM
}
```

9.5.1.7.1 Argument TBOM

Argument TBOM shall return the Transducer Block structure representing the IEEE1451_Dot4 TBOM as defined in Annex L.

9.5.1.8 Operation TBlockReset

Description: Operation TBlockReset provides a mechanism to initialize the `TransducerBlock` to its default state. The TBOM data structure associated with the `TransducerBlock` is cleared, and a reset process takes place in all associated `TransducerInterface` objects. All associated `CommandSource`, `CommandSink`, and `EventSource`, `EventSink` are cleared to their default states and any pending operations are discarded. All PublicXdcrs, XdcrChannels, and MMXdcrs with data model Active element with a value of false shall be purged from the TBOM. See Table 33.

```
OpReturnCode TBlockReset {
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table 33—IEEE1451.4 Transducer Block TransducerInterface managing operations

Operation Name	Requirement
GetNumberOfTransducerInterfaces	(m)
GetAssociatedTransducerInterfaces	(m)
GetTransducerInterfaceStatus	(m)
SetTransducerInterfaceStatus	(o)
GetTransducerInterfaceEventMask	(m)
SetTransducerInterfaceEventMask	(m)
GetTransducerInterfaceMode	(m)
SetTransducerInterfaceMode	(o)
LockTransducerInterfaceMode	(m)
UnLockTransducerInterfaceMode	(m)
GetTransducerInterfaceClass	(o)
ResetTransducerInterface	(m)
RefreshTransducerInterface	(m)
SelfTestTransducerInterface	(o)
GetNumberOfMMXdcrsInTransducerInterface	(o)
GetMMXdcrListInTransducerInterface	(o)
GetNumberOfNodesInTransducerInterface	(o)
GetNodeListInTransducerInterface	(o)
TunnelTransparentProtocol	(o)
TunnelProtocol	(o)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

9.5.1.9 Operation GetNumberOfTransducerInterfaces

Description: Operation GetNumberOfTransducerInterfaces provides a mechanism to count the total number of `TransducerInterface` objects (representing physical MMIs) associated with the `TransducerBlock` object (i.e., the total number of ID elements in the IEEE1451_Dot4TransducerBlock:AssociatedTransducerInterfaces sequence). All `TransducerInterface` objects associated with the `TransducerBlock` object shall be counted even when no IEEE 1451.4 Transducers (represented as `MMXdcr` objects) are connected to the physical MMI.

```
OpReturnCode GetNumberOfTransducerInterfaces{
[out] xs:unsignedShort NumberOfTransducerInterfaces
}
```

9.5.1.9.1 Argument NumberOfTransducerInterfaces

Argument NumberofTransducerInterfaces shall enumerate the number `TransducerInterface object` associated with the `TransducerBlock`.

© IEEE 2004 – All rights reserved

9.5.1.10 Operation GetAssociatedTransducerInterfaces

Description: Operation GetAssociatedTransducerInterfaces provides a mechanism to obtain a read-only data structure containing the IEEE1451_Dot4TransducerBlock:AssociatedTransducerInterfaces element value.

```
OpReturnCode GetAssociatedTransducerInterfaces {
[out]                    IEEE1451_Dot4:IEEE1451_Dot4AssociatedObjectsType
TransducerInterfaceList
}
```

9.5.1.10.1 Argument TransducerInterfaceList

Argument TransducerInterfaceList shall be a read-only data structure containing the value of the IEEE1451_Dot4TransducerBlock:AssociatedTransducerInterfaces element.

9.5.1.11 Operation GetTransducerInterfaceStatus

Description: Operation GetTransducerInterfaceStatus provides a mechanism to read the IEEE1451_Dot4`TransducerInterface:Status` element as defined in Annex L.

```
OpReturnCode GetTransducerInterfaceStatus{
[in] xs:hexBinary MMIID,
[out] IEEE1451_Dot4:IEEE1451_Dot4StatusType TransducerInterfaceStatus
}
```

9.5.1.11.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

9.5.1.11.2 Argument TransducerInterfaceStatus

Argument TransducerInterfaceStatus shall be a data structure representing the IEEE1451_Dot4`TransducerInterface:Status` element.

9.5.1.12 Operation SetTransducerInterfaceStatus

Description: Operation SetTransducerInterfaceStatus provides a mechanism to set the IEEE1451_Dot4TransducerInterface:Status element as defined in Annex L.

NOTE—Operation SetTransducerInterfaceStatus is intended for system designers and programmers of IEEE 1451.4 instruments building support in their devices for IEEE 1451.4 MMIs. Operation SetTransducerInterfaceStatus facilitates wrapping a TransducerInterface object around existing instrument software drivers.

```
OpReturnCode SetTransducerInterfaceStatus {
[in] xs:hexBinary MMIID,
[in] IEEE1451_Dot4:IEEE1451_ Dot4StatusType TransducerInterfaceStatus
}
```

9.5.1.12.1 Argument MMIID

Argument MMIID shall uniquely identify the TransducerInterface object being addressed by the operation.

9.5.1.12.2 Argument TransducerInterfaceStatus

Argument TransducerInterfaceStatus shall set the value of the TransducerInterfaceStatus attribute of the TransducerInterface.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.13 Operation GetTransducerInterfaceEventMask

Description: Operation GetTransducerInterfaceEventMask provides a mechanism to read the current value of the IEEE1451_Dot4TransducerInterface:EventMask element as defined in Annex L.

```
OpReturnCode GetTransducerInterfaceEventMask{
[in] xs:hexBinary MMIID
[out] xs:IEEE1451_Dot4EventMaskType EventMask
}
```

9.5.1.13.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

9.5.1.13.2 Argument EventMask

Argument EventMask shall contain the value of the IEEE1451_Dot4TransducerInterface:EventMask element.

9.5.1.14 Operation SetTransducerInterfaceEventMask

Description: Operation SetTransducerInterfaceEventMask provides a mechanism to configure the Transducer Interface to generate events as defined by the EventMask argument (i.e., HOTSWAP_EVENT, REFRESH_EVENT, RESET_EVENT, and ERROR_EVENT as defined in Annex L). Events not supported by a particular Transducer Interface shall not be enabled by operation SetTransducerInterfaceEventMask.

```
OpReturnCode SetTransducerInterfaceEventMask{
[in] xs:hexBinary MMIID
[in] xs:IEEE1451_Dot4EventMaskType EventMask
}
```

9.5.1.14.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

9.5.1.14.2 Argument EventMask

Argument EventMask shall set the value of the IEEE1451_Dot4TransducerInterface:EventMask element.

9.5.1.15 Operation GetTransducerInterfaceMode

Description: Operation GetTransducerInterfaceMode provides a mechanism to read the IEEE1451_Dot4TransducerInterface:Mode element (i.e., Analog, Digital, Analog+Digital) representing the current operational mode of the physical MMI.

```
OpReturnCode GetTransducerInterfaceMode {
[in] xs:hexBinary MMIID
[out]                                IEEE1451_Dot4:IEEE1451_Dot4MMIMode
TransducerInterfaceModeOfOperation
}
```

9.5.1.15.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.15.2 Argument TransducerInterfaceModeOfOperation

Argument TransducerInterfaceModeOfOperation shall contain the value of the MMIModeOfOperation element of the IEEE1451_Dot4`TransducerInterface` with MMIID element value. Allowed values are defined in Annex L.

9.5.1.16 Operation SetTransducerInterfaceMode

Description: Operation SetTransducerInterfaceMode provides a mechanism to set the value of the IEEE1451_Dot4TransducerInterface:Mode element in the `TransducerInterface object` with MMIID element.

NOTE—Operation SetTransducerInterfaceStatus is intended for system designers and programmers of IEEE 1451.4 instruments building support in their devices for IEEE 1451.4 MMIs. Operation SetTransducerInterfaceMode facilitates wrapping a Transducer Interface object around existing instrument software drivers.

```
OpReturnCode SetTransducerInterfaceMode {
[in] xs:hexBinary MMIID,
[in]IEEE1451_Dot4:IEEE1451_Dot4MMIModeTransducerInterfaceModeOfOperation
}
```

9.5.1.16.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

9.5.1.16.2 Argument TransducerInterfaceModeOfOperation

Argument TransducerInterfaceModeOfOperation shall contain the value that the IEEE1451_Dot4TransducerInterface:Mode element in the `TransducerInterface object` with MMIID element. Allowed values are defined in Annex L.

9.5.1.17 Operation LockTransducerInterfaceMode

Description: Operation LockTransducerInterfaceMode provides a mechanism to set and lock the IEEE1451_Dot4TransducerInterface:Mode element (i.e., Analog, Digital, Analog+Digital) of the IEEE1451_Dot4TransducerInterface with MMIID identifier. The IEEE1451_Dot4TransducerInterface Locked element shall be set to a value of true. Operation LockTransducerInterfaceMode shall prevent any change in the `TransducerInterface` mode of operation including those associated with events of class HOTSWAP_EVENT, REFRESH_EVENT, RESET_EVENT, and ERROR_EVENT.

```
OpReturnCode LockTransducerInterfaceMode {
[in] xs:hexBinary MMIID
[out]IEEE1451_Dot4:IEEE1451_Dot4MMIModeType
TransducerInterfaceModeOfOperation
}
```

9.5.1.17.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

9.5.1.17.2 Argument TransducerInterfaceModeOfOperation

Argument TransducerInterfaceModeOfOperation shall contain the value of the IEEE1451_Dot4TransducerInterface:Mode element enumeration that the `TransducerInterface` shall be set and locked into. Allowed values are defined in Table L.7.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.18 Operation UnLockTransducerInterfaceMode

Description: Operation UnLockTransducerInterfaceMode provides a mechanism to unlock the current mode of operation of the MMI associated with the `TransducerInterface` with MMIID element. The value of the IEEE1451_Dot4TransducerInterface:Locked element shall be set to false.

```
OpReturnCode UnLockTransducerInterfaceMode {
[in] xs:hexBinary MMIID
}
```

9.5.1.18.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

9.5.1.19 Operation GetTransducerInterfaceClass

Description: Operation GetTransducerInterfaceClass provides a mechanism to read the IEEE1451_Dot4TransducerInterface:MMIClass element (i.e., Class1,Class2) representing the type of MMI being used.

```
OpReturnCode GetTransducerInterfaceMode {
[in] xs:hexBinary MMIID
[out] IEEE1451_Dot4:IEEE1451_Dot4MMIClassType TransducerInterfaceClass
}
```

9.5.1.19.1 Argument MMIID

Argument MMIID shall uniquely identify the TransducerInterface object being addressed by the operation.

9.5.1.19.2 Argument TransducerInterfaceClass

Argument TransducerInterfaceClass shall contain the value of the IEEE1451_Dot4TransducerInterface:MMIClass element. Allowed values are defined in Annex L.

9.5.1.20 Operation ResetTransducerInterface

Description: Operation ResetTransducerInterface provides a mechanism to reset the `TransducerInterface`. Operation ResetTransducerInterface shall generate a RESET_EVENT in the `EventSource` associated with `TransducerInterface` with MMIID identifier. All attributes associated with the `TransducerInterface` shall be set to their default states. All events shall be cleared and the `CommandSink:State` element shall be set to its WaitForCommand state. All XdcrChannels and MMXdcrs with Active element value of false shall be removed. Any pending operations shall be discarded.

```
OpReturnCode ResetTransducerInterface{
[in] xs:hexBinary MMIID
}
```

9.5.1.20.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.21 Operation RefreshTransducerInterface

Description: Operation ResetTransducerInterface provides a mechanism to discover new IEEE 1451.4 Transducers attached to the MMI represented by the `TransducerInterface` object. Operation ResetTransducerInterface shall generate a REFRESH_EVENT to the `EventSink` associated with the `TransducerInterface` with MMIID identifier.

```
OpReturnCode RefreshTransducerInterface {
[in] xs:hexBinary MMIID
}
```

9.5.1.21.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

9.5.1.22 Operation SelfTestTransducerInterface

Description: Operation SelfTestTransducerInterface provides a mechanism to verify the correct operation of the MMI associated with `TransducerInterface`. The specific testing mechanisms are outside the scope of this standard.

```
OpReturnCode SelfTestTransducerInterface{
[in] xs:unsignedLong MMIID
[out] xs:unsignedByte SelfTestResult
}
```

9.5.1.22.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation

9.5.1.22.2 Argument SelfTestResult

Argument SelfTestResult shall have values restricted by Table 34. A FAIL value indicates that the `TransducerInterface` is not operating correctly. A PASS value shall indicate that the `TransducerInterface` is operating correctly.

Table 34—SelfTestResult enumerated values

Value	Description
0	FAIL
1	PASS
2–127	Reserved
128–255	User defined

9.5.1.23 Operation GetNumberOfMMXdcrsInTransducerInterface

Description: Operation GetNumberOfMMXdcrsInTransducerInterface provides a mechanism to count the number of IEEE1451_Dot4`MMXdcr` objects associated to the `TransducerInterface` object with MMIID identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
OpReturnCode GetNumberOfMMXdcrsInTransducerInterface{
[in] xs:hexBinary MMIID,
[out] xs:unsignedByte NumberOfMMXdcrs
}
```

9.5.1.23.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

9.5.1.23.2 Argument NumberOfMMXdcrs

Argument NumberOfMMXdcr shall have a value equal to the number of elements in the IEEE1451_Dot4TransducerInterface:MMXdcrList sequence of the `TransducerInterface` object with MMIID identifier.

9.5.1.24 Operation GetMMXdcrListInTransducerInterface

Description: Operation GetMMXdcrListInTransducerInterface provides a mechanism to obtain a read-only data structure containing the value of MMXdcrList attribute associated with the `TransducerInterface` object with MMIID identifier.

```
OpReturnCode GetMMXdcrListInTransducerInterface{
[in] xs:hexBinary MMIID,
[out] IEEE1451_Dot4:IEEE1451_Dot4MMXdcrListType MMXdcrList
}
```

9.5.1.24.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation

9.5.1.24.2 Argument MMXdcrList

Argument MMXdcrList shall contain a read-only data structure containing the value of MMXdcrList element of the IEEE1451_Dot4TransducerInterface object with MMIID identifier.

9.5.1.25 Operation GetNumberOfNodesInTransducerInterface

Description: Operation GetNumberOfNodesInTransducerInterface provides a mechanism to count the number of `Node` objects associated to the `TransducerInterface` object with MMIID identifier.

```
OpReturnCode GetNumberOfNodesInTransducerInterface{
[in] xs:hexBinary MMIID,
[out] xs:unsignedByte NumberofMMINodes
}
```

9.5.1.25.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation

9.5.1.25.2 Argument NumberofTransducerInterfaceNodes

Argument NumberofTransducerInterfaceNodes shall contain the number of `Node` objects associated to the `TransducerInterface` object with MMIID identifier. The number of Node objects shall be equal to

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

the cumulative sum of all Node elements in each IEEE1451_Dot4MMXdcr:NodeList sequence defined in the IEEE1451_Dot4TransducerInterface:MMXdcrList.

9.5.1.26 Operation GetNodeListInTransducerInterface

Description: Operation GetNodeListInTransducerInterface provides a mechanism to obtain a read-only data structure containing the concatenated value of all `MMXdcr` object's NodeList elements associated to the `TransducerInterface` object with MMIID identifier.

```
OpReturnCode GetNodeListInTransducerInterface{
[in] xs:hexBinary MMIID,
[out] IEEE1451_Dot4:IEEE1451_Dot4NodeListType MMINodeList
}
```

9.5.1.26.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation

9.5.1.26.2 Argument TransducerInterfaceNodeList

Argument TransducerInterfaceNodeList shall contain a list of Node objects associated with the `TransducerInterface` object with MMIID identifier. TransducerInterfaceNodeList shall be a concatenated sequence of all `MMXdcr` object's NodeList elements associated to the `TransducerInterface` object with MMIID identifier.

9.5.1.27 Operation TunnelTransparentProtocol

Description: Operation TunnelTransparentProtocol provides a mechanism to pass data directly to the MMI associated with the `TransducerInterface` object using the transparent protocol as the messaging protocol. When operation TunnelTransparentProtocol passes data directly to the MMI, the `TransducerInterface` object associated `CommandSink:State` element shall be set to ExecutingCommand as defined in Annex L. Upon completion of the operation TunnelTransparentProtocol, the `TransducerInterface` object's associate `CommandSink:State` element shall have a value of WaitForCommand or ErrorHandler. In case an error occurs (ErrorHandler), the TransducerInteface object's associate `EventSource` shall generate an Error_Event.

```
OpReturnCode TunnelTransparentProtocol {
[in] xs:hexBinary MMIID,
[in] xs:string BufferIn,
[out] xs:string BufferOut,
[out] xs:string CommandResult
}
```

9.5.1.27.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation.

9.5.1.27.2 Argument BufferIn

Argument BufferIn shall contain a string with valid transparent protocol commands directed to the MMI associated with `TransducerInterface` object with MMIID identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.27.3 Argument BufferOut

Argument BufferOut shall contain a string with the result of the transparent protocol operation(s).

9.5.1.27.4 Argument CommandResult

Argument CommandResult shall contain a string with the result of the transparent protocol operations. SUCCESS shall indicate that the command was completed successfully. FAILURE shall indicate that the command fail to be competed. See Table 35.

Table 35—CommandResult enumerated values

Value	Description
0	FAILURE
1	SUCCESS
2–255	Reserved

9.5.1.28 Operation TunnelProtocol

Description: Operation TunnelTransparentProtocol provides a mechanism to pass data directly to the MMI associated with the `TransducerInterface` object using a manufacturer-defined messaging protocol. When operation TunnelTransparentProtocol passes data directly to the MMI, the `TransducerInterface` object associated `CommandSink:State` element shall be set to ExecutingCommand as defined in Annex L. Upon completion of the operation TunnelTransparentProtocol, the `TransducerInterface` object's associate `CommandSink:State` element shall have a value of WaitForCommand or ErrorHandler. In case an error occurs (ErrorHandler), the `TransducerInterface` object's associate `EventSource` shall generate an Error_Event.

```
OpReturnCode TunnelProtocol {
[in] xs:hexBinary MMIID,
[in] xs:string BufferIn,
[out] xs:string BufferOut,
[out] xs:string CommandResult
}
```

9.5.1.28.1 Argument MMIID

Argument MMIID shall uniquely identify the `TransducerInterface object` being addressed by the operation

9.5.1.28.2 Argument BufferIn

Argument BufferIn shall contain a string with a valid manufacturer-defined messaging protocol command directed to the MMI associated with `TransducerInterface` object with MMIID identifier.

9.5.1.28.3 Argument BufferOut

Argument BufferOut shall contain a string with the result of the command operations. The semantics of BufferOut are outside the scope of this standard.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.28.4 Argument CommandResult

Argument CommandResult shall contain a string with the result of the transparent protocol operations. SUCCESS shall indicate that the command was completed successfully. FAILURE shall indicate that the command fail to be competed. See Table 36 and Table 37.

Table 36—CommandResult enumerated values

Value	Description
0	FAILURE
1	SUCCESS
2–255	Reserved

Table 37—IEEE1451.4 Transducer Block XdcrChannel managing operations

Operation Name	Requirement
GetXdcrChannelName	(o)
GetNumberOfXdcrChannels	(m)
AssociatedXdcrChannelList	(m)
GetNumberOfProperties	(m)
GetPropertyList	(m)
SetPropertyList	(o)
GetXdcrChannelProperty	(m)
SetXdcrChannelProperty	(o)
ExposeProperty	(m)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

9.5.1.29 Operation GetXdcrChannelName specification

Description: Operation GetXdcrChannelName provides a mechanism to read a system or user-defined name label of an `XdcrChannel` object with XdcrChannelID identifier. The name label's format and semantics are outside the scope of this standard.

```
OpReturnCode GetXdcrChannelName {
[in] xs:hexBinary XdcrChannelID,
[out] xs:string XdcrChannelName
}
```

9.5.1.29.1 Argument XdcrChannelID

Argument XdcrChannelID shall uniquely identify the `XdcrChannel` object being addressed by the operation.

© IEEE 2004 – All rights reserved

9.5.1.29.2 Argument XdcrChannelName

Argument XdcrChannelName value shall be equal to the IEEE1451_Dot4XdcrChannel:Name element. The name label's format and semantics are outside the scope of this standard.

9.5.1.30 Operation GetNumberOfXdcrChannels

Description: Operation GetNumberOfXdcrChannels provides a mechanism to count the total number of XdcrChannel objects associated with the IEEE1451_Dot4`TransducerBlock`.

```
OpReturnCode GetNumberOfXdcrChannels {
[out] xs:unsignedShort NumberOfXdcrChannels
}
```

9.5.1.30.1 Argument NumberOfXdcrChannels

Argument NumberOfXdcrChannels shall enumerate the number of `IEEE1451_Dot4XdcrChannel` objects associated with the IEEE1451_Dot4TransducerInterface instance through its associated IEEE1451_Dot4TransducerInterface objects. The value of argument NumberOfXdcrChannels shall be equal to the cumulative sum of ID elements in each associated IEEE1451_Dot4TransducerInterface IEEE1451_Dot4TransducerInterface:AssociatedXdcrChannels sequence.

9.5.1.31 Operation GetAssociatedXdcrChannelList

Description: Operation GetAssociatedXdcrChannelList provides a mechanism to obtain a read-only data structure containing a concatenated sequence of all AssociatedXdcrChannels sequences for each IEEE1451_Dot4TransducerInterface object associated to the IEEE1451_Dot4TransducerBlock.

```
OpReturnCode GetXdcrChannelList {
[out]IEEE1451_Dot4:IEEE1451_Dot4AssociatedObjectsType
AssociatedXdcrChannelList
}
```

9.5.1.31.1 Argument AssociatedXdcrChannelList

Argument XdcrChannelList shall be the concatenated sequence of all IEEE1451_Dot4TransducerInterfaceAssociatedXdcrChannels sequences for each IEEE1451_Dot4TransducerInterface object associated to the IEEE1451_Dot4TransducerBlock.

9.5.1.32 Operation GetNumberOfProperties

Description: Operation GetNumberOfProperties provides a mechanism to count the total number of IEEE1451_Dot4`Property` elements in the XdcrChannelPropertyList sequence of the IEEE1451_Dot4`XdcrChannel` object with XdcrChanelID identifier.

```
OpReturnCode GetNumberOfProperties {
[in] xs:hexBinary XdcrChannelID
[out] xs:unsignedInt NumberOfProperties
}
```

9.5.1.32.1 Argument XdcrChannelID

Argument XdcrChannelID shall uniquely identify the `XdcrChannel` object being addressed by the operation.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.32.2 Argument NumberOfProperties

Argument NumberOfProperties shall enumerate the number of `Property` objects associated with the `XdcrChannel` object with XdcrChanelID identifier.

9.5.1.33 Operation GetPropertyList

Description: Operation GetPropertyList provides a mechanism to obtain a read-only data structure containing the data model of the XdcrChannelPropertyList element sequence associated to the IEEE1451_Dot4`XdcrChannel` with XdcrChannelID identifier.

```
OpReturnCode GetPropertyList {
[in] xs:hexBinary XdcrChannelID
[out] IEEE1451_Dot4:IEEE1451_Dot4PropertyListType PropertyList
}
```

9.5.1.33.1 Argument XdcrChannelID

Argument XdcrChannelID shall uniquely identify the `XdcrChannel` object being addressed by the operation.

9.5.1.33.2 Argument PropertyList

Argument PropertyList shall have a value equal to the IEEE1451_Dot4XdcrChannel:XdcrChannelPropertyList element.

9.5.1.34 Operation SetPropertyList

Description: Operation SetPropertyList provides a mechanism to set XdcrChannelPropertyList element sequence associated to the IEEE1451_Dot4`XdcrChannel` with XdcrChannelID identifier.

```
OpReturnCode SetPropertyList {
[in] xs:hexBinary XdcrChannelID
[in] IEEE1451_Dot4:IEEE1451_Dot4PropertyListType PropertyList
}
```

9.5.1.34.1 Argument XdcrChannelID

Argument XdcrChannelID shall uniquely identify the `XdcrChannel` object being addressed by the operation.

9.5.1.34.2 Argument PropertyList

IEEE1451_Dot4XdcrChannel:XdcrChannelPropertyList element shall be set equal to the value of argument PropertyList.

9.5.1.35 Operation GetXdcrChannelProperty

Description: Operation GetXdcrChannelProperty shall provide a mechanism to obtain a read-only data structure containing the IEEE1451_Dot4`Property` object data model Name element equal to argument PropertyName in the IEEE1451_Dot4`XdcrChannel:XdcrChannelPropertyList` element sequence of the IEEE1451_Dot4XdcrChannel with XdcrChannelID identifier.

```
OpReturnCode GetXdcrChannelProperty {
[in] xs:hexBinary XdcrChannelID
[in] xs:string PropertyName
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
[out] IEEE1451_Dot4:IEEE1451_Dot4PropertyType Property
}
```

9.5.1.35.1 Argument XdcrChannelID

Argument XdcrChannelID shall uniquely identify the `XdcrChannel` object being addressed by the operation.

9.5.1.35.2 Argument PropertyName

Argument PropertyName shall uniquely identify the IEEE1451_Dot4`Property` object being addressed by the operation.

9.5.1.35.3 Argument Property

Argument Property shall have a value equal the IEEE1451_Dot4`Property` object data model of the IEEE1451_Dot4Property object with Name element equal to PropertyName defined in the IEEE1451_Dot4XdcrChannelPropertyList element sequence.

9.5.1.36 Operation SetXdcrChannelProperty

Description: Operation SetXdcrChannelProperty shall provide a mechanism to set the value of a Property object data model associated with attribute Name equal to argument PropertyName in the IEEE1451_Dot4XdcrChannel:XdcrChannelPropertyList with XdcrChannelID identifier. SetXdcrChannelProperty operations shall only be valid for IEEE1451:Property objects with AccessLevel element with value USR or CAL.

```
OpReturnCode SetXdcrChannelProperty {
[in] xs:hexBinary XdcrChannelID
[in] xs:string PropertyName
[in] IEEE1451_Dot4:IEEE1451_Dot4PropertyType Property
}
```

9.5.1.36.1 Argument XdcrChannelID

Argument XdcrChannelID shall uniquely identify the `XdcrChannel` object being addressed by the operation.

9.5.1.36.2 Argument PropertyName

Argument PropertyName shall uniquely identify the `Property` object being addressed by the operation.

9.5.1.36.3 Argument Property

The IEEE1451_Dot4Property element with Name element equal to PropertyName and defined in the IEEE1451_Dot4XdcrChannel:PropertyList sequence shall be set equal to the value of argument Property.

9.5.1.37 Operation ExposeProperty

Description: Operation ExposeProperty shall provide a mechanism to selectively expose IEEE1451_Dot4Property objects defined in the IEEE1451_Dot4XdcrChannel object with XdcrChannelID identifier.

```
OpReturnCode ExposeProperty {
[in] xs:hexBinary XdcrChannelID
[in] xs:string PropertyName
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
[in] xs:boolean Expose
}
```

9.5.1.37.1 Argument XdcrChannelID

Argument XdcrChannelID shall uniquely identify the `XdcrChannel` object being addressed by the operation.

9.5.1.37.2 Argument PropertyName

Argument PropertyName shall uniquely identify the `Property` object being addressed by the operation.

9.5.1.37.3 Argument Expose

The IEEE1451_Dot4Property:Exposed element shall have a value equal to the value of argument Expose. If argument Expose has a value of true then the property shall be exposed through a Parameter element on an IEEE1451_Dot4PublicXdcr. See Table 38.

Table 38—IEEE1451.4 Transducer Block PublicXdcr managing operations

Operation Name	Requirement
GetNumberOfPublicXdcrs	(m)
GetPublicXdcrList	(m)
GetNumberOfXdcrChannelsInPublicXdcr	(m)
GetAssociatedXdcrChannelListInPublicXdcr	(m)
GetParameterNameList	(m)
ReadParameter	(m)
WriteParameter	(o)
UpdateAndReadParameter	(m)
WriteAndUpdateParameter	(o)
GetPublicXdcrInterpretation	(o)
GetPublicXdcrDimension	(o)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

9.5.1.38 Operation GetNumberOfPublicXdcrs

Description: Operation GetNumberOfPublicXdcrs provides a mechanism to count the total number of `PublicXdcr` objects associated with the `TransducerBlock`.

```
OpReturnCode GetNumberOfPublicXdcrs{
[out] xs:unsignedInt NumberOfPublicXdcrs
}
```

9.5.1.38.1 Argument NumberOfPublicXdcrs

Argument NumberOfPublicXdcrs shall have a value equal to the number of PublicXdcr elements defined in the IEEE1451_Dot4TransducerBlock:PublicXdcrList.

© IEEE 2004 – All rights reserved

9.5.1.39 Operation GetPublicXdcrList

Description: Operation GetPublicXdcrList provides a mechanism to obtain a read-only data structure containing the data model of the PublicXdcrList element associated with the IEEE1451_Dot4`TransducerBlock object`.

```
OpReturnCode GetPublicXdcrList {
[out] IEEE1451_Dot4:IEEE1451_Dot4PublicXdcrListType PublicXdcrList
}
```

9.5.1.39.1 Argument PublicXdcrList

Argument PublicXdcrList shall be set to a value equal to the IEEE1451_Dot4TransducerBlock:PublicXdcrList element.

9.5.1.40 Operation GetNumberOfXdcrChannelsInPublicXdcr

Description: Operation GetNumberOfXdcrChannelsInPublicXdcr provides a mechanism to count the total number of `XdcrChannel` objects associated to the `PublicXdcr object with PublicXdcrID identifier`.

```
OpReturnCode GetNumberOfXdcrChannelsInPublicXdcr {
[in] xs:hexBinary PublicXdcrID,
[out] xs:unsignedInt NumberOfXdcrChannelsInPublicXdcr
}
```

9.5.1.40.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

9.5.1.40.2 Argument NumberOfXdcrChannelsInPublicXdcr

Argument NumberOfXdcrChannelsInPublicXdcr shall be the number of `ID elements` defined in the IEEE1451_Dot4PublicXdcr:AssociteadXdcrChannels sequence element.

9.5.1.41 Operation GetAssociatedXdcrChannelListInPublicXdcr

Description: Operation GetAssociatedXdcrChannelListInPublicXdcr provides a mechanism to obtain a read-only data structure with value equal to the Associated`XdcrChannelList element` associated to the `PublicXdcr object with PublicXdcrID identifier`.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
OpReturnCode GetAssociatedXdcrChannelListInPublicXdcr {
[in] xs:hexBinary PublicXdcrID,
[out]                    IEEE1451_Dot4:IEEE1451_Dot4AssociatedObjectsType
AssociatedXdcrChannels
}
```

9.5.1.41.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

9.5.1.41.2 Argument AssociatedXdcrChannels

Argument AssociatedXdcrChannels shall be set to the value of the IEEE1451_Dot4PublicXdcr:AssociatedXdcrChannels element associated to the `PublicXdcr` object with `PublicXdcrID` identifier.

9.5.1.42 Operation GetParameterNameList

Description: Operation GetParameterNameList provides a mechanism to generate a sequence of all MetaData:ParameterName elements in all IEEE1451_Dot4PublicXdcr:Parameter elements defined in the IEEE1451_Dot4PublicXdcr with PublicXdcrID element.

```
OpReturnCode GetParameterNameList {
[in] xs:hexBinary PublicXdcrID,
[out] IEEE1451_Dot4ParameterNameListType ParameterNameList
}
```

9.5.1.42.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

9.5.1.42.2 Argument ParameterNameList

Argument ParameterNameList shall be a sequence of all MetaData:ParameterName elements in all IEEE1451_Dot4PublicXdcr:Parameter elements defined in the IEEE1451_Dot4PublicXdcr with PublicXdcrID element

9.5.1.43 Operation ReadParameter

Description: Operation ReadParameter provides a mechanism to obtain the data model of Parameter element with MetaData:ParameterName element value equal to argument ParameterName.

```
OpReturnCode ReadParameter {
[in] xs:hexBinary PublicXdcrID,
[in] xs:string ParameterName,
[out] IEEE1451_Dot4:IEEE1451_Dot4ParameterType Parameter
}
```

9.5.1.43.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

9.5.1.43.2 Argument ParameterName

Argument ParameterName shall uniquely identify a Parameter element with value equal to IEEE1451_PublicXdcr:Parameter:MetaData:ParameterName defined in the IEEE1451_Dot4PublicXdcr data model.

ISONORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.43.3 Argument Parameter

Argument Parameter element shall have a value equal to the IEEE1451_Dot4PublicXdcr:Parameter element with MetaData:ParameterName value equal to argument ParameterName.

9.5.1.44 Operation WriteParameter

Description: Operation WriteParameter provides a mechanism to set the value of Parameter element data model with MetaData:ParameterName element value equal to argument ParameterName defined in the IEEE1451_Dot4PublicXdcr with PublicXdcrID element. Operation WriteParameter shall return a MajorReturnCode MJ_FAILED_INPUT_ARGUMENT if the Parameter:MetaData:Permission element has a value of Read or MajorReturnCode MJ_SERVICE_UNAVAILABLE if the IEEE1451_Dot4TransducerBlock lacks a TEDS encoding capability.

```
OpReturnCode WriteParameter {
[in] xs:hexBinary PublicXdcrID,
[in] xs:string ParameterName,
[in] IEEE1451_Dot4:IEEE1451_Dot4ParameterType Parameter
}
```

9.5.1.44.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

9.5.1.44.2 Argument ParameterName

Argument ParameterName shall uniquely identify a Parameter element with value equal to IEEE1451_PublicXdcr:Parameter:MetaData:ParameterName defined in the IEEE1451_Dot4PublicXdcr data model.

9.5.1.44.3 Argument Parameter

IEEE1451_Dot4PublicXdcr:Parameter element with MetaData:ParameterName value equal to argument ParameterName shall be set to a value equal to argument Parameter.

9.5.1.45 Operation UpdateAndReadParameter

Description: Operation UpdateAndReadParameter provides a mechanism to update and then obtain the data model of Parameter element with MetaData:ParameterName element value equal to argument ParameterName. Subclass specific operations shall be responsible to update the current value of Parameter values representing exposed information and Extended Functionality IEEE1451_Dot4Propety objects.

```
OpReturnCode UpdateAndReadParameter {
[in] xs:hexBinary PublicXdcrID,
[in] xs:string ParameterName,
[out] IEEE1451_Dot4:IEEE1451_Dot4ParameterType Parameter
}
```

9.5.1.45.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.45.2 Argument ParameterName

Argument ParameterName shall uniquely identify a Parameter element with value equal to IEEE1451_PublicXdcr:Parameter:MetaData:ParameterName defined in the IEEE1451_Dot4PublicXdcr data model.

9.5.1.45.3 Argument Parameter

Argument Parameter element shall have a value equal to the IEEE1451_Dot4PublicXdcr:Parameter element with MetaData:ParameterName value equal to argument ParameterName.

9.5.1.46 Operation WriteAndUpdateParameter

Description: Operation WriteAndUpdateParameter provides a mechanism to set and update the value of Parameter element data model with MetaData:ParameterName element value equal to argument ParameterName defined in the IEEE1451_Dot4PublicXdcr with PublicXdcrID element. Subclass specific operations shall be responsible to update the current value of Parameter values representing exposed information and Extended Functionality IEEE1451_Dot4Propety objects. Operation WriteAndUpdateParameter shall return a MajorReturnCode MJ_FAILED_INPUT_ARGUMENT if the Parameter:MetaData:Permission element has a value of Read or MajorReturnCode MJ_SERVICE_UNAVAILABLE if the IEEE1451_Dot4TransducerBlock lacks a TEDS encoding capability.

```
OpReturnCode WriteAndUpdateParameter {
[in] xs:hexBinary PublicXdcrID,
[in] xs:string ParameterName,
[in] IEEE1451_Dot4:IEEE1451_Dot4ParameterType Parameter
}
```

9.5.1.46.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

9.5.1.46.2 Argument ParameterName

Argument ParameterName shall uniquely identify a Parameter element with value equal to IEEE1451_PublicXdcr:Parameter:MetaData:ParameterName defined in the IEEE1451_Dot4PublicXdcr data model.

9.5.1.46.3 Argument Parameter

IEEE1451_Dot4PublicXdcr:Parameter element with MetaData:ParameterName value equal to argument ParameterName shall be set to a value equal to argument Parameter.

9.5.1.47 Operation GetPublicXdcrInterpretation

Description: Operation GetPublicXdcrInterpretation provides a mechanism to read the value of the IEEE1451_Dot4PublicXdcr:ParameterInterpretation element of the IEEE1451_Dot4`PublicXdcr` with PublicXdcrID identifier.

```
OpReturnCode GetPublicXdcrInterpretation {
[in] xs:hexBinary PublicXdcrID,
[out]IEEE1451_Dot4:IEEE1451_Dot4PhysicalInterpretationType
ParameterInterpretation
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

9.5.1.47.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

9.5.1.47.2 Argument ParameterInterpretation

Argument ParameterInterpretation shall be set to the value of the IEEE1451_Dot4PublicXdcr:ParameterInterpretation element.

9.5.1.48 Operation GetPublicXdcrDimension

Description: Operation GetPublicXdcrDimension provides a mechanism to read the IEEE1451_Dot4PublicXdcr:Dimension element of the IEEE1451_Dot4`PublicXdcr` with PublicXdcrID identifier.

```
OpReturnCode GetPublicXdcrDimension {
[in] xs:hexBinary PublicXdcrID,
[out] xs:unsignedInt Dimension
}
```

9.5.1.48.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

9.5.1.48.2 Argument Dimension

Argument Dimension shall be set to the value of the IEEE1451_Dot4PublicXdcr:Dimension element. For example, a PublicXdcr associated to three XdcrChannels will have a dimensionality value of 3.

9.5.1.49 Operation GetPublicXdcrCoodinateSystem

Description: Operation GetPublicXdcrCoordinateSystem provides a mechanism to read the IEEE1451_Dot4PublicXdcr:CoordinateSystem element of the IEEE1451_Dot4`PublicXdcr` with PublicXdcrID identifier.

```
OpReturnCode GetPublicXdcrDimension {
[in] xs:hexBinary PublicXdcrID,
[out] IEEE1451_Dot4: IEEE1451_Dot4CoordinateSystemType CoordinateSystem
}
```

9.5.1.49.1 Argument PublicXdcrID

Argument PublicXdcrID shall uniquely identify the `PublicXdcr` object being addressed by the operation.

9.5.1.49.2 Argument CoordinateSystem

Argument CoordinateSystemshall be set to the value of the IEEE1451_Dot4PublicXdcr: CoordinateSystem element.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Annex A

(normative)

IEEE standard templates

A.1 Overview of IEEE templates

The standard defines a collection of templates for common classes of transducers. Table A.1 lists these IEEE standard templates along with their template IDs. For informative purposes, the templates are organized into two types: Transducer Type templates defined for the most common transducer types, and Calibration templates designed to contain calibration information and to be used in conjunction with a Transducer Type.

Table A.1—IEEE templates

Type	Template ID	Name of template
Transducer Type templates	25	Accelerometer/Force transducer w. const. curr. ampl.
	26	Charge amplifier (incl. attached accelerometer)
	27	Microphones w. built-in preamp.
	28	Microphone preamps. w. attached micr. or system
	29	Microphones (capacitive)
	30	High-level voltage output sensors
	31	Current loop output sensors
	32	Resistance sensors
	33	Bridge sensors
	34	AC linear/rotary variable differential transformer (LVDT/RVDT) sensors
	35	Strain gage
	36	Thermocouple
	37	Resistance temperature detectors (RTDs)
	38	Thermistor
	39	Potentiometric voltage divider
Calibration templates	40	Calibration table
	41	Calibration curve (polynomial)
	42	Frequency response table
Transducer Type templates	43	Charge amplifier (incl. attached force transducer)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

A.2 Template summaries

Table A.2, Table A.3, Table A.4, Table A.5, Table A.6, Table A.7, Table A.8, Table A.9, Table A.10, Table A.11, Table A.12, Table A.13, Table A.14, Table A.15, Table A.16, Table A.17, Table A.18, Table A.19, Table A.20, and Table A.21 summarize the contents of each IEEE standard template, which are listed in full in 0. The tables in this subclause are intended to provide an overview of the properties included in each of the templates.

Each row of the table corresponds to a property command (designated in templates with "%") or control command that utilizes bits from the TEDS, such as SELECTCASE and STRUCTARRAY. The columns of the tables include:

Function—a descriptive grouping of property commands and control commands. This column is used to organize the commands in the tables in a convenient fashion for easier understanding of the template contents. The function descriptions in this column are not used in the actual templates.

Select—an indication of distinct cases within a Select Case structure in a template. Properties not located within a Select Case are indicated by a "—" in this column.

Property/Command—the property tag or control command name.

Description—the description of the property tag or command as used in the template.

Access—the access level assigned to the property tag (ID, CAL, or USR).

Bits—the number of bits read from the TEDS for the corresponding property tag or command.

Data Type (and Range)—the data type used for this occurrence of the property. If the data type is ConRes or ConRelRes, the range of values and resolution are indicated within parentheses.

Units—the designated units for the property tag occurrence, if applicable.

Each table also includes the result of a calculation for total bits required to implement the template, referred to in the table as the *Total Bits Required for TEDS*. This is often expressed as a range of values as the total number of bits required will often depend on the particular SELECTCASE values in the template. The bit count included in the following tables does not account for all of the data stored in the TEDS memory. At a minimum, the TEDS memory shall also contain the Basic TEDS (64 bits), selector of descriptor at beginning of template (2 bits), template ID (8 bits), selector of descriptor at end (2 bits) of template, and extended selector (1 bit).

For example, the Bridge Sensor template would have the following bit requirements:

Checksum	8 bits
Basic TEDS	64 bits
Selector of Descriptor	2 bits (value of zero for IEEE standard template)
Total Bits Required for Accelerometer TEDS (range):	110–175 bits (includes 8 bits for Template ID)
Selector of Descriptor (end)	2 bits
Extended selector	1 bit
Total required bits:	187–252 bits

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.2—Accelerometer/Force transducer template (ID = 25) summary

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—		TEMPLATE	Template ID	—	8	Integer (value = 25)	—
Measurement	Select Case—Acceleration (0) / Force (1)					1	Select Case	—
	Case 0	Select Case—Extended Functionality: None (0) / Programmable sensitivity (1)				1	Select Case	—
		Case 0	%Sens@Ref	Sensitivity @ reference condition	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/(m/s^2)
			%TF_HP_S	High-pass cut-off frequency (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
		Case 1	%Passive[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Passive[CtrlFunctionMask]	Control Function Mask for Passive	ID	—	Assign = 0b11	—
			%Passive[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Passive[FunctionType]	Passive control type	ID	—	Assign = 0 (checkmark)	—
			%Passive[Function]	Passive mode	USR	—	BitBin, (Assign = “xx,00”)	—
			%Sens[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Sens[CtrlFunctionMask]	Control Function Mask for Sensitivity	ID	—	Assign = 0b11	
			%Sens[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Sens[FunctionType]	Sensitivity control type	ID	—	Assign = 1 (one exactly)	—
			%Sens[Function]	%Sens@Ref[“10”]	USR	—	BitBin, (Assign = “10”)	—
			%Sens[Function]	%Sens@Ref[“01”]	USR	—	BitBin, (Assign = “01”)	—
			%DefaultFR	Default Sens 00: no, 1: low, 2: high	ID	2	UNINT	—

© IEEE 2004 – All rights reserved

Table A.2—Accelerometer/Force transducer template (ID = 25) summary *(continued)*

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 0 (continued)	Case 1 (continued)	%Passive	Supports multiplexer mode	ID	1	UNINT	—
			%Sens@Ref[“01”]	Low sensitivity @ reference	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/(m/s^2)
			%Sens@Ref[“10”]	High sensitivity @ reference	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/(m/s^2)
			%TF_HP_S[“01”]	Low sens. high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_HP_S[”10”]	High sens. high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
	Case 1	Select Case—Extended Functionality: None (0) / Programmable sensitivity (1)				1	Select Case	—
		Case 0	%Sens@Ref	Sensitivity @ reference condition	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/N
			%TF_HP_S	High-pass cut-off frequency (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%Stiffness	Stiffness of transducer	CAL	6	ConRelRes (1E6 to 8.1E10, ±10%)	N/m
			%Mass_below	Mass below gage	CAL	6	ConRelRes (0.1 to 8114, ±10%)	gram
		Case 1	%Passive[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Passive[CtrlFunctionMask]	Control Function Mask for Passive	ID	—	Assign = 0b11	—
			%Passive[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Passive[FunctionType]	Passive control type	ID	—	Assign = 0 (checkmark)	—
			%Passive[Function]	Passive mode	USR	—	BitBin, (Assign = “xx,00”)	—

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table A.2—Accelerometer/Force transducer template (ID = 25) summary *(continued)*

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 1 (con-tinued)	Case 1 (con-tinued)	%Sens[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Sens[CtrlFunctionMask]	Control Function Mask for Sensitivity	ID	—	Assign = 0b11	
			%Sens[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Sens[FunctionType]	Sensitivity control type	ID	—	Assign = 1 (one exactly)	—
			%Sens[Function]	%Sens@Ref[“10”]	USR	—	BitBin, (Assign = “10”)	
			%Sens[Function]	%Sens@Ref[“01”]	USR	—	BitBin, (Assign = “01”)	
			%DefaultFR	Default Sens 00: no, 1: low, 2: high	ID	2	UNINT	
			%Passive	Supports multiplexer mode	ID	1	UNINT	—
			%Sens@Ref[“01”]	Low sensitivity @ reference	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/N
			%Sens@Ref[“10”]	High sensitivity @ reference	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/N
			%TF_HP_S[“01”]	Low sens. high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_HP_S[”10”]	High sens. high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%Stiffness	Stiffness of transducer	CAL	6	ConRelRes (1E6 to 8.1E10, ±10%)	N/m
			%Mass_below	Mass below gage	CAL	6	ConRelRes (0.1 to 8114, ±10%)	g
			%PhaseCorrection	Phase Correction @ reference condition	CAL	6	ConRes (–3.2 to 3.0, step 0.1)	°
	—	—	%Direction	Sensitivity direction (x, y, z)	CAL	2	Enumeration: x | y | z	—
	—	—	%Weight	Transducer weight	CAL	6	ConRelRes (0.1 to 8114, ±10%)	g
Electrical signal output	—	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 0, “Voltage Sensor”	—
	—	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
	—	—	%ACDCCoupling	AC or DC coupling	ID	—	Assign = 1, “AC”	—
	—	—	%Sign	Polarity (sign)	CAL	1	Enumeration: positive _ negative	—

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.2—Accelerometer/Force transducer template (ID = 25) summary *(continued)*

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
Transfer Function	Select Case—Transfer Function: Not specified (0) / Specified (1)					1	Select Case	
	Case 0	—		No transfer function specified				
	Case 1	—	%TF_SP	Low pass cut-off frequency (F lp)	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
		—	%TF_KPr	Resonance frequency (F res)	CAL	9	ConRelRes (100 to 2.4E+6, ±1%)	Hz
		—	%TF_KPq	Quality factor at F res	CAL	9	ConRelRes (0.4 to 9.7k, ±1%)	—
		—	%TF_SL	Amplitude slope (a)	CAL	7	ConRes (–6.3 to 6.3, step 0.1)	%/decade
		—	%TempCoef	Temperature Coefficient (b)	CAL	6	ConRes (–0.8 to 0.75, step 0.025)	%/°C
Calibration information	—	—	%Reffreq	Reference frequency	CAL	8	ConRelRes (0.35 to 2.18k, ±1.75%)	Hz
	—	—	%RefTemp	Reference temperature (T ref)	CAL	5	ConRes (15 to 30, step 0,5)	°C
	—	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range): 111 to 194 bits								

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.3—Charge amplifier template (incl. attached accelerometer) (ID = 26) summary

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—		TEMPLATE	Template ID	—	8	Integer (value = 26)	—
Measurement	Select Case—Attached Transducer: Not specified (0) / Accelerometer (1)					1	Select Case	—
	Case 0	Select Case—Extended Functionality: None (0) / Programmable gain (1)				1	Select Case	—
		Case 0	%Gain	Gain	CAL	13	ConRelRes (1E+7 to 1.28E+14, ±0.1%)	V/C
			%TF_HP_S	High-pass cut-off frequency (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_SP	Low pass cut-off frequency (F lp)	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
		Case 1	%Passive[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Passive[CtrlFunctionMask]	Control Function Mask for Passive	ID	—	Assign = 0b11	—
			%Passive[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Passive[FunctionType]	Passive control type	ID	—	Assign = 0 (checkmark)	—
			%Passive[Function]	Passive mode	USR	—	BitBin, (Assign = “xx,00”)	—
			%Gain[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Gain[CtrlFunctionMask]	Control Function Mask for Gain	ID	—	Assign = 0b11	
			%Gain[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Gain[FunctionType]	Gain control type	ID	—	Assign = 1 //(one exactly)	—
			%Gain[Function]	%Gain[“10”]	USR	—	BitBin, (Assign = “10”)	
			%Gain[Function]	%Gain[“01”]	USR	—	BitBin, (Assign = “01”)	
			%DefaultFR	Default Gain 00: no, 1: low, 2: high	ID	2	UNINT	

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.3—Charge amplifier template (incl. attached accelerometer) (ID = 26) summary *(continued)*

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 0 (continued)	Case 1 (continued)	%Passive	Supports multiplexer mode	ID	1	UNINT	—
			%Gain[“01”]	Low gain	CAL	13	ConRelRes (1E+7 to 1.28E+14, ±0.1%)	V/C
			%Gain[“10”]	High gain	CAL	13	ConRelRes (1E+7 to 1.28E+14, ±0.1%)	V/C
			%TF_HP_S[“01”]	Low-gain high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_HP_S[“10”]	High-gain high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_SP	Low pass cut-off frequency (F lp)	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
	Case 1	—	%Attached_MfgrID	Manufacturer of attached transducer	CAL	14	UNINT	—
		—	%Attached_ModelNum	Model number of attached transducer	CAL	15	UNINT	—
		—	%Attached_VersionLetter	Version letter of attached transducer	CAL	5	CHR5	—
		—	%Attached_VersionNum	Version number of attached transducer	CAL	6	UNINT	—
		—	%Attached_SerialNum	Serial number of attached transducer	CAL	24	UNINT	—
		Select Case—Extended Functionality: None (0) / Programmable gain (1)				1	Select Case	—
		Case 0	%Sens@Ref	Total sensitivity @ reference condition	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/(m/s^2)
			%TF_HP_S	High-pass cut-off frequency (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_SP	Low pass cut-off frequency (F lp)	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.3—Charge amplifier template (incl. attached accelerometer) (ID = 26) summary *(continued)*

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 1 (continued)	Case 1	%Passive[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Passive[CtrlFunctionMask]	Control Function Mask for Passive	ID	—	Assign = 0b11	—
			%Passive[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Passive[FunctionType]	Passive control type	ID	—	Assign = 0 (checkmark)	—
			%Passive[Function]	Passive mode	USR	—	BitBin, (Assign = “xx,00”)	—
			%Sens[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Sens[CtrlFunctionMask]	Control Function Mask for Sensitivity	ID	—	Assign = 0b11	
			%Sens[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Sens[FunctionType]	Sensitivity control type	ID	—	Assign = 1 (one exactly)	—
			%Sens[Function]	%Sens@Ref[“10”]	USR	—	BitBin, (Assign = “10”)	
			%Sens[Function]	%Sens@Ref[“01”]	USR	—	BitBin, (Assign = “01”)	
			%DefaultFR	Default Sens 00: no, 1: low, 2: high	ID	2	UNINT	
			%Passive	Supports multiplexer mode	ID	1	UNINT	—
			%Sens@Ref[“01”]	Low sensitivity @ reference	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/(m/s^2)
			%Sens@Ref[“10”]	High sensitivity @ reference	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/(m/s^2)
			%TF_HP_S[“01”]	Low sens. high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_HP_S[”10”]	High sens. high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_SP	Low pass cut-off frequency (F lp)	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
		—	%Direction	Sensitivity direction (x, y, z)	CAL	2	Enumeration: x | y | z	—
		Select Case—Transfer Function: Not specified (0) / Specified (1)				1	Select Case	—
		Case 0	—	No transfer function specified	—	—	—	—

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.3—Charge amplifier template (incl. attached accelerometer) (ID = 26) summary ***(continued)***

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 1 (continued)	Case 1	%TF_KPr	Resonance frequency (F res)	CAL	9	ConRelRes (100 to 2.4E+6, ±1%)	Hz
			%TF_KPq	Quality factor at F res	CAL	9	ConRelRes (0.4 to 9.7k, ±1%)	—
			%TF_SL	Amplitude slope (a)	CAL	7	ConRes (–6.3 to 6.3, step 0.1)	%/ decade
			%TempCoef	Temperature coefficient (b)	CAL	6	ConRes (–0.8 to 0.75, step 0.025)	%/°C
	—	—	%PhaseCorrection	Phase Correction @ ref condition	CAL	6	ConRes (–3.2 to 3.0, step 0.1)	°
	—	—	%Reffreq	Reference frequency	CAL	8	ConRelRes (0.35 to 2.18k, ±1.75%)	Hz
	—	—	%RefTemp	Reference temperature (T ref)	CAL	5	ConRes (15 to 30, step 0,5)	°C
Electrical signal output	—	—	%Sign	Polarity (sign)	CAL	1	Enumeration: positive | negative	—
	—	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
	—	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 0, “Voltage Sensor”	—
	—	—	%ACDCCoupling	AC or DC coupling	ID	—	Assign = 1, “AC”	—
Calibration information	—	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	—	%CalPeriod	Calibration period	CAL	12	UNINT	Days
Misc.	—	—	%MeasID	Measurement location ID	USR	11	UNINT	—
				Total bits required for TEDS (range):	112 to 240 bits			

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.4—Microphone template (built-in preamplifier) (ID = 27) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 27)	—
Measurement	Select Case—Extended Functionality: None (0) / Programmable sensitivity (1)				1	Select Case	—
	Case 0	%Sens@Ref	Sensitivity @ reference condition	CAL	16	ConRelRes (10E-6 to 4.916, ±0.01%)	V/Pa
	Case 1	%Passive[Initialize]	Initialization not needed	ID	—	Assign = 0	—
		%Passive[CtrlFunctionMask]	Control Function Mask for Passive	ID	—	Assign = 0b11	—
		%Passive[ReadWrite]	Write only	ID	—	Assign = 3	—
		%Passive[FunctionType]	Passive control type	ID	—	Assign = 0 (checkmark)	—
		%Passive[Function]	Passive mode	USR	—	BitBin, (Assign = “xx,00”)	—
		%Sens[Initialize]	Initialization not needed	ID	—	Assign = 0	—
		%Sens[CtrlFunctionMask]	Control Function Mask for Sensitivity	ID	—	Assign = 0b11	
		%Sens[ReadWrite]	Write only	ID	—	Assign = 3	—
		%Sens[FunctionType]	Sensitivity control type	ID	—	Assign = 1 (one exactly)	—
		%Sens[Function]	%Sens@Ref[“10”]	USR	—	BitBin, (Assign = “10”)	
		%Sens[Function]	%Sens@Ref[“01”]	USR	—	BitBin, (Assign = “01”)	
		%DefaultFR	Default Sens 00: no, 1: low, 2: high	ID	2	UNINT	
		%Passive	Supports multiplexer mode	ID	1	UNINT	—
		%Sens@Ref[“01”]	Low sensitivity @ ref. condition	CAL	16	ConRelRes (10E-6 to 4.916, ±0.01%)	V/Pa
		%Sens@Ref[“10”]	High sensitivity @ ref. condition	CAL	16	ConRelRes (10E-6 to 4.916, ±0.01%)	V/Pa
	—	%Reffreq	Reference frequency	CAL	8	ConRelRes (0.35 to 2.18k, ±1.75%)	Hz
	—	%Refpol	Polarization voltage	CAL	2	Enum: Prepolarized; 28; 200 V	—

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.4—Microphone template (built-in preamplifier) (ID = 27) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
	Select Case—System test (Test gain): Not available (0) / Available (1)				1	Select Case	—
	Case 0	—	No system test available	—	—	—	—
	Case 1	%TestGain	Test gain	CAL	10	ConRes (0 to –102.2, step 0.1)	dB
Type information		%MicType	Microphone Type	CAL	2	Enum: Free | Press | Random | Other	—
		%MicSize	Microphone Size	CAL	2	Enum: 1" | 1/2" | 1/4" | 1/8"	—
		%Equi_Vol	Equivalent microphone volume	CAL	8	ConRes (0 to 254E-9, step 1E-9)	m^3
Transfer Function	Select Case—Transfer Function: Not specified (0) / Specified (1)				1	Select Case	—
	Case 0	—	No transfer function	—	—	—	—
	Case 1	%Resp_Type	Actuator / Corrected response	CAL	1	Enum: Actuator; Corrected	—
		%TF_HP_S	Lowest high-pass cut-off frequency	CAL	7	ConRelRes (0.005 to 821, ±5%)	Hz
		%TF_HP_S	High-pass cut-off frequency	CAL	8	ConRelRes (0.05 to 7.6, ±1%)	Hz
		%TF_SP	F low lift (Low pass cut-off frequency)	CAL	7	ConRelRes (5 to 700, ±2%)	Hz
		%TF_SZm	F high /F low lift	CAL	8	ConRelRes (1 to 2.1, ±0.15%)	Hz
		%TF_KPr	Resonance frequency (F res)	CAL	8	ConRelRes (2k to 306k, ±1%)	Hz
		%TF_KPq	Quality factor at F res	CAL	8	ConRelRes (0.2 to 31, ±1%)	—
		%TF_KPr	Second Resonance frequency (F res)	CAL	6	ConRelRes (10k to 371k, ±3%)	Hz
		%TF_KPq	Quality factor at second F res	CAL	7	ConRelRes (0.2 to 309, ±3%)	—
—	—	%Sign	Polarity (sign)	CAL	1	Enumeration: positive | negative	—
—	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
—	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 0, “Voltage Sensor”	—
—	—	%ACDCCoupling	AC or DC coupling	ID	—	Assign = 1, “AC”	—

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.4—Microphone template (built-in preamplifier) (ID = 27) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	Usr	11	UNINT	—
			Total bits required for TEDS (range):	104 to 193 bits			

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.5—Microphone preamplifier and microphone template (ID = 28) summary

Function	Select1	Select2	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	—	TEMPLATE	Template ID	—	8		—
Basic preamplifier	Select Case—TEDS selection: Preamplifier (0) / Combined with Microphone (1) / Separate TEDS (2)					2	Select Case	—
	Case 0	—	%Rin	Amplifier resistance	CAL	6	ConRelRes (1E+8 to 8.1E+12, ±10%)	Ω
		—	%Cin	Amplifier capacitance	CAL	6	ConRes (0 to 1.24E-12, step 0.02E-12)	F
		—	%TF_HP_S	High-pass cut-off frequency	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
		—	%TF_SP	Low pass cut-off frequency	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
		—	%Rout	Output impedance	CAL	5	ConRelRes (5 to 1.2E+3, ±10%)	Ω
		—	%Gate	Gate present	CAL	1	Enum: No | Yes	—
Extended functionality		Select Case—Extended Functionality: None (0) / Programmable gain (1)				1	Select Case	—
		Case 0	%Gain	Gain	CAL	12	ConRes (–40 to 62, step 0.025)	dB
		Case 1	%Passive[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Passive[CtrlFunctionMask]	Control Function Mask for Passive	ID	—	Assign = 0b11	—
			%Passive[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Passive[FunctionType]	Passive control type	ID	—	Assign = 0 (checkmark)	—
			%Passive[Function]	Passive mode	USR	—	BitBin, (Assign = “xx,00”)	—
			%Gain[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Gain[CtrlFunctionMask]	Control Function Mask for Gain	ID	—	Assign = 0b11	—
			%Gain[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Gain[FunctionType]	Gain control type	ID	—	Assign = 1 (one exactly)	—

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.5—Microphone preamplifier and microphone template (ID = 28) summary *(continued)*

Function	Select1	Select2	Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 0 (contin-ued)	Case 1 (contin-ued)	%Gain[Function]	%Gain["10"]	USR	—	BitBin, (Assign = "10")	—
			%Gain[Function]	%Gain["01"]	USR	—	BitBin, (Assign = "01")	—
			%DefaultFR	Default Gain 00: no, 1: low, 2: high	ID	2	UNINT	—
			%Passive	Supports multiplexer mode	ID	1	UNINT	—
			%Gain["01"]	Low gain	CAL	12	ConRes (–40 to 62, step 0.025)	dB
			%Gain["10"]	High gain	CAL	12	ConRes (–40 to 62, step 0.025)	dB
Appended attachment info		Select Case—Microphone Appended TEDS: Not specified (0) / Specified (1)				1		—
		Case 0		No Appended Microphone TEDS				—
			Basic TEDS	Microphone Basic TEDS		64		—
		Case 1	%Appended_TEDS	Appended Microphone TEDS	USR	1	Enum: No | Yes	—
			%Appended_TEDS_location	Appended Microphone TEDS location	USR	—	String5 (0 to 31 char)	—
System info / Extended functionality	Case 1	Select Case—Extended Functionality: None (0) / Programmable sensitivity (1)				1	Select Case	—
		Case 0	%Sens@Ref	Total sensitivity @ reference condition	CAL	16	ConRelRes (10E-6 to 4.9, ±0.01%)	V/Pa
		Case 1	%Passive[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Passive[CtrlFunctionMask]	Control Function Mask for Passive	ID	—	Assign = 0b11	—
			%Passive[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Passive[FunctionType]	Passive control type	ID	—	Assign = 0 (checkmark)	—
			%Passive[Function]	Passive mode	USR	—	BitBin, (Assign = "xx,00")	—
			%Sens[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Sens[CtrlFunctionMask]	Control Function Mask for Sensitivity	ID	—	Assign = 0b11	—

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.5—Microphone preamplifier and microphone template (ID = 28) summary *(continued)*

Function	Select1	Select2	Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 1 (continued)	Case 1 (continued)	%Sens[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Sens[FunctionType]	Sensitivity control type	ID	—	Assign = 1 (one exactly)	—
			%Sens[Function]	%Sens@Ref[“10”]	USR	—	BitBin, (Assign = “10”)	—
			%Sens[Function]	%Sens@Ref[“01”]	USR	—	BitBin, (Assign = “01”)	—
			%DefaultFR	Default sens 00: no, 1: low, 2: high	ID	2	UNINT	—
			%Passive	Supports multiplexer mode	ID	1	UNINT	—
			%Sens@Ref[“01”]	Low total sensitivity @ ref. condition	CAL	16	ConRelRes (10E-6 to 4.916, ±0.01%)	V/Pa
			%Sens@Ref[“10”]	High total sensitivity @ ref. condition	CAL	16	ConRelRes (10E-6 to 4.916, ±0.01%)	V/Pa
System info		Select Case—Basic TEDS for: Microphone (0) / System (1)				1		—
		Case 0	Basic TEDS	Microphone Basic TEDS (Attached)		64		—
		Case 1	Basic TEDS	System Basic TEDS (System)		64		—
		—	%Refpol	Polarization voltage	CAL	2	Enum: Prepolarized; 28; 200 V	—
		—	%MicType	Microphone Type	CAL	2	Enum: Free | Press | Random | Other	—
		—	%MicSize	Microphone Size	CAL	2	Enum: 1" | 1/2" | 1/4" | 1/8"	—
		Select Case—System test (Test gain): Not available (0) / Available (1)				1		—
		Case 0	—	No system test available				—
		Case 1	%TestGain	Test gain	CAL	10	ConRes (0 to –102.2, step 0.1)	dB
		—	%Equi_Vol	Equivalent microphone volume	CAL	8	ConRes (0 to 254E-9, step 1E-9)	m^3

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.5—Microphone preamplifier and microphone template (ID = 28) summary *(continued)*

Function	Select1	Select2	Property/Command	Description	Access	Bits	Data type (and range)	Units
Transfer Function	Case 1 (continued)	Select Case—Transfer Function: Not specified (0) / Specified (1)				1	Select Case	—
		Case 0		No transfer function		—		—
		Case 1	%Resp_Type	Actuator / Corrected response	CAL	—	Enum: Actuator; Corrected	—
			%TF_HP_S	Lowest High-pass cut-off frequency	CAL	7	ConRelRes (0.005 to 821, ±5%)	Hz
			%TF_HP_S	High-pass cut-off frequency	CAL	8	ConRelRes (0.05 to 7.6, ±1%)	Hz
			%TF_SP	F low lift (Low pass cut-off frequency)	CAL	7	ConRelRes (5 to 700, ±2%)	Hz
			%TF_SZm	F high /F low lift	CAL	8	ConRelRes (1 to 2.1, ±0.15%)	Hz
			%TF_KPr	Resonance frequency (f res)	CAL	8	ConRelRes (2k to 312k, ±1%)	Hz
			%TF_KPq	Quality factor at f res	CAL	8	ConRelRes (0.2 to 31, ±1%)	—
			%TF_KPr	Second Resonance frequency (f res)	CAL	6	ConRelRes (10k to 371k, ±3%)	Hz
			%TF_KPq	Quality factor at second f res	CAL	7	ConRelRes (0.2 to 309, ±3%)	—
			%TF_SP	Low pass cut-off frequency	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
Basic preamplifier	Case 2	—	%Rin	Amplifier resistance	CAL	6	ConRelRes (1E+8 to 8.1E+12, ±10%)	Ω
		—	%Cin	Amplifier capacitance	CAL	6	ConRes (0 to 1.24E-12, step 0.02E-12)	F
		—	%TF_HP_S	High-pass cut-off frequency	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
		—	%TF_SP	Low pass cut-off frequency	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
		—	%Rout	Output impedance	CAL	5	ConRelRes (5 to 1.2E+3, 10%)	Ω
		—	%Gate	Gate present	CAL	1	Enum: No | Yes	—

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table A.5—Microphone preamplifier and microphone template (ID = 28) summary *(continued)*

Function	Select1	Select2	Property/Command	Description	Access	Bits	Data type (and range)	Units
Extended functionality	Case 2 (continued)	Select Case—Extended Functionality: None (0) / Programmable gain (1)				1	Select Case	—
		Case 0	%Gain	Gain	CAL	12	ConRes (–40 to 62, step 0.025)	dB
		Case 1	%Passive[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Passive[CtrlFunctionMask]	Control Function Mask for Passive	ID	—	Assign = 0b11	—
			%Passive[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Passive[FunctionType]	Passive control type	ID	—	Assign = 0 (checkmark)	—
			%Passive[Function]	Passive mode	USR	—	BitBin, (Assign = “xx,00”)	—
			%Gain[Initialize]	Initialization not needed	ID	—	Assign = 0	—
			%Gain[CtrlFunctionMask]	Control Function Mask for Gain	ID	—	Assign = 0b11	—
			%Gain[ReadWrite]	Write only	ID	—	Assign = 3	—
			%Gain[FunctionType]	Gain control type	ID	—	Assign = 1 (one exactly)	—
			%Gain[Function]	%Gain[“10”]	USR	—	BitBin, (Assign = “10”)	—
			%Gain[Function]	%Gain[“01”]	USR	—	BitBin, (Assign = “01”)	—
			%DefaultFR	Default gain 00: no, 1: low, 2: high	ID	2	UNINT	—
			%Passive	Supports multiplexer mode	ID	1	UNINT	—
			%Gain[“01”]	Low gain	CAL	12	ConRes (–40 to 62, step 0.025)	dB
			%Gain[“10”]	High gain	CAL	12	ConRes (–40 to 62, step 0.025)	dB

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table A.5—Microphone preamplifier and microphone template (ID = 28) summary *(continued)*

Function	Select1	Select2	Property/Command	Description	Access	Bits	Data type (and range)	Units
Microphone calibration info	Case 2 (continued)	—		Microphone Basic TEDS (Attached)		64		—
		—	%Sens@Ref	Sens@Ref	CAL	16	ConRelRes (10E-6 to 4.9, ±0.01%)	V/Pa
		—	%RefPol	Polarization voltage	CAL	2	Enum: Prepolarized; 28; 200 V	—
		—	%MicType	Microphone type	CAL	2	Enum: Free | Press | Random | Other	—
		—	%MicSize	Microphone size	CAL	2	Enum: 1" | 1/2" | 1/4" | 1/8"	—
		—	%Cmic	Active capacitance	CAL	9	ConRes (5E-12 to 158E-12, step 0.3E-12)	F
		—	%Cstray	Passive capacitance	CAL	5	ConRelRes (0.15E-12 to 35.6E-12, ±10%)	F
		—	%Equi_Vol	Equivalent microphone volume	CAL	8	ConRes (0 to 254E-9, step 1E-9)	m^3
		—	%TF_HP_S	High-pass cut-off frequency	CAL	8	ConRelRes (0.05 to 7.6, ±1%)	Hz
		—	%RefTemp	Reference temperature	CAL	5	ConRes (15 to 30, step 0,5)	°C
		—	%Refpress	Reference pressure	CAL	6	ConRes (80000 to 111000, step 500)	Pa
		—	%Reffreq	Reference frequency	CAL	8	ConRelRes (0.35 to 2.18k, ±1.75%)	Hz
		—	%CalDate	Calibration Date	CAL	16	DATE	—
		—	%CalPeriod	Calibration period	CAL	12	UNINT	days

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.5—Microphone preamplifier and microphone template (ID = 28) summary *(continued)*

Function	Select1	Select2	Property/Command	Description	Access	Bits	Data type (and range)	Units
Calibration information	—	—	%Sign	Polarity (sign)	CAL	1	Enumeration: positive | negative	—
	—	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
	—	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 0, “Voltage Sensor”	—
	—	—	%ACDCCoupling	AC or DC coupling	ID	—	Assign = 1, “AC”	—
	—	—	%RefTemp	Reference temperature	CAL	5	ConRes (15 to 30, step 0,5)	°C
	—	—	%Refpress	Reference pressure	CAL	6	ConRes (80000 to 111000, step 500)	Pa
	—	—	%Reffreq	Reference frequency	CAL	8	ConRelRes (0.35 to 2.18k, ±1.75%)	Hz
	—	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (different selections):					116 to 350 bits			

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.6—Capacitive microphone template (ID = 29)

Function	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	TEMPLATE	Template ID	—	8	Integer (value = 29)	—
Calibration information	%Sens@Ref	Sens@Ref	CAL	16	ConRelRes (10E-6 to 4.9, ±0.01%)	V/Pa
	%RefPol	Polarization voltage	CAL	2	Enum: Prepolarized; 28; 200 V	—
	%MicType	Microphone Type	CAL	2	Enum: Free | Press | Random | Other	—
	%MicSize	Microphone Size	CAL	2	Enum: 1" | 1/2" | 1/4" | 1/8"	—
	%Cmic	Active capacitance	CAL	9	ConRes (5E-12 to 158E-12, step 0.3E-12)	F
	%Cstray	Passive capacitance	CAL	5	ConRelRes (0.15E-12 to 35.6E-12, ±10%)	F
	%Equi_Vol	Equivalent microphone volume	CAL	8	ConRes (0 to 254E-9, step 1E-9)	m^3
	%TF_HP_S	High-pass cut-off frequency	CAL	8	ConRelRes (0.05 to 7.6, ± 1%)	Hz
	%RefTemp	Reference temperature	CAL	5	ConRes (15 to 30, step 0,5)	°C
	%Refpress	Reference pressure	CAL	6	ConRes (80000 to 111000, step 500)	Pa
	%Reffreq	Reference frequency	CAL	8	ConRelRes (0.35 to 2.18k, ±1.75%)	Hz
	%CalInitials	Calibration Initials	CAL	15	CHR5	—
	%CalDate	Calibration Date	CAL	16	DATE	—
	%CalPeriod	Calibration period	CAL	12	UNINT	days
				Total bits required for TEDS:	122	

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.7—High-level voltage output template (ID = 30) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 30)	—
Measurement	Select Case—Physical Measurand				6	Select Case	—
	Cases 0–45	%MinPhysVal	Minimum physical value	CAL	32	Single	Various[a]
		%MaxPhysVal	Maximum physical value	CAL	32	Single	Various[a]
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 0, “Voltage Sensor”	—
	Select Case—Full-Scale Electrical Value Precision				2	Select Case	—
	Case 0	%MinElecVal	Minimum voltage output	CAL	—	Assign = 0.0	V
		%MaxElecVal	Maximum voltage output	CAL	—	Assign = 10.0	V
	Case 1	%MinElecVal	Minimum voltage output	CAL	—	Assign = -10.0	V
		%MaxElecVal	Maximum voltage output	CAL	—	Assign = 10.0	V
	Case 2	%MinElecVal	Minimum voltage output	CAL	11	ConRes (–20.5 to 20.4, step 0.02)	V
		%MaxElecVal	Maximum voltage output	CAL	11	ConRes (–20.5 to 20.4, step 0.02)	V
	Case 3	%MinElecVal	Minimum voltage output	CAL	32	Single	V
		%MaxElecVal	Maximum voltage output	CAL	32	Single	V
	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
	—	%ACDCCoupling	AC or DC coupling	ID	1	Enumeration: DC | AC	—
	—	%SensorImped	Sensor output impedance	ID	12	ConRelRes (1 to 1.1M, ±0.17%)	Ω
	—	%RespTime	Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s

Table A.7—High-level voltage output template (ID = 30) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
Power supply	Select Case—Excitation/Power Requirements				1	Select Case	—
	Case 0	—	No power supply or excitation source	—	—	—	—
	Case 1	%ExciteAmplNom	Power-supply level, nominal	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
		%ExciteAmplMin	Power-supply level, min.	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
		%ExciteAmplMax	Power-supply level, max	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
		%ExciteType	Power-supply type	ID	2	Enumeration: DC | Bipolar DC | AC	—
		%ExciteCurrentDraw	Max current at nominal power level	ID	6	ConRelRes (1E-6 to 1.6, ±13%)	A
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range):					154 to 253 bits		

[a]Units for %MinPhysVal and %MaxPhysVal are determined by value of the Select Case “Physical Measurand” as summarized in Table A.22.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.8—Current Loop Output sensors template (ID = 31) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 31)	—
Measurement	Select Case—Physical Measurand				6	Select Case	—
	Cases 0–45	%MinPhysVal	Minimum physical value	CAL	32	Single	Various[a]
		%MaxPhysVal	Maximum physical value	CAL	32	Single	Various[a]
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign =1, “Current Sensor”	—
	Select Case—Full-Scale Electrical Value Precision				1	Select Case	—
	Case 0	%MinElecVal	Minimum current output	CAL	—	Assign = 4.0	mA
		%MaxElecVal	Maximum current output	CAL	—	Assign = 20.0	mA
	Case 1	%MinElecVal	Minimum current output	CAL	32	Single	A
		%MaxElecVal	Maximum current output	CAL	32	Single	A
	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
	—	%RespTime	Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Power supply	Select Case—Loop powered versus external powered				1	Select Case	—
	Case 0	%LoopSuppyMin	Minimum compliance	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
		%LoopSuppyMax	Maximum compliance	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
	Case 1	%ExciteAmplNom	Power-supply level, nominal	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
		%ExciteAmplMin	Power-supply level, min.	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
		%ExciteAmplMax	Power-supply level, max	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
		%ExciteType	Power-supply type	ID	1	Enumeration: DC | Bipolar DC	—
		%ExciteCurrentDraw	Max current at nominal power	ID	6	ConRelRes (1E-6 to 1.6, ±13%)	A

Table A.8—Current Loop Output sensors template (ID = 31) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range):					158 to 238 bits		

[a]Units for %MinPhysVal and %MaxPhysVal are determined by value of the Select Case “Physical Measurand” as summarized in Table A.22.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.9—Resistance sensors template (ID = 32) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 32)	—
Measurement	Select Case—Physical Measurand				6	Select Case	—
	Cases 0–45	%MinPhysVal	Minimum physical value	CAL	32	Single	Various[a]
		%MaxPhysVal	Maximum physical value	CAL	32	Single	Various[*]
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 2, “Resistance Sensor”	—
	Select Case—Full-Scale Electrical Value Precision				2	Select Case	—
	Case 0	%MinElecVal	Minimum resistance	CAL	7	ConRes (0 to 1.3k, step 10)	Ω
		%MaxElecVal	Maximum resistance	CAL	7	ConRes (0 to 1.3k, step 10)	Ω
	Case 1	%MinElecVal	Minimum resistance	CAL	10	ConRes (0 to 1M, step 1k)	Ω
		%MaxElecVal	Maximum resistance	CAL	10	ConRes (0 to 1M, step 1k)	Ω
	Case 2	%MinElecVal	Minimum resistance	CAL	16	ConRes (0 to 65.4k, step 1)	Ω
		%MaxElecVal	Maximum resistance	CAL	16	ConRes (0 to 65.4k, step 1)	Ω
	Case 3	%MinElecVal	Minimum resistance	CAL	32	Single	Ω
		%MaxElecVal	Maximum resistance	CAL	32	Single	Ω
	—	%MapMeth	Mapping Method	ID	2	Enumeration: Linear | Inverse m/(x+b) | Inverse b+m/x | Inverse 1/(b+m/x)	—
	—	%RespTime	Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Power supply	—	%ExciteAmplNom	Excitation current, nominal	ID	8	ConRelRes (1E-6 to 0.12, ±2%)	A
	—	%ExciteAmplMax	Excitation current, max	ID	8	ConRelRes (1E-6 to 0.12, ±2%)	A

Table A.9—Resistance sensors template (ID = 32) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range):					172 to 222 bits		

[a]Units for %MinPhysVal and %MaxPhysVal are determined by value of the Select Case “Physical Measurand” as summarized in Table A.22.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.10—Bridge sensors template (ID = 33) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 33)	—
Measurement	Select Case—Physical Measurand				6	Select Case	—
	Cases 0–45	%MinPhysVal	Minimum physical value	CAL	32	Single	Various[a]
		%MaxPhysVal	Maximum physical value	CAL	32	Single	Various[a]
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 3, “Bridge Sensor”	—
	Select Case—Full-Scale Electrical Value Precision				2	Select Case	—
	Case 0	%MinElecVal	Minimum electrical output	CAL	11	ConRes (±1, step 1E-3)	V/V
		%MaxElecVal	Maximum electrical output	CAL	11	ConRes (±1, step 1E-3)	V/V
	Case 1	%MinElecVal	Minimum electrical output	CAL	19	ConRes (±6.55E-3, step 25E-9)	V/V
		%MaxElecVal	Maximum electrical output	CAL	19	ConRes (±6.55E-3, step 25E-9)	V/V
	Case 2	%MinElecVal	Minimum electrical output	CAL	32	Single	V/V
		%MaxElecVal	Maximum electrical output	CAL	32	Single	V/V
	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
	—	%BridgeType	Bridge Type	ID	2	Enumeration: Quarter | Half | Full	—
	—	%SensorImped	Bridge element impedance	ID	18	ConRes (1 to 26.2k, step 0.1)	Ω
	—	%RespTime	Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Excitation supply	—	%ExciteAmplNom	Excitation level, nominal	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
	—	%ExciteAmplMin	Excitation level, min.	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
	—	%ExciteAmplMax	Excitation level, max	ID	9	ConRes (0.1 to 51.1, step 0.1)	V

Table A.10—Bridge sensors template (ID = 33) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range):					209 to 251 bits		

[a]Units for %MinPhysVal and %MaxPhysVal are determined by value of the Select Case “Physical Measurand” as summarized in Table A.22.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.11—AC Linear/Rotary Variable Differential Transformer (LVDT/RVDT) template (ID = 34) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 34)	—
Measurement	Select Case—Physical Measurand				3	Select Case	—
	Cases 0–4	%MinPhysVal	Negative full-scale position	CAL	32	Single	m, mm, in, radians, deg.
		%MaxPhysVal	Positive full-scale position	CAL	32	Single	m, mm, in, radians, deg.
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 4, “LVDT Sensor”	—
	Select Case—Full-Scale Electrical Value Precision				1	Select Case	—
	Case 0	%MinElecVal	Electrical output at negative full-scale	CAL	11	ConRes (-1 to 1, step 0.001)	V/V
		%MaxElecVal	Electrical output at positive full-scale	CAL	11	ConRes (-1 to 1, step 0.001)	V/V
	Case 1	%MinElecVal	Electrical output at negative full-scale	CAL	32	Single	V/V
		%MaxElecVal	Electrical output at positive full-scale	CAL	32	Single	V/V
	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
	—	%RespTime	Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Excitation supply	—	%ExciteAmplNom	Excitation amplitude, nominal	ID	8	ConRes (0.1 to 25.5, step 0,1)	V
	—	%ExciteAmplMax	Excitation amplitude, max	ID	8	ConRes (0.1 to 25.5, step 0,1)	V
	—	%ExciteType	Excitation voltage type	ID	—	Assign = 2, “AC (rms)”	—
	—	%ExciteFreqNom	Excitation frequency, nominal	ID	12	ConRes (1 to 40,950, step 10)	Hz
	—	%ExciteFreqMin	Excitation frequency, minimum	ID	12	ConRes (1 to 40,950, step 10)	Hz
	—	%ExciteFreqMax	Excitation frequency, maximum	ID	12	ConRes (1 to 40,950, step 10)	Hz
	—	%SensorImped	Sensor input impedance	ID	7	ConRelRes (1 to 32.7k, ±4%)	Ω

Table A.11—AC Linear/Rotary Variable Differential Transformer (LVDT/RVDT) template (ID = 34) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range):					217 to 259 bits		

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.12—Strain-gage template (ID = 35) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 35)	—
Measurement	—	%MinPhysVal	Negative full-scale strain	CAL	14	ConRes (–0.4 to 0.4, step 50E-6)	strain
	—	%MaxPhysVal	Positive full-scale strain	CAL	14	ConRes (–0.4 to 0.4, step 50E-6)	strain
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 3, “Bridge Sensor”	—
	—	%MinElecVal	Minimum electrical output	CAL	14	ConRes (–1 to 1, step 125E-6)	V/V
	—	%MaxElecVal	Maximum electrical output	CAL	14	ConRes (–1 to 1, step 125E-6)	V/V
	—	%MapMeth	Mapping Method	ID	—	Assign = 7, “Bridge”	—
	—	%GageType	Gage Type	ID	5	Enumeration	—
	—	%GageFactor	Sensitivity of strain gage	CAL	13	ConRelRes (1.5 to 1500, ±0.0422%)	—
	—	%GageTransSens	Transverse sensitivity	CAL	9	ConRes(–5 to 5, step 0.02)	%
	—	%GageOffset	Zero offset after installation	USR	20	ConRes (–50E-3 to 50E-3, step 100E-9)	V/V
	—	%PoissonCoef	Poisson Coefficient after installation	USR	14	ConRes (0 to 1, step 1E-4)	—
	—	%YoungsMod	Young’s Modulus	USR	14	ConRes (0 to 1.6E+6, step 100)	MPa
	—	%GageArea	Area of each gage element	ID	7	ConRelRes(0.2 to 3250, ±4%)	mm^2
	—	%BridgeType	Type of bridge	ID	2	Enumeration: Quarter | Half | Full	—
	—	%SensorImped	Resistance of gage (uninstalled)	ID	13	ConRelRes (50 to 7.9k, ±0.0328%)	Ω
	—	%RespTime	Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Excitation supply	—	%ExciteAmplNom	Excitation voltage, nom.	ID	8	ConRes (0.1 to 25.5, step 0,1)	V
	—	%ExciteAmplMax	Excitation voltage, max.	ID	8	ConRes (0.1 to 25.5, step 0,1)	V

Table A.12—Strain-gage template (ID = 35) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
			Total bits required for TEDS:		237 bits		

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.13—Thermocouple template (ID = 36) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID		8	Integer (value = 36)	—
Measurement	—	%MinPhysVal	Minimum temperature	CAL	11	ConRes (–273 to 1,770, step 1)	ºC
	—	%MaxPhysVal	Maximum temperature	CAL	11	ConRes (–273 to 1,770, step 1)	ºC
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 0, “Voltage Sensor”	—
	—	%MinElecVal	Minimum electrical output	CAL	7	ConRes (–25E-3 to 0.1 step 1E-3)	V
	—	%MaxElecVal	Maximum electrical output	CAL	7	ConRes (–25E-3 to 0.1 step 1E-3)	V
	—	%MapMeth	Mapping Method	ID	—	Assign = 4, “Thermocouple”	—
	—	%TCType	Thermocouple type	ID	4	Enumeration: B | E | J | K | N | R | S | T | Non-std.	—
	—	%CJSource	Cold-junction compensation required	ID	1	Enumeration: CJC required | Compensated	—
	—	%SensorImped	Thermocouple resistance	ID	12	ConRelRes (1 to 319k, ±0.155%)	Ω
	—	%RespTime	Sensor Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
				Total bits required for TEDS (range):	121 bits		

Table A.14—Resistance temperature detectors (RTDs) template (ID = 37) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 37)	—
Measurement	—	%MinPhysVal	Minimum temperature	CAL	11	ConRes (–200 to 1,846, step 1)	ºC
	—	%MaxPhysVal	Maximum temperature	CAL	11	ConRes (–200 to 1,846, step 1)	ºC
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 2, “Resistance Sensor”	—
	—	%MinElecVal	Minimum electrical output	CAL	11	ConRes (0 to 2.05 k, step 1)	Ω
	—	%MaxElecVal	Maximum electrical output	CAL	13	ConRes (0 to 8.2 k, step 1)	Ω
	—	%MapMeth	Mapping Method	ID	—	Assign = 5, “RTD”	—
	Select Case—R0 Resistance				2	Select Case	—
	Case 0	%RTDCoef_R0	Resistance R0	ID	—	Assign = 100.0	Ω
	Case 1	%RTDCoef_R0	Resistance R0	ID	—	Assign = 120.0	Ω
	Case 2	%RTDCoef_R0	Resistance R0	ID	—	Assign = 1000.0	Ω
	Case 3	%RTDCoef_R0	Resistance R0	ID	20	ConRelRes (1 to 12.5k, ±4.5 ppm)	Ω
	Select Case—RTD Curve (Callendar-Van Dusen Coefficients)				3	Select Case	—
	Case 0	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.8100E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = –6.0200E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = –6.000E-12	$1/C^3$
	Case 1	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9083E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = –5.7750E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = –4.183E-12	$1/C^3$

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table A.14—Resistance temperature detectors (RTDs) template (ID = 37) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 2	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9692E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = –5.8495E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = –4.229E-12	$1/C^3$
	Case 3	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9739E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = –5.8700E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = –4.39E-12	$1/C^3$

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table A.15—Resistance temperature detectors (RTDs) template (ID = 37) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 4	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9787E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = –5.8685E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = –4.160E-12	$1/C^3$
	Case 5	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9888E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = –5.915E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = –3.816E-12	$1/C^3$
	Case 6	%RTDCoef_A	CVD Coefficient A	ID	13	ConRes (3.8E-3 to 4E-3, step 2.5E-8)	1/C
		%RTDCoef_B	CVD Coefficient B	ID	10	ConRes (–6.1E-7 to –5.6E-7, step 5E-11)	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	7	ConRes (–6E-12 to –3E-12, step 2.3E-14)	$1/C^3$
	Case 7	%RTDCoef_A	CVD Coefficient A	ID	32	Single	1/C
		%RTDCoef_B	CVD Coefficient B	ID	32	Single	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	32	Single	$1/C^3$
	—	%RespTime	Sensor Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Excitation or power supply	—	%ExciteAmplNom	Excitation current, nom	CAL	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A
	—	%ExciteAmplMax	Excitation current, max	ID	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range):					135 to 251 bits		

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.16—Thermistor template (ID = 38) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 38)	—
Measurement	—	%MinPhysVal	Minimum temperature	CAL	11	ConRes (–200 to 1,846, step 1)	ºC
	—	%MaxPhysVal	Maximum temperature	CAL	11	ConRes (–200 to 1,846, step 1)	ºC
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 2, “Resistance Sensor”	—
	—	%MinElecVal	Minimum resistance output	CAL	18	ConRes (0 to 262k, step 1)	Ω
	—	%MaxElecVal	Maximum resistance output	CAL	18	ConRes (0 to 262k, step 1)	Ω
	—	%MapMeth	Mapping Method	ID	—	Assign = 6, “Thermistor”	—
	—	%RTDCoef_R0	Resistance of thermistor at 0°C	ID	20	ConRelRes (10 to 5.5E+6, ±6.3ppm)	Ω
	—	%SteinhartA	Steinhart-Hart Coefficient A	ID	32	Single	1/C
	—	%SteinhartB	Steinhart-Hart Coefficient B	ID	32	Single	1/C
	—	%SteinhartC	Steinhart-Hart Coefficient C	ID	32	Single	1/C
	—	%RespTime	Sensor Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Excitation or power supply	—	%ExciteAmplNom	Nominal current excitation	CAL	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A
	—	%ExciteAmplMax	Maximum current excitation	ID	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A
	—	%SelfHeating	Self-heating constant	ID	5	ConRelRes (0.25E-3 to 16E-3, ±7.4%)	W/ºC
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range):					263 bits		

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.17— Potentiometric voltage divider template (ID = 39) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 39)	—
Measurement	Select Case—Physical Measurand				6	Select Case	—
	Cases 0–45	%MinPhysVal	Minimum physical value	CAL	32	Single	Various[a]
		%MaxPhysVal	Maximum physical value	CAL	32	Single	Various[a]
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 5, “Potentiometric Voltage Divider Sensor”	—
	Select Case—Electrical Value Precision				1	Select Case	—
	Case 0	%MinElecVal	Minimum electrical output	CAL	—	Assign = 0.0	V/V
		%MaxElecVal	Maximum electrical output	CAL	—	Assign = 1.0	V/V
	Case 1	%MinElecVal	Minimum electrical output	CAL	20	ConRes (0 to 1, step 1E-6)	V/V
		%MaxElecVal	Maximum electrical output	CAL	20	ConRes (0 to 1, step 1E-6)	V/V
	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
	—	%SensorImped	Sensor input impedance	ID	12	ConRelRes (1 to 1.1M, ±0.17%)	Ω
	—	%RespTime	Sensor Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Excitation voltage supply	—	%ExciteAmplNom	Excitation level, nominal	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
	—	%ExciteAmplMin	Excitation level, min	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
	—	%ExciteAmplMax	Excitation level, max	ID	9	ConRes (0.1 to 51.1, step 0.1)	V
	—	%ExciteType	Power-supply type	ID	2	Enumeration: DC | Bipolar DC | AC	—
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range):					180 to 220 bits		

[a]Units for %MinPhysVal and %MaxPhysVal are determined by value of the Select Case “Physical Measurand” as summarized in Table A.22.

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.18—Calibration table template (ID = 40) summary

Function	Property/Command		Description	Access	Bits	Data type (and range)	Units
ID	TEMPLATE		Template ID	—	8	Integer (value = 40)	—
Table data	%CalTable_Domain		Domain parameter	CAL	1	Enumeration: Electrical | Physical	—
	STRUCTARRAY CalTable		Number of data sets	CAL	7	Dimension size of 1 to 127	—
		%CalPoint_DomainValue	Domain Calibration Point (% of full span)	CAL	16	ConRes (0 to 100, step 0.0015)	%
		%CalPoint_RangeValue	Range Calibration Deviation (% of full span)	CAL	21	ConRes (–100 to 100, step 0.0001)	%
Total bits required for TEDS (range):					16 to 4715 bits		

Table A.19—Calibration curve template (ID = 41) summary

Function	Property/Command			Description	Access	Bits	Data type (and range)	Units
ID	TEMPLATE			Template ID	—	8	Integer (value = 41)	—
Curve data	%CalCurve_Domain			Domain parameter	CAL	1	Enumeration: Electrical | Physical	—
	STRUCTARRAY CalCurve			Number of Calibration Curve Segments	CAL	8	Dimension size of 1 to 255	—
		%CalCurve_PieceStart		Start of segment (array of size CalCurve)	CAL	13	ConRes (0 to 100, step 0.0123)	%
		STRUCTARRAY CalCurve_Poly		Number of polynomials	CAL	7	Dimension size of 1 to 127	—
			%CalCurve_Power	Power of domain value (2D array with dimension sizes of CalCurve and CalCurve_Poly)	CAL	7	ConRes (–32 to 32, step 0.5)	—
			%CalCurve_Coef	Polynomial coefficient (2D array with dimension sizes of CalCurve and CalCurve_Poly)	CAL	32	Single	—
Total bits required for TEDS (range):						17 to 1268132 bits		

Table A.20—Frequency response table template (ID = 42) summary

Function	Property/Command		Description	Access	Bits	Data type (and range)	Units
ID	TEMPLATE		Template ID	—	8	Integer (value = 42)	—
Table Data	STRUCTARRAY TF_Table		Transfer function table	CAL	7	Dimension size of 1 to 127	—
		%TF_Table_Freq	Frequency (array of size TF_Table)	CAL	15	ConRelRes (1 to 1.3E+6, ±0.02%)	Hz
		%TF_Table_Ampl	Amplitude (array of size TF_Table)	CAL	21	ConRes (–100 to 100, step 0.0001)	%
			Total bits required for TEDS (range):		15 to 4587 bits		

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.21—Charge amplifier template (incl. attached force transducer) (ID = 43) summary

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—		TEMPLATE	Template ID	—	8	Integer (value = 43)	—
Measurement	Select Case—Attached Transducer: Not specified (0) / Specified (1)					1	Select Case	—
	Case 0	Select Case—Extended Functionality: None (0) / Programmable gain (1)				1	Select Case	—
		Case 0	%Gain	Gain	CAL	12	ConRelRes (5E+7 to 17.8E+10, ±0.1%)	V/C
			%TF_HP_S	High-pass cut-off frequency (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_SP	Low pass cut-off frequency (F lp)	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
		Case 1	%Gain[CtrlFunctionMask]	Associate with gain	CAL	—	Assign = 0b11	—
			%Gain[Default]	Default gain 00: no 1: low, 2: high	CAL	2	UNINT	—
			%Gain[“01”]	Low gain	CAL	12	ConRelRes (5E+6 to 17.8E+9, ±0.1%)	V/C
			%Gain[“10”]	High gain	CAL	12	ConRelRes (5E+7 to 17.8E+10, ±0.1%)	V/C
			%TF_HP_S[CtrlFunctionMask]	Associate with gain	CAL	—	Assign = 0b11	—
			%TF_HP_S[“01”]	Low-gain high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_HP_S[“10”]	High-gain high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_SP	Low pass cut-off frequency (F lp)	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
			%Passive[CtrlFunctionMask]	Associate with gain	CAL	—	Assign = 0b11	—
			%Passive[00]	Supports multiplexed (passive) mode via no gain selected	CAL	1	UNINT	—

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.21—Charge amplifier template (incl. attached force transducer) (ID = 43) summary *(continued)*

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 1	—	%Attached_MfgrID	Manufacturer of attached transducer	CAL	14	UNINT	—
		—	%Attached_ModelNum	Model number of attached transducer	CAL	15	UNINT	—
		—	%Attached_VersionLetter	Version letter of attached transducer	CAL	5	CHR5	—
		—	%Attached_VersionNum	Version number of attached transducer	CAL	6	UNINT	—
		—	%Attached_SerialNum	Serial number of attached transducer	CAL	24	UNINT	—
		Select Case—Extended Functionality: None (0) / Programmable gain (1)				1	Select Case	—
		Case 0	%Sens@Ref	Total sensitivity @ reference condition	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/N
			%TF_HP_S	High-pass cut-off frequency (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_SP	Low pass cut-off frequency (F lp)	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
		Case1	%Sens@Ref[CtrlFunctionMask]	Associate with gain	CAL	—	Assign = 0b11	—
			% Sens@Ref [Default]	Default gain 00: no, 1: low, 2: high	CAL	2	UNINT	—
			% Sens@Ref [“01”]	Low gain total sensitivity	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/N
			% Sens@Ref [“10”]	High gain total sensitivity	CAL	16	ConRelRes (5E-7 to 172, ±0.015%)	V/N
			%TF_HP_S[CtrlFunctionMask]	Associate with gain	CAL	—	Assign = 0b11	—
			%TF_HP_S[“01”]	Low-gain high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_HP_S[“10”]	High-gain high-pass cut-off freq. (F hp)	CAL	8	ConRelRes (0.005 to 13k, ±3%)	Hz
			%TF_SP	Low pass cut-off frequency (F lp)	CAL	7	ConRelRes (10 to 1.6E+6, ±5%)	Hz
			%Passive[CtrlFunctionMask]	Associate with gain	CAL	—	Assign = 0b11	—
			%Passive[00]	Supports multiplexed (passive) mode via no gain selected	CAL	1	UNINT	—
		—	%Direction	Sensitivity direction (x, y, z)	CAL	2	Enumeration: x | y | z	—
		Select Case—Transfer Function: Not specified (0) / Specified (1)				1	Select Case	—
		Case 0	—	No transfer function specified	—	—		—

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

Table A.21—Charge amplifier template (incl. attached force transducer) (ID = 43) summary *(continued)*

Function	Select		Property/Command	Description	Access	Bits	Data type (and range)	Units
		Case 1	%TF_KPr	Resonance frequency (F res)	CAL	9	ConRelRes (100 to 2.4E+6, ±1%)	Hz
			%TF_KPq	Quality factor at F res	CAL	9	ConRelRes (0.4 to 9.7k, ±1%)	—
			%TF_SL	Amplitude slope (a)	CAL	7	ConRes (–6.3 to 6.3, step 0.1)	%/ decade
			%TempCoef	Temperature Coefficient (b)	CAL	6	ConRes (–0.8 to 0.75, step 0.025)	%/°C
			%Stiffness	Stiffness of transducer	CAL	6	ConRelRes (1E6 to 8.11E10, ±10%)	N/m
			%Mass_below	Mass below gage	CAL	6	ConRelRes (0.1 to 8114, ±10%)	gram
Electrical signal output	—	—	%Sign	Polarity (sign)	CAL	1	Enumeration: positive | negative	—
	—	—	%MapMeth	Mapping Method	ID	—	Assign = 0, “Linear”	—
	—	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 0, “Voltage Sensor”	—
	—	—	%ACDCCoupling	AC or DC coupling	ID	—	Assign = 1, “AC”	—
Calibration information	—	—	%PhaseCorrection	Phase Correction @ ref condition	CAL	6	ConRes (–3.2 to 3.0, step 0.1)	°
	—	—	%Reffreq	Reference frequency	CAL	8	ConRelRes (0.35 to 2.18k, ±1.75%)	Hz
	—	—	%RefTemp	Reference temperature (T ref)	CAL	5	ConRes (15 to 30, step 0,5)	°C
	—	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	—	%CalPeriod	Calibration period	CAL	12	UNINT	Days
Misc.	—	—	%MeasID	Measurement location ID	USR	11	UNINT	—
				Total bits required for TEDS (range):	111 to 252 bits			

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table A.22—Enumeration of Select Case values for "Physical Measurand" used in several templates

Case	Physical units
0	K
1	°C
2	strain
3	microstrain
4	N
5	lb
6	kgf
7	m/s^2
8	ga
9	Nm/radian
10	Nm
11	oz-in
12	Pa
13	psi
14	Kg
15	G

Case	Physical units
16	m
17	mm
18	in
19	m/s
20	mph
21	fps
22	radians
23	degrees
24	radian/s
25	rpm
26	Hz
27	g/l
28	kg/m^3
29	$mole/m^3$
30	mole/l
31	m^3/m^3

Case	Physical units
32	l/l
33	kg/s
34	m^3/s
35	m^3/hr
36	gpm
37	cfm
38	l/min
39	RH
40	%
41	V
42	V rms
43	Amperes
44	Amperes rms
45	Watts

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

A.3 IEEE ENUMERATE listings

The listing below contains the enumerations defined in the standard. If they are redefined in the templates, they shall be identical.

```
ENUMERATE ACDCCouplingEnum,"DC","AC"
ENUMERATE BridgeTypeEnum, "Quarter", "Half", "Full"
ENUMERATE CalSetEnum, "Electrical", "Physical"
ENUMERATE CJSourceEnum, "CJC not provided by sensor","Sensor compensated for 0°C cold junction"
ENUMERATE DirectionEnum,"x","y","z"
ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
ENUMERATE ExciteTypeEnum, "DC", "Bipolar DC", "AC (rms)"
ENUMERATE GageTypeEnum, "Single element", "Two Poisson elements", "Two elements opposite sign", "Two elements same
sign", "Two element chevron", "Four element Poisson same sign", "Four element Poisson opposite sign", "Four uniaxial
elements", "Four element dual chevron", "Tee Rosette grid 1 (0°)", "Tee Rosette grid 2 (90°)", "Delta Rosette grid 1
(0°)", "Delta Rosette grid 2 (60°)", "Delta Rosette grid 3 (120°)", "Rectangular Rosette grid 1 (0°)", "Rectangular
Rosette grid 2 (45°)", "Rectangular Rosette grid 3 (90°)"
ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
ENUMERATE MicSizeEnum, "1 inch", "1/2 inch", "1/4 inch", "1/8 inch"
ENUMERATE MicTypeEnum, "Free field", "Pressure", "Random incidence", "Other"
ENUMERATE PolarizationEnum, "Pre-polarized", "28 V", "200 V"
ENUMERATE Resp_TypeEnum, "Actuator", "Corrected"
ENUMERATE SignEnum,"Positive","Negative"
ENUMERATE TCTypeEnum, "B","E","J","K","N","R","S","T","non-standard"
ENUMERATE YesNoEnum, "No", "Yes"
```

A.4 IEEE PHYSICAL_UNIT listings

The listing below contains the PHYSICAL_UNITs defined in the standard. If they are redefined in the templates, they shall be consistent (conversion constants could, e.g., be defined with more digits).

```
PHYSICAL_UNIT "%",          (1, 0,0, 0,0, 0, 0,0,0,0,100,0)       // Dimensionless Ratio: percent
PHYSICAL_UNIT "%/°C",       (0,0,0,0,0,0,0,-1,0,0,0.01,0)         // Temperature drift: %/°C = 0.01 x (1/Kelvin)
PHYSICAL_UNIT "%/decade",   (6,0,0,0,0,-1,0,0,0,0,0.01,0)         // Flatness over frequency: 0.01 / (log10(Hz/Hz))
PHYSICAL_UNIT "°C",         (0, 0,0, 0,0, 0, 0,1,0,0,1,-273.15)   // Celsius is (kelvin - 273.15 K)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
PHYSICAL_UNIT "A rms",      (0, 0,0, 0,0, 0, 1,0,0,0,1,0)          // Current: base SI unit is Ampere
PHYSICAL_UNIT "A",          (0, 0,0, 0,0, 0, 1,0,0,0,1,0)          // Current: base SI unit is Ampere
PHYSICAL_UNIT "cfm",        (0, 0,0, 3,0,-1, 0,0,0,0,4.71947e-4,0)// cubic feet per minute = 4.71947e-4 m^3/s
PHYSICAL_UNIT "days",       (0, 0,0, 0,0,1, 0,0,0,0,86400,0)       // Time: Days = 86400 seconds
PHYSICAL_UNIT "dB",         (3, 0,0, 2,1,-3,-1,0,0,0,0.05,0)       // A decibel of voltage gain is 0.05 x (log10(V/V))
PHYSICAL_UNIT "degrees",    (0,1,0, 0,0,0, 0,0,0,0,0.0174533,0)    // degrees = 0.0174533 radians
PHYSICAL_UNIT "F",          (0, 0,0,-2,-1,2, 0,0,0,0,1,0)          // Farad equals sec^2/(m^2*kg)
PHYSICAL_UNIT "fps",        (0, 0,0, 1,0,-1, 0,0,0,0,0.3048,0)     // feet per second = 0.3048 m/s
PHYSICAL_UNIT "g",          (0, 0,0, 0,1, 0, 0,0,0,0,0.001,0)      // gram = 0.001 kg
PHYSICAL_UNIT "g/l",        (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // gram per liter equals kg/m^3
PHYSICAL_UNIT "ga",         (0, 0,0, 1,0,-2, 0,0,0,0,9.80665,0)    // ga = 9.80665 m/s^2
PHYSICAL_UNIT "gpm",        (0, 0,0, 3,0,-1, 0,0,0,0,6.30902e-5,0)// gallons(US) per minute = 6.30902e-5 m^3/s
PHYSICAL_UNIT "Hz",         (0, 0,0, 0,0,-1, 0,0,0,0,1,0)          // Frequency: Hertz = 1/second
PHYSICAL_UNIT "in",         (0, 0,0, 1,0, 0, 0,0,0,0,0.0254,0)     // inches = 0.0254 m
PHYSICAL_UNIT "K",          (0, 0,0, 0,0, 0, 0,1,0,0,1,0)          // Temperature: base SI unit is kelvin
PHYSICAL_UNIT "kg",         (0, 0,0, 0,1, 0, 0,0,0,0,1,0)          // Mass:  base SI unit is kilogram
PHYSICAL_UNIT "kg/m^3",     (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // Density: kg/m^3
PHYSICAL_UNIT "kg/s",       (0, 0,0, 0,1,-1, 0,0,0,0,1,0)          // Mass Flow:  preferred SI unit is kg/s
PHYSICAL_UNIT "kgf",        (0, 0,0, 1,1,-2, 0,0,0,0,9.80665,0)    // kilogram-force (kilopond) is 9.80665 x Newton
PHYSICAL_UNIT "l/l",        (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // liter per liter equals m^3/m^3
PHYSICAL_UNIT "l/min",      (0, 0,0, 3,0,-1, 0,0,0,0,1.66667e-5,0)// liters per minute = 1.66667e-5 m^3/s
PHYSICAL_UNIT "lb",         (0, 0,0, 1,1,-2, 0,0,0,0,4.44822,0)    // pound is 4.44822 x Newton
PHYSICAL_UNIT "m",          (0, 0,0, 1,0, 0, 0,0,0,0,1,0)          // Distance:  base SI unit is meter
PHYSICAL_UNIT "m/s",        (0, 0,0, 1,0,-1, 0,0,0,0,1,0)          // Velocity: Preferred SI unit is meter/second
PHYSICAL_UNIT "m/s^2",      (0, 0,0, 1,0,-2, 0,0,0,0,1,0)          // Acceleration: m/s^2
PHYSICAL_UNIT "m^3",        (0,0,0,3,0,0,0,0,0,0,1,0)              // Volume equals m^3
PHYSICAL_UNIT "m^3/hr",     (0, 0,0, 3,0,-1, 0,0,0,0,3600,0)       // m^3 per hour = 3600 m^3/s
PHYSICAL_UNIT "m^3/m^3",    (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // Volumetric concentration: base SI unit is m^3/m^3
PHYSICAL_UNIT "m^3/s",      (0, 0,0, 3,0,-1, 0,0,0,0,1,0)          // Volumetric Flow: preferred SI unit is m^3/s
PHYSICAL_UNIT "mA",         (0, 0,0, 0,0, 0, 1,0,0,0,0.001,0)      // milliAmpere = 0.001 x Ampere
PHYSICAL_UNIT "microstrain",(1, 0,0, 1,0, 0, 0,0,0,0,1e-6,0)       // microstrain = 10^(-6) x strain
PHYSICAL_UNIT "mm",         (0, 0,0, 1,0, 0, 0,0,0,0,0.001,0)      // millimeter = 0.001 m
PHYSICAL_UNIT "mm^2",       (0, 0,0, 2,0, 0, 0,0,0,0,1E-6,0)       // square millimeter = 1E-6 square meters
PHYSICAL_UNIT "mole/l",     (0, 0,0,-3,0, 0, 0,0,1,0,1000,0)       // mole per liter = 1000 mole/m^3
PHYSICAL_UNIT "mole/m^3",   (0, 0,0,-3,0, 0, 0,0,1,0,1,0)          // Molar concentration: preferred SI unit is mole/m^3
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

```
PHYSICAL_UNIT "mph",          (0, 0,0, 1,0,-1, 0,0,0,0,0.44704,0)   // miles per hour = 0.44704 m/s
PHYSICAL_UNIT "N",            (0, 0,0, 1,1,-2, 0,0,0,0,1,0)         // Force: Newton equals m•kg/s^2
PHYSICAL_UNIT "N/m",          (0,0,0,0,1,-2, 0,0,0,0,1,0)           // Newton/m equals (m•kg/s^2)/m = kg/s^2
PHYSICAL_UNIT "Nm",           (0,-1,0, 2,1,-2, 0,0,0,0,1,0)         // Common usage drops radian from abbreviation
PHYSICAL_UNIT "Nm/radian",    (0,-1,0, 2,1,-2, 0,0,0,0,1,0)         // Torque: N•m/rad equals m^2•kg/(s^2•rad)
PHYSICAL_UNIT "Ohm",          (0, 0,0, 2,1,-3,-2,0,0,0,1,0)         // Resistance: Ohms equals m^2•kg/(s^3•A^2)
PHYSICAL_UNIT "oz-in",        (0,-1,0, 2,1,-2, 0,0,0,0,0.00706155,0)// oz-in is 0.00706155•Nm
PHYSICAL_UNIT "Pa",           (0, 0,0,-1,1,-2, 0,0,0,0,1,0)         // Pressure: Pascal equals Newton/m^2 = kg/(m•s^2)
PHYSICAL_UNIT "PSI",          (0, 0,0,-1,1,-2, 0,0,0,0,6894.757,0)  // Pounds per Square Inch = 6894.757 Pa
PHYSICAL_UNIT "radian",       (0, 1,0, 0,0, 0, 0,0,0,0,1,0)         // Angular position: base SI unit is radian
PHYSICAL_UNIT "radian/s",     (0, 1,0, 0,0,-1, 0,0,0,0,1,0)         // Angular velocity: preferred SI unit is radian/s
PHYSICAL_UNIT "RH",           (1, 0,0,-3,1, 0, 0,0,0,0,1,0)         // Relative Humidity: = (kg/m^3)/(kg/m^3)
PHYSICAL_UNIT "rpm",          (0, 1,0, 0,0,-1, 0,0,0,0,0.104720,0)  // rev. per minute = 0.104720 radian/s
PHYSICAL_UNIT "sec",     (0, 0,0, 0,0, 1, 0,0,0,0,1,0)          // Time: base SI unit is seconds
PHYSICAL_UNIT "strain",       (1, 0,0, 1,0, 0, 0,0,0,0,1,0)         // Strain: equals meter/meter
PHYSICAL_UNIT "V/(m/s^2)",    (0,0,0,1,1,-1,-1,0,0,0,1,0)           // Volts per Acceleration: V/(m/s^2) = m*kg/(s*A)
PHYSICAL_UNIT "V/C",          (0,0,0,2,1,-2,0,0,0,0,1,0)            // Volt/Columb equals m^2*kg/(sec^2)
PHYSICAL_UNIT "V/N",          (0,0,0,1,0,-1,-1,0,0,0,1,0)           // Volts per Newton: V/(m•kg/s^2) = m/(s*A)
PHYSICAL_UNIT "V/Pa",         (0,0,0,3,0,-1,-1,0,0,0,1,0)           // Volts/Pascal equals m^3/(sec*A)
PHYSICAL_UNIT "V/V",          (1, 0,0, 2,1,-3,-1,0,0,0,1,0)         // Voltage ratio: [m^2•kg/(sec^3•A)]/[m^2•kg/(sec^3•A)]
PHYSICAL_UNIT "V rms",        (0, 0,0, 2,1,-3,-1,0,0,0,1,0)         // Voltage RMS: Volts equals m^3•kg/(sec^3•A)
PHYSICAL_UNIT "V",            (0, 0,0, 2,1,-3,-1,0,0,0,1,0)         // Voltage: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "W/°C",         (0,0,0,2,1,-3,0,-1,0,0,1,0)           // Thermal Conductivity: W/°C equals m^2•kg/(s^3•K)
PHYSICAL_UNIT "W",            (0, 0,0, 2,1,-3, 0,0,0,0,1,0)         // Watts equals m^2•kg/(sec^3)
```

A.5 IEEE standard template listings

```
TEMPLATE 0,8,25,"Accelerometer and Force Transducer"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 25 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Accelerometer and Force Transducer Template
SPACING
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Amp, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "V/(m/s^2)",    (0,0,0,1,1,-1,-1,0,0,0,1,0)          // Volts per Acceleration: V/(m/s^2) = m*kg/(s*A)
PHYSICAL_UNIT "V/N",          (0,0,0,1,0,-1,-1,0,0,0,1,0)          // Volts per Newton: V/(m•kg/s^2) = m/(s*A)
PHYSICAL_UNIT "Hz",           (0,0,0,0,0,-1,0,0,0,0,1,0)           // Frequency: Hertz = 1/second
PHYSICAL_UNIT "degrees",      (0,1,0,0,0,0,0,0,0,0,0.0174533,0)    // degrees = 0.0174533 radians
PHYSICAL_UNIT "°C",           (0,0,0,0,0,0,0,1,0,0,1,-273.15)      // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "%/decade",     (6,0,0,0,0,0,0,0,0,0,0.01,0)         // Flatness over frequency: 0.01 / (log10(Hz/Hz))
PHYSICAL_UNIT "%/°C",         (0,0,0,0,0,0,0,-1,0,0,0.01,0)        // Temperature drift: %/°C = 0.01 x (1/Kelvin)
PHYSICAL_UNIT "N/m",          (0,0,0,0,1,-2, 0,0,0,0,1,0)          // Newton/m equals (m•kg/s^2)/m = kg/s^2
PHYSICAL_UNIT "g",            (0, 0,0, 0,1, 0, 0,0,0,0,0.001,0)    // gram = 0.001 kg
PHYSICAL_UNIT "days",         (0,0,0,0,0,1,0,0,0,0,86400,0)        // Time: Days = 86400 seconds

SELECTCASE "Transducer Type", ID, 1

  CASE "Accelerometer", 0
    SELECTCASE "Extended Functionality", ID, 1
      CASE "None", 0
         UGID "I25-0-0-0", "Accelerometer"
        %Sens@Ref, "Sensitivity @ reference condition", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/(m/s^2)"
        %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
      ENDCASE
      CASE "Programmable sensitivity", 1
         UGID "I25-0-1-0", "Accelerometer, programmable sensitivity"
        %passive[Initialize], "Initialize not needed", ID, 1, UNINT, "", "" = 0
        %passive[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
        %passive[ReadWrite], "Write only", ID, 2, UNINT, "", "" = 3
        %passive[FunctionType], "Passive control type", ID, 2, UNINT, "","" = 0//checkmark
        %passive[Function], "Passive mode", USR, 10, BitBin, "","" =  "xx,00"

        %sens[Initialize], "Initialize not needed", ID, 1, UNINT, "", "" = 0
        %sens[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
        %sens[ReadWrite], "Write only", ID, 2, UNINT, "", "" = 3
        %sens[FunctionType], "Sensitivity control type", ID, 2, UNINT, "", "" = 1//One Exactly
        %sens[Function], %Sens@Ref["10"], USR, 4, BitBin, "", "" = "10" //High sensitivity
        %sens[Function], %Sens@Ref["01"], USR, 4, BitBin, "", "" = "01" //Low sensitivity
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
      %defaultFR, "Default setting", ID, 2, UNINT, "", ""
      %Passive, "Supports multiplexer mode", ID, 1, UNINT, "", ""
      %Sens@Ref["01"], "Low sensitivity @ Fref", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/(m/s^2)"
      %Sens@Ref["10"], "High sensitivity @ Fref", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/(m/s^2)"
      %TF_HP_S["01"], "Low sensitivity high pass cut-off frequency", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
      %TF_HP_S["10"], "High sensitivity high pass cut-off frequency", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
    ENDCASE
  ENDSELECT
ENDCASE

CASE "Force Transducer", 1
  SELECTCASE "Extended Functionality", ID, 1
    CASE "None", 0
     UGID "I25-1-0-0", "Force Transducer"
      %Sens@Ref, "Sensitivity @ reference condition", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/N"
      %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
      %Stiffness, "Stiffness of transducer", CAL, 6, ConRelRes, 1E6, 0.10, "rp", "N/m"
      %Mass_below, "Mass below gage", CAL, 6, ConRelRes, 0.1, 0.1, "rp", "g"
    ENDCASE
    CASE "Programmable sensitivity", 1
     UGID "I25-1-1-0", "Force Transducer, programmable sensitivity"
      %passive[Initialize], "Initialize not needed", ID, 1, UNINT, "", "" = 0
      %passive[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
      %passive[ReadWrite], "Write only", ID, 2, UNINT, "", "" = 3
      %passive[FunctionType], "Passive control type", ID, 2, UNINT, "","" = 0//checkmark
      %passive[Function], "Passive mode", USR, 10, BitBin, "","" =  "xx,00"

      %sens[Initialize], "Initialize not needed", ID, 1, UNINT, "", "" = 0
      %sens[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
      %sens[ReadWrite], "Write only", ID, 2, UNINT, "", "" = 3
      %sens[FunctionType], "Sensitivity control type", ID, 2, UNINT, "", "" = 1//One Exactly
      %sens[Function], %Sens@Ref["10"], USR, 4, BitBin, "", "" = "10" //High sensitivity
      %sens[Function], %Sens@Ref["01"], USR, 4, BitBin, "", "" = "01" //Low sensitivity

      %defaultFR, "Default setting", ID, 2, UNINT, "",""
      %Passive, "Supports multiplexer mode", ID, 1, UNINT, "", ""
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
        %Sens@Ref["01"], "Low sensitivity @ Fref", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/N"
        %Sens@Ref["10"], "High sensitivity @ Fref", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/N"
        %TF_HP_S["01"], "Low sensitivity high pass cut-off frequency", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
        %TF_HP_S["10"], "High sensitivity high pass cut-off frequency", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
        %Stiffness, "Stiffness of transducer", CAL, 6, ConRelRes, 1E6, 0.10, "rp", "N/m"
        %Mass_below, "Mass below gage", CAL, 6, ConRelRes, 0.1, 0.1, "rp", "g"
        %PhaseCorrection, "Phase correction @ reference condition", CAL, 6, CONRES, -3.2, 0.1, "rp", "degrees"
      ENDCASE
    ENDSELECT
  ENDCASE

ENDSELECT

ENUMERATE DirectionEnum,"x","y","z"
%Direction, "Sensitivity direction (x,y,z)",  CAL, 2, DirectionEnum, "e", ""

%Weight, "Transducer weight", CAL, 6, CONRELRES, 0.1, 0.1, "rp", "g"

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Voltage Sensor"

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

ENUMERATE ACDCCouplingEnum,"DC","AC"
%ACDCCoupling, "AC or DC Coupling", ID, 0, ACDCCouplingEnum,"e","" = "AC"

ENUMERATE SignEnum,"Positive","Negative"
%Sign, "Polarity (Sign)", CAL, 1, SignEnum, "e", ""

SELECTCASE "Transfer Function", ID, 1
  CASE "Not specified", 0
  ENDCASE
  CASE "Specified", 1
    %TF_SP, "Low pass cut-off frequency (F lp)", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
    %TF_KPr, "Resonance frequency (F res)", CAL, 9, ConRelRes, 100, 0.01, "rp", "Hz"
    %TF_KPq, "Quality factor @ F res (Q)", CAL, 9, ConRelRes, 0.4, 0.01,  "rp",""
    %TF_SL, "Amplitude slope (a)", CAL, 7, ConRes, -6.3, 0.1, "0.0", "%/decade"
    %TempCoef, "Temperature coefficient (b)", CAL, 6, ConRes, -0.8, 0.025,"0.000","%/°C"
  ENDCASE
ENDSELECT

%Reffreq, "Reference frequency (F ref)", CAL, 8, ConRelRes, 0.35, 0.0175, "0p","Hz"
%RefTemp, "Reference temperature (T ref)", CAL, 5, ConRes, 15, 0.5, "0.0","°C"

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT,"0","",""

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------

TEMPLATE 0,8,26,"Charge Amplifier (with Attached Accelerometer)"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 26 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Charge Amplifier (with Attached Accelerometer) Template
ABSTRACT For charge amplifier devices, with attached piezoelectric accelerometer
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Amp, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "Hz",          (0,0,0,0,0,-1,0,0,0,0,1,0)          // Frequency: Hertz = 1/second
PHYSICAL_UNIT "V/C",         (0,0,0,2,1,-2,0,0,0,0,1,0)          //Volt/Columb equals m^2*kg/(sec^2)
PHYSICAL_UNIT "V/(m/s^2)",   (0,0,0,1,1,-1,-1,0,0,0,1,0)         // Volts per Acceleration: V/(m/s^2) = m*kg/(s*A)
PHYSICAL_UNIT "Hz",          (0,0,0,0,0,-1,0,0,0,0,1,0)          // Frequency: Hertz = 1/second
PHYSICAL_UNIT "degrees",     (0,1,0,0,0,0,0,0,0,0,0.0174533,0)   // degrees = 0.0174533 radians
PHYSICAL_UNIT "°C",          (0,0,0,0,0,0,0,1,0,0,1,-273.15)     // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "%/decade",    (6,0,0,0,0,-1,0,0,0,0,0.01,0)       // Flatness over frequency: 0.01 / (log10(Hz/Hz))
PHYSICAL_UNIT "%/°C",        (0,0,0,0,0,0,0,-1,0,0,0.01,0)       // Temperature drift: %/°C = 0.01 x (1/Kelvin)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
PHYSICAL_UNIT "V",          (0,0,0, 2,1,-3,-1,0,0,0,1,0)          // Voltage: Volts equals m²•kg/(sec³•A)
PHYSICAL_UNIT "days",       (0,0,0,0,0,1,0,0,0,0,86400,0)         // Time: Days = 86400 seconds

SELECTCASE "Attached Transducer", ID, 1

  CASE "Not specified", 0
    SELECTCASE "Extended Functionality", ID, 1
      CASE "None", 0
         UGID "I26-0-0-0", "Charge Amplifier"
        %Gain, "Gain", CAL, 13, ConRelRes, 1E7, 0.001,  "rp", "V/C"
        %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
        %TF_SP, "Low pass cut-off frequency (F lp)", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"
      ENDCASE
      CASE "Programmable Gain", 1
         UGID "I26-0-1-0", "Charge Amplifier, Programmable"
        %passive[Initialize], "Initialize not needed", ID, 1, UNINT, "", "" = 0
        %passive[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
        %passive[ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
        %passive[FunctionType], "Passive control type", ID, 2, UNINT, "","" = 0//checkmark
        %passive[Function], "Passive mode", USR, 10, BitBin, "","" =  "xx,00"

        %gain[Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
        %gain[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
        %gain[ReadWrite], "Write only", ID, 2, UNINT, "", "" = 3
        %gain[FunctionType], "Gain control type", ID, 2, UNINT, "", "" = 1//One Exactly
        %gain[Function], %gain["10"], USR, 4, BitBin, "","" = "10" //High gain
        %gain[Function], %gain["01"], USR, 4, BitBin, "", "" = "01" //Low gain

        %defaultFR, "Default setting", ID, 2, UNINT, "",""
        %Passive, "Supports multiplexer mode", ID, 1, UNINT, "", ""

        %gain["01"], "Low gain", CAL, 13, ConRelRes, 1E7, 0.001, "rp", "V/C"
        %gain["10"], "High gain", CAL, 13, ConRelRes, 1E7, 0.001, "rp", "V/C"
        %TF_HP_S["01"], "Low-gain high pass cut-off frequency",  CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
        %TF_HP_S["10"], "High-gain high pass cut-off frequency",  CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
        %TF_SP, "Low pass cut-off frequency", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"
      ENDCASE
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  ENDSELECT
ENDCASE

CASE "Accelerometer", 1
  %Attached_MfgrID, "Manufacturer of attached transducer", CAL, 14, UNINT, "", ""
  %Attached_ModelNum, "Model number of attached transducer", CAL, 15, UNINT, "", ""
  %Attached_VersionLetter, "Version letter of attached transducer", CAL, 5, CHR5, "s", ""
  %Attached_VersionNum, "Version number of attached transducer", CAL, 6, UNINT, "", ""
  %Attached_SerialNum, "Serial number of attached transducer", CAL, 24, UNINT, "", ""
  SELECTCASE "Extended Functionality", ID, 1
    CASE "None", 0
      UGID "I26-0-0-0", "Charge Amplifier"
       %Sens@Ref, "Total sensitivity @ ref. condition", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/(m/s^2)"
       %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
       %TF_SP, "Low pass cut-off frequency (F lp)", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"
    ENDCASE
    CASE "Programmable sensitivity", 1
      UGID "I26-0-1-0", "Charge Amplifier, Programmable"
       %passive[Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
       %passive[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
       %passive[ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
       %passive[FunctionType], "Passive control type", ID, 2, UNINT, "","" = 0//checkmark
       %passive[Function], "Passive mode", USR, 10, BitBin, "", "" = "xx,00"

       %sens[Initialize], "Initialize not needed", ID, 1, UNINT, "", "" = 0
       %sens[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "","" = "11"
       %sens[ReadWrite], "Write only", ID, 2, UNINT, "", "" = 3
       %sens[FunctionType], "Sensitivity control type", ID, 2, UNINT, "","" = 1//One Exactly
       %sens[Function], %Sens@Ref["10"], USR, 4, BitBin, "","" = "10" //High sensitivity
       %sens[Function], %Sens@Ref["01"], USR, 4, BitBin, "","" = "01" //Low sensitivity

       %defaultFR, "Default setting", ID, 2, UNINT, "",""
       %Passive, "Supports multiplexer mode", ID, 1, UNINT, "", ""


       %Sens@Ref["01"], "Low sensitivity @ Fref (total)", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/(m/s^2)"
       %Sens@Ref["10"], "High sensitivity @ Fref (total)", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/(m/s^2)"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
    %TF_HP_S["01"], "Low sensitivity high pass cut-off frequency", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
    %TF_HP_S["10"], "High sensitivity high pass cut-off frequency", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
    %TF_SP, "Low pass cut-off frequency", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"
  ENDCASE
ENDSELECT

ENUMERATE DirectionEnum,"x","y","z"
%Direction, "Sensitivity direction (x,y,z)", CAL, 2, DirectionEnum, "e", ""

SELECTCASE "Transfer Function", ID, 1
  CASE "Not Specified", 0
  ENDCASE
  CASE "Specified", 1
    %TF_KPr, "Resonance frequency (F res)", CAL, 9, ConRelRes, 100, 0.01, "rp", "Hz"
    %TF_KPq, "Quality factor @ F res (Q)", CAL, 9, ConRelRes, 0.4, 0.01, "rp",""
    %TF_SL, "Amplitude slope (a)", CAL, 7, ConRes, 16.3, 0.1, "0.0", "%/decade"
    %TempCoef, "Temperature coefficient (b)", CAL, 6, ConRes, -0.8, 0.025,"0.000","%/°C"
  ENDCASE
ENDSELECT
ENDCASE

ENDSELECT

ENUMERATE SignEnum,"Positive","Negative"
%Sign, "Polarity (Sign)", CAL, 1, SignEnum, "e", ""

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Voltage Sensor"

ENUMERATE ACDCCouplingEnum,"DC","AC"
%ACDCCoupling, "AC or DC Coupling", ID, 0, ACDCCouplingEnum,"e","" = "AC"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
%PhaseCorrection, "Phase correction @ F ref", CAL, 6, CONRES, -3.2, 0.1, "rp", "degrees"
%Reffreq, "Reference frequency (F ref)", CAL, 8, ConRelRes, 0.35, 0.0175, "0p","Hz"
%RefTemp, "Reference temperature (T ref)", CAL, 5, ConRes, 15, 0.5, "0.0","°C"

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID",  USR, 11, UNINT,"0", ""

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------


TEMPLATE 0,8,27,"Microphones with Built-in Preamplifier"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 27 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Microphone Template
ABSTRACT For microphone transducers with built-in preamplifiers
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Amp, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "V/Pa",        (0,0,0,3,0,-1,-1,0,0,0,1,0)          // Volts/Pascal equals m3/(sec*A)
PHYSICAL_UNIT "Hz",          (0,0,0,0,0,-1,0,0,0,0,1,0)           // Frequency: Hertz = 1/second
PHYSICAL_UNIT "dB",          (3,0,0,2,1,-3,-1,0,0,0,0.05,0)       // A decibel of voltage gain is 0.05 x (log10(V/V))
PHYSICAL_UNIT "m3",          (0,0,0,3,0,0,0,0,0,0,1,0)            // Volume equals m3
PHYSICAL_UNIT "%/decade",    (6,0,0,0,0,-1,0,0,0,0,0.01,0)        // Flatness over frequency: 0.01 / (log10(Hz/Hz))
PHYSICAL_UNIT "V",           (0,0,0,2,1,-3,-1,0,0,0,1,0)          // Voltage: Volts equals m2•kg/(sec3•A)
PHYSICAL_UNIT "days",        (0,0,0,0,0,1,0,0,0,0,86400,0)        // Time: Days = 86400 seconds

SELECTCASE "Extended Functionality", ID, 1
  CASE "None", 0
    UGID "I27-0-0-0", "Microphones with Built-in Preamplifier"
    %Sens@Ref, "Sensitivity @ reference condition", CAL, 16, ConRelRes, 10E-6, 0.0001, "rp", "V/Pa"
  ENDCASE
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
  CASE "Programmable sensitivity", 1
    UGID "I27-1-0-0", "Microphones with Built-in Preamplifier, Programmable sensitivity"
    %passive[Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
    %passive[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
    %passive[ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
    %passive[FunctionType], "Passive control type", ID, 2, UNINT, "","" = 0//checkmark
    %passive[Function], "Passive mode", USR, 10, BitBin, "", "" =  "xx,00"

    %sens[Initialize], "Initialize not needed", ID, 1, UNINT, "", "" = 0
    %sens[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "","" = "11"
    %sens[ReadWrite], "Write only", ID, 2, UNINT, "", "" = 3
    %sens[FunctionType], "Sensitivity control type", ID, 2, UNINT, "","" = 1//One Exactly
    %sens[Function], %Sens@Ref["10"], USR, 4, BitBin, "","" = "10" //High sensitivity
    %sens[Function], %Sens@Ref["01"], USR, 4, BitBin, "","" = "01" //Low sensitivity

    %defaultFR, "Default setting", ID, 2, UNINT, "",""
    %Passive, "Supports multiplexer mode", ID, 1, UNINT, "", ""
    %Sens@Ref["01"], "Low Sensitivity @ F ref", CAL, 16, ConRelRes, 10E-6, 0.0001, "rp", "V/Pa"
    %Sens@Ref["10"], "High Sensitivity @ F ref", CAL, 16, ConRelRes, 10E-6, 0.0001, "rp", "V/Pa"
  ENDCASE
ENDSELECT

%Reffreq, "Reference frequency (F ref)", CAL, 8, ConRelRes, 0.35, 0.0175,"rp","Hz"

ENUMERATE PolarizationEnum, "Pre-polarized", "28 V", "200 V"
%RefPol, "Polarization Voltage", CAL, 2, PolarizationEnum, "e", ""

SELECTCASE "System Test (Test Gain)", ID, 1
  CASE "Not available", 0
  ENDCASE
  CASE "Available", 1
    %TestGain, "Test gain", CAL, 10, ConRes, 0.0, 0.1, "0.0", "dB"
  ENDCASE
ENDSELECT

ENUMERATE MicTypeEnum, "Free field", "Pressure", "Random incidence", "Other"
%MicType, "Microphone Type", CAL, 2, MicTypeEnum, "e", ""
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
ENUMERATE MicSizeEnum, "1 inch", "1/2 inch", "1/4 inch", "1/8 inch"
%MicSize, "Microphone Size", CAL, 2, MicSizeEnum, "e", ""

%Equi_Vol, "Equivalent microphone volume", CAL, 8, ConRes, 0.0, 1E-9, "0.0p", " m3"

SELECTCASE "Transfer Function", ID, 1
  CASE "Not Specified", 0
  ENDCASE
  CASE "Specified", 1
    ENUMERATE Resp_TypeEnum, "Actuator", "Corrected"
    %Resp_Type, "Actuator or corrected response", CAL, 1, Resp_TypeEnum, "e", ""
    %TF_HP_S, "Lowest high pass cut-off frequency (F hp)", CAL, 7, ConRelRes, 0.005, 0.05, "rp","Hz"
    %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.05, 0.01, "rp","Hz"
    %TF_SP, "F low lift (Low pass cut-off frequency)", CAL, 7, ConRelRes, 5, 0.02, "rp", "Hz"
    %TF_SZm, "F high/ F low lift", CAL, 8, ConRelRes, 1, 0.0015, "rp", ""
    %TF_KPr, "Resonance frequency (F res)", CAL, 8, ConRelRes, 2000, 0.01, "rp", "Hz"
    %TF_KPq, "Quality factor @ F res (Q)", CAL, 8, ConRelRes, 0.2, 0.01,  "rp",""
    %TF_KPr, "Second resonance frequency (F res)", CAL, 6, ConRelRes, 10000, 0.03, "rp", "Hz"
    %TF_KPq, "Quality factor @ second F res (Q)", CAL, 7, ConRelRes, 0.2, 0.03,  "rp",""
  ENDCASE
ENDSELECT

ENUMERATE SignEnum,"Positive","Negative"
%Sign, "Polarity (Sign)", CAL, 1, SignEnum, "e", ""

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Voltage Sensor"

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

ENUMERATE ACDCCouplingEnum,"DC","AC"
%ACDCCoupling, "AC or DC Coupling", ID, 0, ACDCCouplingEnum,"e","" = "AC"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT,"0", ""

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------

TEMPLATE 0,8,28,"Microphone Preamplifier & Microphone"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 28 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Microphone Preamplifier Template
ABSTRACT For microphone preamplifiers, with options for attached microphone

SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Amp, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "Hz",          (0,0,0,0,0,-1,0,0,0,0,1,0)          // Frequency: Hertz = 1/second
PHYSICAL_UNIT "dB",          (3,0,0,2,1,-3,-1,0,0,0,0.05,0)      // A decibel of voltage gain is 0.05 x (log10(V/V))
PHYSICAL_UNIT "V/Pa",        (0,0,0,3,0,-1,-1,0,0,0,1,0)         // Volts/Pascal equals m^3/(sec*A)
PHYSICAL_UNIT "Ohm",         (0,0,0,2,1,-3,-2,0,0,0,1,0)         // Resistance: Ohms equals m^2•kg/(s^3•A^2)
PHYSICAL_UNIT "F",           (0, 0,0,-2,-1,2,0,0,0,0,1,0)        // Farad equals sec^2/(m^2*kg)
PHYSICAL_UNIT "%/decade",    (6,0,0,0,0,-1,0,0,0,0,0.01,0)       // Flatness over frequency: 0.01 / (log10(Hz/Hz))
PHYSICAL_UNIT "V",           (0,0,0, 2,1,-3,-1,0,0,0,1,0)        // Voltage: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "days",        (0,0,0,0,0,1,0,0,0,0,86400,0)       // Time: Days = 86400 seconds


SELECTCASE "TEDS selection", ID, 2

  CASE "Preamplifier with optional Appended TEDS for Attached Microphone", 0
            %Rin, "Amplifier resistance", CAL, 6, ConRelRes, 1E8, 0.1,  "rp", "Ohm"
            %Cin, "Amplifier capacitance", CAL, 6, ConRes, 0, 0.02E-12, "0.000p", "F"
            %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
   %TF_SP, "F low lift (low pass cutoff frequency)", CAL, 7, ConRelRes, 5, 0.02, "rp", "Hz"
   %Rout, "Output impedance", CAL, 5, ConRelRes, 5, 0.10, "0.0p", "Ohm"
ENUMERATE YesNoEnum, "No","Yes"
%Gate,    "Gate present", ID, 1, YesNoEnum, "e",""
SELECTCASE "Extended Functionality", ID, 1
   CASE "None", 0
       UGID "I28-0-0-0", "Preamp. w. optional Attached Mic. "
       %Gain, "Gain", CAL, 12, ConRes, -40, 0.025,  "0.000", "dB"
   ENDCASE
   CASE "Programmable Gain", 1
       UGID "I28-0-1-0", "Preamp. w. optional Attached Mic., Programmable"
       %passive[Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
       %passive[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
       %passive[ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
       %passive[FunctionType], "Passive control type", ID, 2, UNINT, "","" = 0//checkmark
       %passive[Function], "Passive mode", USR, 10, BitBin, "", "" =  "xx,00"
       %gain[Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
       %gain[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
       %gain[ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
       %gain[FunctionType], "Gain control type", ID, 2, UNINT, "", "" = 1//One Exactly
       %gain[Function], %gain["10"], USR, 4, BitBin, "", "" = "10" //High gain
       %gain[Function], %gain["01"], USR, 4, BitBin, "", "" = "01" //Low gain
       %defaultFR, "Default setting", ID, 2, UNINT, "",""
       %Passive, "Supports multiplexer mode", ID, 1, UNINT, "", ""
       %gain["01"], "Low gain", CAL, 12, ConRes, -40, 0.025, "0.000", "dB"
       %gain["10"], "High gain", CAL, 12, ConRes, -40, 0.025, "0.000", "dB"
   ENDCASE
ENDSELECT
SELECTCASE "Appended Microphone TEDS", ID, 1
   CASE "Not specified", 0
   ENDCASE
   CASE "Specified", 1
       %Attached_MfgrID, "Manufacturer of attached microphone", CAL, 14, UNINT, "", ""
       %Attached_ModelNum, "Model number of attached microphone", CAL, 15, UNINT, "", ""
       %Attached_VersionLetter, "Version letter of attached microphone", CAL, 5, CHR5, "s", ""
       %Attached_VersionNum, "Version number of attached microphone", CAL, 6, UNINT, "", ""
       %Attached_SerialNum, "Serial number of attached microphone", CAL, 24, UNINT, "", ""
       %Appended_TEDS_location, "Appended TEDS location", USR, 5, String5, "", ""
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
    ENDCASE
  ENDSELECT

ENDCASE

CASE "Combined TEDS for Preamplifier with Attached Microphone", 1
    SELECTCASE "Extended Functionality", ID, 1
      CASE "None", 0
        UGID "I28-0-0-0", "Preamp. w. optional Attached Mic. "
          %Sens@Ref, "Sensitivity @ reference condition", CAL, 16, ConRelRes, 10E-6, 0.0001, "rp", "V/Pa"
      ENDCASE
      CASE "Programmable sensitivity", ID, 2
        UGID "I28-0-1-0", "Preamp. w. optional Attached Mic., Programmable"
          %passive[Initialize], "Initialize hot needed", ID, 1, UINT, "","" = 0
          %passive[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "","" = "11"
          %passive[ReadWrite], "Write only", ID, 2, UINT, "", "" = 3
          %passive[FunctionType], "Passive control type", ID, 2, UINT, "","" = 0//checkmark
          %passive[Function], "Passive mode", USR, 0, BitBin, "", "" = "xx,00"
          %sens[Initialize], "Initialize not needed",ID, 1, UINT, "","" = 0
          %sens[CtrlFunctionMask], "Control Function Mask", ID, 4, BitBin, "","" = "11"
          %sens[ReadWrite], "Write only", ID, 2, UINT, "","" = 3
          %sens[FunctionType], "Sensitivity control type", ID, 2, UINT, "", "" = 1//One Exactly
          %sens[Function], %Sens@Ref["10"], USR, 4, BitBin, "", "" = "10" //High sensitivity
          %sens[Function], %Sens@Ref["01"], USR, 4, BitBin, "", "" = "01" //Low sensitivity
          %defaultFR, "Default setting", ID, 2, UINT, "","",""
          %Passive, "Supports multiplexer mode", ID, 1, UINT, "", ""
          %Sens@Ref["01"], "Low Sensitivity @ F ref", CAL, 16, ConRelRes, 10E-6, 0.0001, "rp", "V/Pa"
          %Sens@Ref["10"], "High Sensitivity @ F ref", CAL, 16, ConRelRes, 10E-6, 0.0001, "rp", "V/Pa"
      ENDCASE
    ENDSELECT
    SELECTCASE "Additional Basic TEDS", ID, 1
      CASE "Microphone", 0
        %Attached_MfgrID, "Manufacturer of attached microphone", CAL, 14, UNINT,"", ""
        %Attached_ModelNum, "Model number of attached microphone", CAL, 15, UNINT,"", ""
        %Attached_VersionLetter, "Version letter of attached microphone", CAL, 5, CHRS1,"s", ""
        %Attached_VersionNum, "Version number of attached microphone", CAL, 6, UNINT, "", ""
        %Attached_SerialNum, "Serial number of attached microphone", CAL, 24, UNINT, "","",""
      ENDCASE
      CASE "System", 1
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
      %System_MfgrID, "Manufacturer of System", CAL, 14, UNINT, "", ""
      %System_ModelNum, "Model number of System", CAL, 15, UNINT, "", ""
      %System_VersionLetter, "Version letter of System", CAL, 5, CHR5, "s", ""
      %System_VersionNum, "Version number of System", CAL, 6, UNINT, "", ""
      %System_SerialNum, "Serial number of System", CAL, 24, UNINT, "", ""
   ENDCASE
ENDSELECT
ENUMERATE PolarizationEnum, "Pre-polarized", "28 V", "200 V"
   %RefPol, "Polarization Voltage", CAL, 2, PolarizationEnum, "e", ""

SELECTCASE "System Test (Test Gain)", ID, 1
   CASE "Not available", 0
   ENDCASE
   CASE "Available", 1
      %TestGain, "Test gain", CAL, 10, ConRes, 0.0, 0.1, "0.0", "dB"
   ENDCASE
ENDSELECT

ENUMERATE MicTypeEnum, "Free field", "Pressure", "Random incidence", "Other"
   %MicType, "Microphone Type", CAL, 2, MicTypeEnum, "e", ""

ENUMERATE MicSizeEnum, "1 inch", "1/2 inch", "1/4 inch", "1/8 inch"
   %MicSize, "Microphone Size", CAL, 2, MicSizeEnum, "e", ""


   %Equi_Vol, "Equivalent microphone volume", CAL, 8, ConRes, 0.0, 1E-9, "0.0p", " m3"

SELECTCASE "Transfer Function", ID, 1
   CASE "Not specified", 0
   ENDCASE
   CASE "Specified", 1
      ENUMERATE Resp_TypeEnum, "Actuator", "Corrected"
      %Resp_Type, "Actuator or corrected response", CAL, 1, Resp_TypeEnum, "e", ""
      %TF_HP_S, "Lowest high pass cut-off frequency (F hp)", CAL, 7, ConRelRes, 0.005, 0.05, "rp","Hz"
      %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.05, 0.01, "rp","Hz"
      %TF_SP, "F low lift (Low pass cut-off frequency)", CAL, 7, ConRelRes, 5, 0.02, "rp", "Hz"
      %TF_SZm, "F high/ F low lift", CAL, 8, ConRelRes, 1, 0.0015, "rp", ""
      %TF_KPr, "Resonance frequency (F res)", CAL, 8, ConRelRes, 2000, 0.01, "rp", "Hz"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
            %TF_KPq, "Quality factor @ F res (Q)", CAL, 8, ConRelRes, 0.2, 0.01,  "rp",""
            %TF_KPr, "Second resonance frequency (F res)", CAL, 6, ConRelRes, 10000, 0.03, "rp", "Hz"
            %TF_KPq, "Quality factor @ second F res (Q)", CAL, 7, ConRelRes, 0.2, 0.03,  "rp",""
         ENDCASE
      ENDSELECT

ENDCASE

CASE "Separate TEDS for preamplifier with Attached Microphone", 2
         %Rin, "Amplifier resistance", CAL, 6, ConRelRes, 1E8, 0.1,  "rp", "Ohm"
         %Cin, "Amplifier capacitance", CAL, 6, ConRes, 0, 0.02E-12, "0.000p", "F"
         %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
         %TF_SP, "F low lift (low pass cutoff frequency)", CAL, 7, ConRelRes, 5, 0.02, "rp", "Hz"
         %Rout, "Output impedance", CAL, 5, ConRelRes, 5, 0.10, "0.0p", "Ohm"
      ENUMERATE YesNoEnum, "No","Yes"
      %Gate,   "Gate present", ID, 1, YesNoEnum, "e",""
      SELECTCASE "Extended Functionality", ID, 1
         CASE "None", 0
            UGID "I28-0-0-0", "Preamp. w. optional Attached Mic. "
            %Gain, "Gain", CAL, 12, ConRes, -40, 0.025,  "0.000", "dB"
         ENDCASE
         CASE "Programmable Gain", 1
            UGID "I28-0-1-0", "Preamp. w. optional Attached Mic., Programmable"
            %passive[Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
            %passive[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "","" = "11"
            %passive[ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
            %passive[FunctionType], "Passive control type", ID, 2, UNINT, "","" = 0//checkmark
            %passive[Function], "Passive mode", USR, 10, BitBin, "", "" =  "xx,00"
            %gain[Initialize], "Initialize not needed", ID, 1, UNINT, "", "" = 0
            %gain[CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "","" = "11"
            %gain[ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
            %gain[FunctionType], "Gain control type", ID, 2, UNINT, "","" = 1//One Exactly
            %gain[Function], %gain["10"], USR, 4, BitBin, "", "" = "10" //High gain
            %gain[Function], %gain["01"], USR, 4, BitBin, "", "" = "01" //Low gain
            %defaultFR, "Default setting", ID, 2, UNINT, "",""
            %Passive, "Supports multiplexer mode", ID, 1, UNINT, "", ""
            %gain["01"], "Low gain", CAL, 12, ConRes, -40, 0.025, "0.000", "dB"
            %gain["10"], "High gain", CAL, 12, ConRes, -40, 0.025, "0.000", "dB"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
      ENDCASE
    ENDSELECT
    %Attached_MfgrID, "Manufacturer of attached microphone", CAL, 14, UNINT, "", ""
    %Attached_ModelNum, "Model number of attached microphone", CAL, 15, UNINT, "", ""
    %Attached_VersionLetter, "Version letter of attached microphone", CAL, 5, CHR5, "s", ""
    %Attached_VersionNum, "Version number of attached microphone", CAL, 6, UNINT, "", ""
    %Attached_SerialNum, "Serial number of attached microphone", CAL, 24, UNINT, "", ""

      %Sens@Ref, "Sensitivity @ reference condition", CAL, 16, ConRelRes, 10E-6, 0.0001, "rp", "V/Pa"
      %RefPol, "Polarization Voltage", CAL, 2, PolarizationEnum, "e", ""
      %MicType, "Microphone Type", CAL, 2, MicTypeEnum, "e", ""
      %MicSize, "Microphone Size", CAL, 2, MicSizeEnum, "e", ""
      %Cmic, "Microphone capacitance", CAL, 9, ConRes, 5E-12, 0.3E-12, "0.000p", "F"
      %Cstray, "Passive capacitance", CAL, 5, ConRelRes, 0.15E-12, 0.1,  "0.000p", "F"
      %Equi_Vol, "Equivalent microphone volume", CAL, 8, ConRes, 0.0, 1E-9, "0.0p", " m^3"
      %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.05, 0.01, "rp","Hz"
      %RefTemp, "Reference temperature (T ref)", CAL, 5, ConRes, 15, 0.5, "0.0","°C"
      %Refpress, "Reference pressure", CAL, 6, ConRes, 80000, 500, "0", "Pa"
      %Reffreq, "Reference frequency (F ref)", CAL, 8, ConRelRes, 0.35, 0.0175, "0p","Hz"
      %CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
      %CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

  ENDCASE

ENDSELECT

ENUMERATE SignEnum,"Positive","Negative"
%Sign, "Polarity (Sign)", CAL, 1, SignEnum, "e", ""

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Voltage Sensor"

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
ENUMERATE ACDCCouplingEnum,"DC","AC"
%ACDCCoupling, "AC or DC Coupling", ID, 0, ACDCCouplingEnum,"e","" = "AC"


%RefTemp, "Reference temperature (T ref)", CAL, 5, ConRes, 15, 0.5, "0.0","°C"
%Refpress, "Reference pressure", CAL, 6, ConRes, 80000, 500, "0", "Pa"
%Reffreq, "Reference frequency (F ref)", CAL, 8, ConRelRes, 0.35, 0.0175, "0p","Hz"
%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"


%MeasID, "Measurement location ID", USR, 11, UNINT,"0", ""



ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------

TEMPLATE 0,8,29,"Capacitive Microphone"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 30 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Capacitive Microphone Template
ABSTRACT For capacitive microphones without preamplier

SPACING

%Sens@Ref, "Sensitivity @ reference condition", CAL, 16, ConRelRes, 10E-6, 0.0001, "rp","V/Pa"
%RefPol, "Polarization Voltage", CAL, 2, PolarizationEnum, "e", ""
%MicType, "Microphone Type", CAL, 2, MicTypeEnum, "e", ""
%MicSize, "Microphone Size", CAL, 2, MicSizeEnum, "e", ""
%Cmic, "Microphone capacitance", CAL, 9, ConRes, 5E-12, 0.3E-12, "0.000p", "F"
%Cstray, "Passive capacitance", CAL, 5, ConRelRes, 0.15E-12, 0.1,  "0.000p", "F"
%Equi_Vol, "Equivalent microphone volume", CAL, 8, ConRes, 0.0, 1E-9, "0.0p", " m^3"
%TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.05, 0.01, "rp","Hz"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
%RefTemp, "Reference temperature (T ref)", CAL, 5, ConRes, 15, 0.5, "0.0","°C"
%Refpress, "Reference pressure", CAL, 6, ConRes, 80000, 500, "0", "Pa"
%Reffreq, "Reference frequency (F ref)", CAL, 8, ConRelRes, 0.35, 0.0175, "0p","Hz"
%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------

TEMPLATE 0,8,30,"High Level Voltage Output Sensor"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 30 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Voltage Sensor Template
ABSTRACT For sensors that linearly convert a physical measurand to a voltage output signal
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "K",           (0, 0,0, 0,0, 0, 0,1,0,0,1,0)          // Temperature: base SI unit is kelvin
PHYSICAL_UNIT "°C",          (0, 0,0, 0,0, 0, 0,1,0,0,1,-273.15)    // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "strain",      (1, 0,0, 1,0, 0, 0,0,0,0,1,0)          // Strain: equals meter/meter
PHYSICAL_UNIT "microstrain",(1, 0,0, 1,0, 0, 0,0,0,0,1e-6,0)       // microstrain = 10^(-6) x strain
PHYSICAL_UNIT "N",           (0, 0,0, 1,1,-2, 0,0,0,0,1,0)          // Force: Newton equals m•kg/s^2
PHYSICAL_UNIT "lb",          (0, 0,0, 1,1,-2, 0,0,0,0,4.44822,0)    // pound is 4.44822 x Newton
PHYSICAL_UNIT "kgf",         (0, 0,0, 1,1,-2, 0,0,0,0,9.80665,0)    // kilogram-force (kilopond) is 9.80665 x Newton
PHYSICAL_UNIT "m/s^2",       (0, 0,0, 1,0,-2, 0,0,0,0,1,0)          // Acceleration: m/s^2
PHYSICAL_UNIT "ga",          (0, 0,0, 1,0,-2, 0,0,0,0,9.80665,0)    // ga = 9.80665 m/s^2
PHYSICAL_UNIT "Nm/radian",   (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Torque: N•m/rad equals m^2•kg/(s^2•rad)
PHYSICAL_UNIT "Nm",          (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Common usage drops radian from abbreviation
PHYSICAL_UNIT "oz-in",       (0,-1,0, 2,1,-2, 0,0,0,0,0.00706155,0) // oz-in is 0.00706155•Nm
PHYSICAL_UNIT "Pa",          (0, 0,0,-1,1,-2, 0,0,0,0,1,0)          // Pressure: Pascal equals Newton/m^2 = kg/(m•s^2)
PHYSICAL_UNIT "PSI",         (0, 0,0,-1,1,-2, 0,0,0,0,6894.757,0)   // Pounds per Square Inch = 6894.757 Pa
PHYSICAL_UNIT "kg",          (0, 0,0, 0,1, 0, 0,0,0,0,1,0)          // Mass:  base SI unit is kilogram
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
PHYSICAL_UNIT "g",          (0, 0,0, 0,1, 0, 0,0,0,0,0.001,0)      // gram = 0.001 kg
PHYSICAL_UNIT "m",          (0, 0,0, 1,0, 0, 0,0,0,0,1,0)          // Distance:  base SI unit is meter
PHYSICAL_UNIT "mm",         (0, 0,0, 1,0, 0, 0,0,0,0,0.001,0)      // millimeter = 0.001 m
PHYSICAL_UNIT "in",         (0, 0,0, 1,0, 0, 0,0,0,0,0.0254,0)     // inches = 0.0254 m
PHYSICAL_UNIT "m/s",        (0, 0,0, 1,0,-1, 0,0,0,0,1,0)          // Velocity: Preferred SI unit is meter/second
PHYSICAL_UNIT "mph",        (0, 0,0, 1,0,-1, 0,0,0,0,0.44704,0)    // miles per hour = 0.44704 m/s
PHYSICAL_UNIT "fps",        (0, 0,0, 1,0,-1, 0,0,0,0,0.3048,0)     // feet per second = 0.3048 m/s
PHYSICAL_UNIT "radian",     (0, 1,0, 0,0, 0, 0,0,0,0,1,0)          // Angular position: base SI unit is radian
PHYSICAL_UNIT "degrees",    (0, 1,0, 0,0, 0, 0,0,0,0,0.0174533,0) // degrees = 0.0174533 radians
PHYSICAL_UNIT "radian/s",   (0, 1,0, 0,0,-1, 0,0,0,0,1,0)          // Angular velocity: preferred SI unit is radian/s
PHYSICAL_UNIT "rpm",        (0, 1,0, 0,0,-1, 0,0,0,0,0.104720,0)   // rev. per minute = 0.104720 radian/s
PHYSICAL_UNIT "Hz",         (0, 0,0, 0,0,-1, 0,0,0,0,1,0)          // Frequency: Hertz = 1/second
PHYSICAL_UNIT "kg/m^3",     (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // Density: kg/m^3
PHYSICAL_UNIT "g/l",        (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // gram per liter equals kg/m^3
PHYSICAL_UNIT "mole/m^3",   (0, 0,0,-3,0, 0, 0,0,1,0,1,0)          // Molar concentration: preferred SI unit is mole/m^3
PHYSICAL_UNIT "mole/l",     (0, 0,0,-3,0, 0, 0,0,1,0,1000,0)       // mole per liter = 1000 mole/m^3
PHYSICAL_UNIT "m^3/m^3",    (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // Volumetric concentration: base SI unit is m^3/m^3
PHYSICAL_UNIT "l/l",        (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // liter per liter equals m^3/m^3
PHYSICAL_UNIT "kg/s",       (0, 0,0, 0,1,-1, 0,0,0,0,1,0)          // Mass Flow:  preferred SI unit is kg/s
PHYSICAL_UNIT "m^3/s",      (0, 0,0, 3,0,-1, 0,0,0,0,1,0)          // Volumetric Flow: preferred SI unit is m^3/s
PHYSICAL_UNIT "m^3/hr",     (0, 0,0, 3,0,-1, 0,0,0,0,3600,0)       // m^3 per hour = 3600 m^3/s
PHYSICAL_UNIT "gpm",        (0, 0,0, 3,0,-1, 0,0,0,0,6.30902e-5,0) // gallons(US) per minute = 6.30902e-5 m^3/s
PHYSICAL_UNIT "cfm",        (0, 0,0, 3,0,-1, 0,0,0,0,4.71947e-4,0) // cubic feet per minute = 4.71947e-4 m^3/s
PHYSICAL_UNIT "l/min",      (0, 0,0, 3,0,-1, 0,0,0,0,1.66667e-5,0) // liters per minute = 1.66667e-5 m^3/s
PHYSICAL_UNIT "RH",         (1, 0,0,-3,1, 0, 0,0,0,0,1,0)          // Relative Humidity: = (kg/m^3)/(kg/m^3)
PHYSICAL_UNIT "%",          (1, 0,0, 0,0, 0, 0,0,0,0,100,0)        // Dimensionless Ratio: percent
PHYSICAL_UNIT "Ohm",        (0, 0,0, 2,1,-3,-2,0,0,0,1,0)          // Resistance: Ohms equals m^2•kg/(s^3•A^2)
PHYSICAL_UNIT "V",          (0, 0,0, 2,1,-3,-1,0,0,0,1,0)          // Voltage: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "V rms",      (0, 0,0, 2,1,-3,-1,0,0,0,1,0)          // Voltage RMS: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "A",          (0, 0,0, 0,0, 0, 1,0,0,0,1,0)          // Current: base SI unit is Ampere
PHYSICAL_UNIT "A rms",      (0, 0,0, 0,0, 0, 1,0,0,0,1,0)          // Current RMS: base SI unit is Ampere
PHYSICAL_UNIT "W",          (0, 0,0, 2,1,-3, 0,0,0,0,1,0)          // Watts equals m^2•kg/(sec^3)
PHYSICAL_UNIT "sec",        (0, 0,0, 0,0, 1, 0,0,0,0,1,0)          // Time: base SI unit is seconds
PHYSICAL_UNIT "days",       (0,0,0,0,0,1,0,0,0,0,86400,0)          // Time: Days = 86400 seconds
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT Sensor","Potentio-
metric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Voltage Sensor"

SELECTCASE "Physical Measurand", ID, 6
  CASE "Temperature (kelvin)", 0
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "K"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "K"
  ENDCASE
  CASE "Temperature (Celsius)", 1
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
  ENDCASE
  CASE "Strain", 2
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000p", "strain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000p", "strain"
  ENDCASE
  CASE "microstrain", 3
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
  ENDCASE
  CASE "Force/Weight (Newton)", 4
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000p", "N"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000p", "N"
  ENDCASE
  CASE "Force/Weight (pounds)", 5
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "lb"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000E+0", "lb"
  ENDCASE
  CASE "Force/Weight (kilogram-force/kilopond)", 6
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
  ENDCASE
  CASE "Acceleration (m/s^2)", 7
    %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s^2"
    %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s^2"
  ENDCASE
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
CASE "Acceleration (g)", 8
  %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
  %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
ENDCASE
CASE "Torque (Nm/radian)", 9
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
ENDCASE
CASE "Torque (Nm)", 10
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
ENDCASE
CASE "Torque (oz-in)", 11
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
ENDCASE
CASE "Pressure (Pascal)", 12
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
ENDCASE
CASE "Pressure (PSI)", 13
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
ENDCASE
CASE "Mass (kg)", 14
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
ENDCASE
CASE "Mass (g)", 15
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000p", "g"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000p", "g"
ENDCASE
CASE "Distance (m)", 16
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000p", "m"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000p", "m"
ENDCASE
CASE "Distance (mm)", 17
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
    %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
  ENDCASE
  CASE "Distance (inches)", 18
    %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
    %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
  ENDCASE
  CASE "Velocity (m/s)", 19
    %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
    %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
  ENDCASE
  CASE "Velocity (mph)", 20
    %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
    %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
  ENDCASE
  CASE "Velocity (fps)", 21
    %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
    %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
  ENDCASE
  CASE "Angular Position (radian)", 22
    %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
    %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
  ENDCASE
  CASE "Angular Position (degrees)", 23
    %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
    %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
  ENDCASE
  CASE "Rotational Velocity (radian/s)", 24
    %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
    %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
  ENDCASE
  CASE "Rotational Velocity (rpm)", 25
    %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
    %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
  ENDCASE
  CASE "Frequency", 26
    %MinPhysVal, "Minimum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
    %MaxPhysVal, "Maximum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
  ENDCASE
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
CASE "Concentration (gram/liter)", 27
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
ENDCASE
CASE "Concentration (kg/liter)", 28
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
ENDCASE
CASE "Molar Concentration (mole/m3)", 29
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
ENDCASE
CASE "Molar Concentration (mole/l)", 30
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
ENDCASE
CASE "Volumetric Concentration (m3/m3)", 31
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
ENDCASE
CASE "Volumetric Concentration (l/l)", 32
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
ENDCASE
CASE "Mass Flow", 33
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
ENDCASE
CASE "Volumetric Flow (m3/s)", 34
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
ENDCASE
CASE "Volumetric Flow (m3/hr)", 35
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
ENDCASE
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
CASE "Volumetric Flow (gpm)", 36
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
ENDCASE
CASE "Volumetric Flow (cfm)", 37
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
ENDCASE
CASE "Volumetric Flow (l/min)", 38
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
ENDCASE
CASE "Relative Humidity", 39
  %MinPhysVal, "Minimum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
  %MaxPhysVal, "Maximum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
ENDCASE
CASE "Ratio (percent)", 40
  %MinPhysVal, "Minimum Percentage", CAL, 32, SINGLE, "0.0000", "%"
  %MaxPhysVal, "Maximum Percentage", CAL, 32, SINGLE, "0.0000", "%"
ENDCASE
CASE "Voltage", 41
  %MinPhysVal, "Minimum Voltage", CAL, 32, SINGLE, "0.000p", "V"
  %MaxPhysVal, "Maximum Voltage", CAL, 32, SINGLE, "0.000p", "V"
ENDCASE
CASE "RMS Voltage", 42
  %MinPhysVal, "Minimum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
  %MaxPhysVal, "Maximum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
ENDCASE
CASE "Current", 43
  %MinPhysVal, "Minimum Current", CAL, 32, SINGLE, "0.000p", "A"
  %MaxPhysVal, "Maximum Current", CAL, 32, SINGLE, "0.000p", "A"
ENDCASE
CASE "RMS Current", 44
  %MinPhysVal, "Minimum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
  %MaxPhysVal, "Maximum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
ENDCASE
CASE "Power (Watts)", 45
  %MinPhysVal, "Minimum Power", CAL, 32, SINGLE, "0.000p", "W"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
    %MaxPhysVal, "Maximum Power", CAL, 32, SINGLE, "0.000p", "W"
  ENDCASE

ENDSELECT

SELECTCASE "Full Scale Electrical Value Precision", CAL, 2
  CASE "Standard 0-10V", 0
    %MinElecVal, "Minimum Electrical Value", CAL, 0, SINGLE, "0", "V" = 0.0
    %MaxElecVal, "Maximum Electrical Value", CAL, 0, SINGLE, "0", "V" = 10.0
  ENDCASE
  CASE "Standard ±10V", 1
    %MinElecVal, "Minimum Electrical Value", CAL, 0, SINGLE, "0", "V" = -10.0
    %MaxElecVal, "Maximum Electrical Value", CAL, 0, SINGLE, "0", "V" = 10.0
  ENDCASE
  CASE "20mV precision", 2
    %MinElecVal, "Minimum Electrical Value", CAL, 11, ConRes, -20.48, 0.02, "0.00", "V"
    %MaxElecVal, "Maximum Electrical Value", CAL, 11, ConRes, -20.48, 0.02, "0.00", "V"
  ENDCASE
  CASE "Full precision", 3
    %MinElecVal, "Minimum Electrical Value", CAL, 32, SINGLE, "0.00000p", "V"
    %MaxElecVal, "Maximum Electrical Value", CAL, 32, SINGLE, "0.00000p", "V"
  ENDCASE
ENDSELECT

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

ENUMERATE ACDCCouplingEnum,"DC","AC"
%ACDCCoupling, "AC or DC Coupling", ID, 1, ACDCCouplingEnum,"e",""

%SensorImped, "Output Impedance of the Sensor", ID, 12, ConRelRes, 1, 0.001706, "rp", "Ohm"

%RespTime, "Transducer Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"

SELECTCASE "Excitation/Power Requirements", ID, 1
  CASE "No excitation or power supply required", 0
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  ENDCASE
  CASE "Sensor requires excitation/power", 1
    %ExciteAmplNom, "Excitation Level (Nominal)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
    %ExciteAmplMin, "Excitation Level (Minimum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
    %ExciteAmplMax, "Excitation Level (Maximum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
    ENUMERATE ExciteTypeEnum, "DC", "Bipolar DC", "AC (rms)"
    %ExciteType, "Excitation Voltage Type", ID, 2, ExciteTypeEnum,"e",""
    %ExciteCurrentDraw, "Maximum current draw at nominal excitation level", ID, 6, ConRelRes, 1e-6, 0.129462705898,
"rp", "A"
  ENDCASE
ENDSELECT

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT,"0", ""

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------


TEMPLATE 0,8,31,"Current Loop Output Sensor"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 31 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Current Loop Output Sensor Template
ABSTRACT For sensors that linearly convert a physical measurand to a current output signal
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "K",           (0, 0,0, 0,0, 0, 0,1,0,0,1,0)         // Temperature: base SI unit is kelvin
PHYSICAL_UNIT "°C",          (0, 0,0, 0,0, 0, 0,1,0,0,1,-273.15)   // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "strain",      (1, 0,0, 1,0, 0, 0,0,0,0,1,0)         // Strain: equals meter/meter
PHYSICAL_UNIT "microstrain",(1, 0,0, 1,0, 0, 0,0,0,0,1e-6,0)      // microstrain = 10^(-6) x strain
PHYSICAL_UNIT "N",           (0, 0,0, 1,1,-2, 0,0,0,0,1,0)         // Force: Newton equals m•kg/s2
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
PHYSICAL_UNIT "lb",          (0, 0,0, 1,1,-2, 0,0,0,0,4.44822,0)    // pound is 4.44822 x Newton
PHYSICAL_UNIT "kgf",         (0, 0,0, 1,1,-2, 0,0,0,0,9.80665,0)    // kilogram-force (kilopond) is 9.80665 x Newton
PHYSICAL_UNIT "m/s²",        (0, 0,0, 1,0,-2, 0,0,0,0,1,0)          // Acceleration: m/s²
PHYSICAL_UNIT "ga",          (0, 0,0, 1,0,-2, 0,0,0,0,9.80665,0)    // ga = 9.80665 m/s²
PHYSICAL_UNIT "Nm/radian",   (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Torque: N•m/rad equals m²•kg/(s²•rad)
PHYSICAL_UNIT "Nm",          (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Common usage drops radian from abbreviation
PHYSICAL_UNIT "oz-in",       (0,-1,0, 2,1,-2, 0,0,0,0,0.00706155,0)// oz-in is 0.00706155•Nm
PHYSICAL_UNIT "Pa",          (0, 0,0,-1,1,-2, 0,0,0,0,1,0)          // Pressure: Pascal equals Newton/m² = kg/(m•s²)
PHYSICAL_UNIT "PSI",         (0, 0,0,-1,1,-2, 0,0,0,0,6894.757,0)   // Pounds per Square Inch = 6894.757 Pa
PHYSICAL_UNIT "kg",          (0, 0,0, 0,1, 0, 0,0,0,0,1,0)          // Mass:  base SI unit is kilogram
PHYSICAL_UNIT "g",           (0, 0,0, 0,1, 0, 0,0,0,0,0.001,0)      // gram = 0.001 kg
PHYSICAL_UNIT "m",           (0, 0,0, 1,0, 0, 0,0,0,0,1,0)          // Distance:  base SI unit is meter
PHYSICAL_UNIT "mm",          (0, 0,0, 1,0, 0, 0,0,0,0,0.001,0)      // millimeter = 0.001 m
PHYSICAL_UNIT "in",          (0, 0,0, 1,0, 0, 0,0,0,0,0.0254,0)     // inches = 0.0254 m
PHYSICAL_UNIT "m/s",         (0, 0,0, 1,0,-1, 0,0,0,0,1,0)          // Velocity: Preferred SI unit is meter/second
PHYSICAL_UNIT "mph",         (0, 0,0, 1,0,-1, 0,0,0,0,0.44704,0)    // miles per hour = 0.44704 m/s
PHYSICAL_UNIT "fps",         (0, 0,0, 1,0,-1, 0,0,0,0,0.3048,0)     // feet per second = 0.3048 m/s
PHYSICAL_UNIT "radian",      (0, 1,0, 0,0, 0, 0,0,0,0,1,0)          // Angular position: base SI unit is radian
PHYSICAL_UNIT "degrees",     (0, 1,0, 0,0, 0, 0,0,0,0,0.0174533,0)  // degrees = 0.0174533 radians
PHYSICAL_UNIT "radian/s",    (0, 1,0, 0,0,-1, 0,0,0,0,1,0)          // Angular velocity: preferred SI unit is radian/s
PHYSICAL_UNIT "rpm",         (0, 1,0, 0,0,-1, 0,0,0,0,0.104720,0)   // rev. per minute = 0.104720 radian/s
PHYSICAL_UNIT "Hz",          (0, 0,0, 0,0,-1, 0,0,0,0,1,0)          // Frequency: Hertz = 1/second
PHYSICAL_UNIT "kg/m³",       (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // Density: kg/m³
PHYSICAL_UNIT "g/l",         (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // gram per liter equals kg/m³
PHYSICAL_UNIT "mole/m³",     (0, 0,0,-3,0, 0, 0,0,1,0,1,0)          // Molar concentration: preferred SI unit is mole/m³
PHYSICAL_UNIT "mole/l",      (0, 0,0,-3,0, 0, 0,0,1,0,1000,0)       // mole per liter = 1000 mole/m³
PHYSICAL_UNIT "m³/m³",       (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // Volumetric concentration: base SI unit is m³/m³
PHYSICAL_UNIT "l/l",         (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // liter per liter equals m³/m³
PHYSICAL_UNIT "kg/s",        (0, 0,0, 0,1,-1, 0,0,0,0,1,0)          // Mass Flow:  preferred SI unit is kg/s
PHYSICAL_UNIT "m³/s",        (0, 0,0, 3,0,-1, 0,0,0,0,1,0)          // Volumetric Flow: preferred SI unit is m³/s
PHYSICAL_UNIT "m³/hr",       (0, 0,0, 3,0,-1, 0,0,0,0,3600,0)       // m³ per hour = 3600 m³/s
PHYSICAL_UNIT "gpm",         (0, 0,0, 3,0,-1, 0,0,0,0,6.30902e-5,0)// gallons(US) per minute = 6.30902e-5 m³/s
PHYSICAL_UNIT "cfm",         (0, 0,0, 3,0,-1, 0,0,0,0,4.71947e-4,0)// cubic feet per minute = 4.71947e-4 m³/s
PHYSICAL_UNIT "l/min",       (0, 0,0, 3,0,-1, 0,0,0,0,1.66667e-5,0)// liters per minute = 1.66667e-5 m³/s
PHYSICAL_UNIT "RH",          (1, 0,0,-3,1, 0, 0,0,0,0,1,0)          // Relative Humidity: = (kg/m³)/(kg/m³)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
PHYSICAL_UNIT "%",          (1, 0,0, 0,0, 0, 0,0,0,0,100,0)       // Dimensionless Ratio: percent
PHYSICAL_UNIT "Ohm",        (0, 0,0, 2,1,-3,-2,0,0,0,1,0)         // Resistance: Ohms equals m^2•kg/(s^3•A^2)
PHYSICAL_UNIT "V",          (0, 0,0, 2,1,-3,-1,0,0,0,1,0)         // Voltage: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "V rms",      (0, 0,0, 2,1,-3,-1,0,0,0,1,0)         // Voltage RMS: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "A",          (0, 0,0, 0,0, 0, 1,0,0,0,1,0)         // Current: base SI unit is Ampere
PHYSICAL_UNIT "A rms",      (0, 0,0, 0,0, 0, 1,0,0,0,1,0)         // Current RMS: base SI unit is Ampere
PHYSICAL_UNIT "mA",         (0, 0,0, 0,0, 0, 1,0,0,0,0.001,0)     // milliAmpere = 0.001 x Ampere
PHYSICAL_UNIT "W",          (0, 0,0, 2,1,-3, 0,0,0,0,1,0)         // Watts equals m^2•kg/(sec^3)
PHYSICAL_UNIT "sec",        (0, 0,0, 0,0, 1, 0,0,0,0,1,0)         // Time: base SI unit is seconds
PHYSICAL_UNIT "days",       (0,0,0,0,0,1,0,0,0,0,86400,0)         // Time: Days = 86400 seconds

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Current Sensor"

SELECTCASE "Physical Measurand", ID, 6
  CASE "Temperature (kelvin)", 0
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "K"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "K"
  ENDCASE
  CASE "Temperature (Celsius)", 1
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
  ENDCASE
  CASE "Strain", 2
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000p", "strain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000p", "strain"
  ENDCASE
  CASE "microstrain", 3
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
  ENDCASE
  CASE "Force/Weight (Newton)", 4
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000p", "N"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000p", "N"
  ENDCASE
  CASE "Force/Weight (pounds)", 5
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "lb"
  %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000E+0", "lb"
ENDCASE
CASE "Force/Weight (kilogram-force/kilopond)", 6
  %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
  %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
ENDCASE
CASE "Acceleration (m/s2)", 7
  %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s2"
  %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s2"
ENDCASE
CASE "Acceleration (g)", 8
  %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
  %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
ENDCASE
CASE "Torque (Nm/radian)", 9
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
ENDCASE
CASE "Torque (Nm)", 10
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
ENDCASE
CASE "Torque (oz-in)", 11
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
ENDCASE
CASE "Pressure (Pascal)", 12
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
ENDCASE
CASE "Pressure (PSI)", 13
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
ENDCASE
CASE "Mass (kg)", 14
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
ENDCASE
CASE "Mass (g)", 15
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000p", "g"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000p", "g"
ENDCASE
CASE "Distance (m)", 16
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000p", "m"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000p", "m"
ENDCASE
CASE "Distance (mm)", 17
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
ENDCASE
CASE "Distance (inches)", 18
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
ENDCASE
CASE "Velocity (m/s)", 19
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
ENDCASE
CASE "Velocity (mph)", 20
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
ENDCASE
CASE "Velocity (fps)", 21
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
ENDCASE
CASE "Angular Position (radian)", 22
  %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
  %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
ENDCASE
CASE "Angular Position (degrees)", 23
  %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
  %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
ENDCASE
CASE "Rotational Velocity (radian/s)", 24
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
ENDCASE
CASE "Rotational Velocity (rpm)", 25
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
ENDCASE
CASE "Frequency", 26
  %MinPhysVal, "Minimum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
  %MaxPhysVal, "Maximum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
ENDCASE
CASE "Concentration (gram/liter)", 27
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
ENDCASE
CASE "Concentration (kg/liter)", 28
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
ENDCASE
CASE "Molar Concentration (mole/m3)", 29
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
ENDCASE
CASE "Molar Concentration (mole/l)", 30
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
ENDCASE
CASE "Volumetric Concentration (m3/m3)", 31
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
ENDCASE
CASE "Volumetric Concentration (l/l)", 32
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
ENDCASE
CASE "Mass Flow", 33
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
ENDCASE
CASE "Volumetric Flow (m3/s)", 34
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
ENDCASE
CASE "Volumetric Flow (m3/hr)", 35
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
ENDCASE
CASE "Volumetric Flow (gpm)", 36
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
ENDCASE
CASE "Volumetric Flow (cfm)", 37
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
ENDCASE
CASE "Volumetric Flow (l/min)", 38
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
ENDCASE
CASE "Relative Humidity", 39
  %MinPhysVal, "Minimum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
  %MaxPhysVal, "Maximum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
ENDCASE
CASE "Ratio (percent)", 40
  %MinPhysVal, "Minimum Percentage", CAL, 32, SINGLE, "0.0000", "%"
  %MaxPhysVal, " Maximum Percentage", CAL, 32, SINGLE, "0.0000", "%"
ENDCASE
CASE "Voltage", 41
  %MinPhysVal, " Minimum Voltage", CAL, 32, SINGLE, "0.000p", "V"
  %MaxPhysVal, " Maximum Voltage", CAL, 32, SINGLE, "0.000p", "V"
ENDCASE
CASE "RMS Voltage", 42
  %MinPhysVal, " Minimum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
  %MaxPhysVal, " Maximum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  ENDCASE
  CASE "Current", 43
    %MinPhysVal, " Minimum Current", CAL, 32, SINGLE, "0.000p", "A"
    %MaxPhysVal, " Maximum Current", CAL, 32, SINGLE, "0.000p", "A"
  ENDCASE
  CASE "RMS Current", 44
    %MinPhysVal, " Minimum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
    %MaxPhysVal, " Maximum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
  ENDCASE
  CASE "Power (Watts)", 45
    %MinPhysVal, " Minimum Power", CAL, 32, SINGLE, "0.000p", "W"
    %MaxPhysVal, " Maximum Power", CAL, 32, SINGLE, "0.000p", "W"
  ENDCASE
ENDSELECT

SELECTCASE "Full Scale Electrical Value Precision", CAL, 1
  CASE "Standard 4-20mA", 0
    %MinElecVal, "Minimum Electrical Value", CAL, 0, SINGLE, "0", "mA" = 4.0
    %MaxElecVal, "Maximum Electrical Value", CAL, 0, SINGLE, "0", "mA" = 20.0
  ENDCASE
  CASE "Full precision", 1
    %MinElecVal, "Minimum Electrical Value", CAL, 32, SINGLE, "0.00000p", "A"
    %MaxElecVal, "Maximum Electrical Value", CAL, 32, SINGLE, "0.00000p", "A"
  ENDCASE
ENDSELECT

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

%RespTime, "Transducer Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"

SELECTCASE "Loop powered versus external powered", ID, 1
  CASE "Loop powered sensor", 0
    %LoopSupplyMin, "Loop Supply (Minimum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
    %LoopSupplyMax, "Loop Supply (Maximum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
  ENDCASE
  CASE "Sensor requires external excitation/power", 1
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
    %ExciteAmplNom, "Excitation Level (Nominal)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
    %ExciteAmplMin, "Excitation Level (Minimum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
    %ExciteAmplMax, "Excitation Level (Maximum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
    ENUMERATE ExciteTypeEnum, "DC", "Bipolar DC", "AC (rms) "
    %ExciteType, "Excitation Voltage Type", ID, 1, ExciteTypeEnum,"e",""
    %ExciteCurrentDraw, "Maximum current draw at nominal excitation level", ID, 6, ConRelRes, 1e-6, 0.129462705898,
"rp", "A"
  ENDCASE
ENDSELECT


%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID",  USR, 11, UNINT,"0",""

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------


TEMPLATE 0,8,32,"Resistance Sensor"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 32 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Resistance Sensor Template
ABSTRACT For sensors that provide a resistance output to a measuring device
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "K",           (0, 0,0, 0,0, 0, 0,1,0,0,1,0)          // Temperature: base SI unit is kelvin
PHYSICAL_UNIT "°C",          (0, 0,0, 0,0, 0, 0,1,0,0,1,-273.15)    // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "strain",      (1, 0,0, 1,0, 0, 0,0,0,0,1,0)          // Strain: equals meter/meter
PHYSICAL_UNIT "microstrain",(1, 0,0, 1,0, 0, 0,0,0,0,1e-6,0)       // microstrain = 10^(-6) x strain
PHYSICAL_UNIT "N",           (0, 0,0, 1,1,-2, 0,0,0,0,1,0)          // Force: Newton equals m•kg/s2
PHYSICAL_UNIT "lb",          (0, 0,0, 1,1,-2, 0,0,0,0,4.44822,0)    // pound is 4.44822 x Newton
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
PHYSICAL_UNIT "kgf",          (0, 0,0, 1,1,-2, 0,0,0,0,9.80665,0)    // kilogram-force (kilopond) is 9.80665 x Newton
PHYSICAL_UNIT "m/s²",         (0, 0,0, 1,0,-2, 0,0,0,0,1,0)          // Acceleration: m/s²
PHYSICAL_UNIT "ga",           (0, 0,0, 1,0,-2, 0,0,0,0,9.80665,0)    // ga = 9.80665 m/s²
PHYSICAL_UNIT "Nm/radian",    (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Torque: N•m/rad equals m²•kg/(s²•rad)
PHYSICAL_UNIT "Nm",           (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Common usage drops radian from abbreviation
PHYSICAL_UNIT "oz-in",        (0,-1,0, 2,1,-2, 0,0,0,0,0.00706155,0)// oz-in is 0.00706155•Nm
PHYSICAL_UNIT "Pa",           (0, 0,0,-1,1,-2, 0,0,0,0,1,0)          // Pressure: Pascal equals Newton/m² = kg/(m•s²)
PHYSICAL_UNIT "PSI",          (0, 0,0,-1,1,-2, 0,0,0,0,6894.757,0)   // Pounds per Square Inch = 6894.757 Pa
PHYSICAL_UNIT "kg",           (0, 0,0, 0,1, 0, 0,0,0,0,1,0)          // Mass:  base SI unit is kilogram
PHYSICAL_UNIT "g",            (0, 0,0, 0,1, 0, 0,0,0,0,0.001,0)      // gram = 0.001 kg
PHYSICAL_UNIT "m",            (0, 0,0, 1,0, 0, 0,0,0,0,1,0)          // Distance:  base SI unit is meter
PHYSICAL_UNIT "mm",           (0, 0,0, 1,0, 0, 0,0,0,0,0.001,0)      // millimeter = 0.001 m
PHYSICAL_UNIT "in",           (0, 0,0, 1,0, 0, 0,0,0,0,0.0254,0)     // inches = 0.0254 m
PHYSICAL_UNIT "m/s",          (0, 0,0, 1,0,-1, 0,0,0,0,1,0)          // Velocity: Preferred SI unit is meter/second
PHYSICAL_UNIT "mph",          (0, 0,0, 1,0,-1, 0,0,0,0,0.44704,0)    // miles per hour = 0.44704 m/s
PHYSICAL_UNIT "fps",          (0, 0,0, 1,0,-1, 0,0,0,0,0.3048,0)     // feet per second = 0.3048 m/s
PHYSICAL_UNIT "radian",       (0, 1,0, 0,0, 0, 0,0,0,0,1,0)          // Angular position: base SI unit is radian
PHYSICAL_UNIT "degrees",      (0, 1,0, 0,0, 0, 0,0,0,0,0.0174533,0)  // degrees = 0.0174533 radians
PHYSICAL_UNIT "radian/s",     (0, 1,0, 0,0,-1, 0,0,0,0,1,0)          // Angular velocity: preferred SI unit is radian/s
PHYSICAL_UNIT "rpm",          (0, 1,0, 0,0,-1, 0,0,0,0,0.104720,0)   // rev. per minute = 0.104720 radian/s
PHYSICAL_UNIT "Hz",           (0, 0,0, 0,0,-1, 0,0,0,0,1,0)          // Frequency: Hertz = 1/second
PHYSICAL_UNIT "kg/m³",        (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // Density: kg/m³
PHYSICAL_UNIT "g/l",          (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // gram per liter equals kg/m³
PHYSICAL_UNIT "mole/m³",      (0, 0,0,-3,0, 0, 0,0,1,0,1,0)          // Molar concentration: preferred SI unit is mole/m³
PHYSICAL_UNIT "mole/l",       (0, 0,0,-3,0, 0, 0,0,1,0,1000,0)       // mole per liter = 1000 mole/m³
PHYSICAL_UNIT "m³/m³",        (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // Volumetric concentration: base SI unit is m³/m³
PHYSICAL_UNIT "l/l",          (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // liter per liter equals m³/m³
PHYSICAL_UNIT "kg/s",         (0, 0,0, 0,1,-1, 0,0,0,0,1,0)          // Mass Flow:  preferred SI unit is kg/s
PHYSICAL_UNIT "m³/s",         (0, 0,0, 3,0,-1, 0,0,0,0,1,0)          // Volumetric Flow: preferred SI unit is m³/s
PHYSICAL_UNIT "m³/hr",        (0, 0,0, 3,0,-1, 0,0,0,0,3600,0)       // m³ per hour = 3600 m³/s
PHYSICAL_UNIT "gpm",          (0, 0,0, 3,0,-1, 0,0,0,0,6.30902e-5,0)// gallons(US) per minute = 6.30902e-5 m³/s
PHYSICAL_UNIT "cfm",          (0, 0,0, 3,0,-1, 0,0,0,0,4.71947e-4,0)// cubic feet per minute = 4.71947e-4 m³/s
PHYSICAL_UNIT "l/min",        (0, 0,0, 3,0,-1, 0,0,0,0,1.66667e-5,0)// liters per minute = 1.66667e-5 m³/s
PHYSICAL_UNIT "RH",           (1, 0,0,-3,1, 0, 0,0,0,0,1,0)          // Relative Humidity: = (kg/m³)/(kg/m³)
PHYSICAL_UNIT "%",            (1, 0,0, 0,0, 0, 0,0,0,0,100,0)        // Dimensionless Ratio: percent
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
PHYSICAL_UNIT "Ohm",          (0, 0,0, 2,1,-3,-2,0,0,0,1,0)        // Resistance: Ohms equals m²•kg/(s³•A²)
PHYSICAL_UNIT "V",            (0, 0,0, 2,1,-3,-1,0,0,0,1,0)        // Voltage: Volts equals m²•kg/(sec³•A)
PHYSICAL_UNIT "V rms",        (0, 0,0, 2,1,-3,-1,0,0,0,1,0)        // Voltage RMS: Volts equals m²•kg/(sec³•A)
PHYSICAL_UNIT "A",            (0, 0,0, 0,0, 0, 1,0,0,0,1,0)        // Current: base SI unit is Ampere
PHYSICAL_UNIT "A rms",        (0, 0,0, 0,0, 0, 1,0,0,0,1,0)        // Current: base SI unit is Ampere
PHYSICAL_UNIT "W",            (0, 0,0, 2,1,-3, 0,0,0,0,1,0)        // Watts equals m²•kg/(sec³)
PHYSICAL_UNIT "sec",          (0, 0,0, 0,0, 1, 0,0,0,0,1,0)        // Time: base SI unit is seconds
PHYSICAL_UNIT "days",         (0,0,0,0,0,1,0,0,0,0,86400,0)        // Time: Days = 86400 seconds

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Resistance Sensor"

SELECTCASE "Physical Measurand", ID, 6
  CASE "Temperature (kelvin)", 0
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "K"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "K"
  ENDCASE
  CASE "Temperature (Celsius)", 1
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
  ENDCASE
  CASE "Strain", 2
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000p", "strain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000p", "strain"
  ENDCASE
  CASE "microstrain", 3
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
  ENDCASE
  CASE "Force/Weight (Newton)", 4
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000p", "N"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000p", "N"
  ENDCASE
  CASE "Force/Weight (pounds)", 5
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "lb"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000E+0", "lb"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
ENDCASE
CASE "Force/Weight (kilogram-force/kilopond)", 6
  %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
  %MaxPhysVal, "Maximum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
ENDCASE
CASE "Acceleration (m/s^2)", 7
  %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s^2"
  %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s^2"
ENDCASE
CASE "Acceleration (g)", 8
  %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
  %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
ENDCASE
CASE "Torque (Nm/radian)", 9
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
ENDCASE
CASE "Torque (Nm)", 10
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
ENDCASE
CASE "Torque (oz-in)", 11
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
ENDCASE
CASE "Pressure (Pascal)", 12
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
ENDCASE
CASE "Pressure (PSI)", 13
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
ENDCASE
CASE "Mass (kg)", 14
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
ENDCASE
CASE "Mass (g)", 15
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000p", "g"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000p", "g"
ENDCASE
CASE "Distance (m)", 16
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000p", "m"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000p", "m"
ENDCASE
CASE "Distance (mm)", 17
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
ENDCASE
CASE "Distance (inches)", 18
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
ENDCASE
CASE "Velocity (m/s)", 19
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
ENDCASE
CASE "Velocity (mph)", 20
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
ENDCASE
CASE "Velocity (fps)", 21
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
ENDCASE
CASE "Angular Position (radian)", 22
  %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
  %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
ENDCASE
CASE "Angular Position (degrees)", 23
  %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
  %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
ENDCASE
CASE "Rotational Velocity (radian/s)", 24
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
ENDCASE
CASE "Rotational Velocity (rpm)", 25
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
ENDCASE
CASE "Frequency", 26
  %MinPhysVal, "Minimum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
  %MaxPhysVal, "Maximum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
ENDCASE
CASE "Concentration (gram/liter)", 27
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
ENDCASE
CASE "Concentration (kg/liter)", 28
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
ENDCASE
CASE "Molar Concentration (mole/m3)", 29
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
ENDCASE
CASE "Molar Concentration (mole/l)", 30
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
ENDCASE
CASE "Volumetric Concentration (m3/m3)", 31
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
ENDCASE
CASE "Volumetric Concentration (l/l)", 32
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
ENDCASE
CASE "Mass Flow", 33
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
ENDCASE
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
CASE "Volumetric Flow (m3/s)", 34
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
ENDCASE
CASE "Volumetric Flow (m3/hr)", 35
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
ENDCASE
CASE "Volumetric Flow (gpm)", 36
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
ENDCASE
CASE "Volumetric Flow (cfm)", 37
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
ENDCASE
CASE "Volumetric Flow (l/min)", 38
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
ENDCASE
CASE "Relative Humidity", 39
  %MinPhysVal, "Minimum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
  %MaxPhysVal, "Maximum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
ENDCASE
CASE "Ratio (percent)", 40
  %MinPhysVal, "Minimum Percentage", CAL, 32, SINGLE, "0.0000", "%"
  %MaxPhysVal, "Maximum Percentage", CAL, 32, SINGLE, "0.0000", "%"
ENDCASE
CASE "Voltage", 41
  %MinPhysVal, "Minimum Voltage", CAL, 32, SINGLE, "0.000p", "V"
  %MaxPhysVal, "Maximum Voltage", CAL, 32, SINGLE, "0.000p", "V"
ENDCASE
CASE "RMS Voltage", 42
  %MinPhysVal, "Minimum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
  %MaxPhysVal, "Maximum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
ENDCASE
CASE "Current", 43
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
    %MinPhysVal, "Minimum Current", CAL, 32, SINGLE, "0.000p", "A"
    %MaxPhysVal, "Maximum Current", CAL, 32, SINGLE, "0.000p", "A"
  ENDCASE
  CASE "RMS Current", 44
    %MinPhysVal, "Minimum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
    %MaxPhysVal, "Maximum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
  ENDCASE
  CASE "Power (Watts)", 45
    %MinPhysVal, "Minimum Power", CAL, 32, SINGLE, "0.000p", "W"
    %MaxPhysVal, "Maximum Power", CAL, 32, SINGLE, "0.000p", "W"
  ENDCASE
ENDSELECT

SELECTCASE "Full Scale Electrical Value Precision", CAL, 2
  CASE "Values to 1270 Ohm in 10 Ohm steps", 0
    %MinElecVal, "Minimum Electrical Value", CAL, 7, ConRes, 0, 10, "0", "Ohm"
    %MaxElecVal, "Maximum Electrical Value", CAL, 7, ConRes, 0, 10, "0", "Ohm"
  ENDCASE
  CASE "Values to 1.023 MOhm in 1 kOhm steps", 1
    %MinElecVal, "Minimum Electrical Value", CAL, 10, ConRes, 0, 1000, "0.###p", "Ohm"
    %MaxElecVal, "Maximum Electrical Value", CAL, 10, ConRes, 0, 1000, "0.###p", "Ohm"
  ENDCASE
  CASE "Values to 65.535 kOhm in 1 Ohm steps", 2
    %MinElecVal, "Minimum Electrical Value", CAL, 16, ConRes, 0, 1, "0.###p", "Ohm"
    %MaxElecVal, "Maximum Electrical Value", CAL, 16, ConRes, 0, 1, "0.###p", "Ohm"
  ENDCASE
  CASE "Full precision", 3
    %MinElecVal, "Minimum Electrical Value", CAL, 32, SINGLE, "0.00000p", "Ohm"
    %MaxElecVal, "Maximum Electrical Value", CAL, 32, SINGLE, "0.00000p", "Ohm"
  ENDCASE
ENDSELECT

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 2, MapMethEnum,"e",""

%RespTime, "Transducer Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved

```
%ExciteAmplNom, "Excitation current (Nominal)", CAL, 8, ConRelRes, 1e-6, 0.02356427402545, "rp", "A"
%ExciteAmplMax, "Excitation current (Maximum)", ID, 8, ConRelRes, 1e-6, 0.02356427402545, "rp", "A"

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT,"0", ""

ENDTEMPLATE

//-----------------------------------------------------------------------------------------------------------


TEMPLATE 0,8,33,"Bridge Sensor"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 33 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Bridge Sensor Template
ABSTRACT For sensors that provide a bridge circuit to a measuring device
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "K",           (0, 0,0, 0,0, 0, 0,1,0,0,1,0)          // Temperature: base SI unit is kelvin
PHYSICAL_UNIT "°C",          (0, 0,0, 0,0, 0, 0,1,0,0,1,-273.15)    // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "strain",      (1, 0,0, 1,0, 0, 0,0,0,0,1,0)          // Strain: equals meter/meter
PHYSICAL_UNIT "microstrain",(1, 0,0, 1,0, 0, 0,0,0,0,1e-6,0)       // microstrain = 10^(-6) x strain
PHYSICAL_UNIT "N",           (0, 0,0, 1,1,-2, 0,0,0,0,1,0)          // Force: Newton equals m•kg/s2
PHYSICAL_UNIT "lb",          (0, 0,0, 1,1,-2, 0,0,0,0,4.44822,0)    // pound is 4.44822 x Newton
PHYSICAL_UNIT "kgf",         (0, 0,0, 1,1,-2, 0,0,0,0,9.80665,0)    // kilogram-force (kilopond) is 9.80665 x Newton
PHYSICAL_UNIT "m/s2",        (0, 0,0, 1,0,-2, 0,0,0,0,1,0)          // Acceleration: m/s2
PHYSICAL_UNIT "ga",          (0, 0,0, 1,0,-2, 0,0,0,0,9.80665,0)    // ga = 9.80665 m/s2
PHYSICAL_UNIT "Nm/radian",   (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Torque: N•m/rad equals m2•kg/(s2•rad)
PHYSICAL_UNIT "Nm",          (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Common usage drops radian from abbreviation
PHYSICAL_UNIT "oz-in",       (0,-1,0, 2,1,-2, 0,0,0,0,0.00706155,0)// oz-in is 0.00706155•Nm
PHYSICAL_UNIT "Pa",          (0, 0,0,-1,1,-2, 0,0,0,0,1,0)          // Pressure: Pascal equals Newton/m2 = kg/(m•s2)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
PHYSICAL_UNIT "PSI",          (0, 0,0,-1,1,-2, 0,0,0,0,6894.757,0)  // Pounds per Square Inch = 6894.757 Pa
PHYSICAL_UNIT "kg",           (0, 0,0, 0,1, 0, 0,0,0,0,1,0)         // Mass:  base SI unit is kilogram
PHYSICAL_UNIT "g",            (0, 0,0, 0,1, 0, 0,0,0,0,0.001,0)     // gram = 0.001 kg
PHYSICAL_UNIT "m",            (0, 0,0, 1,0, 0, 0,0,0,0,1,0)         // Distance:  base SI unit is meter
PHYSICAL_UNIT "mm",           (0, 0,0, 1,0, 0, 0,0,0,0,0.001,0)     // millimeter = 0.001 m
PHYSICAL_UNIT "in",           (0, 0,0, 1,0, 0, 0,0,0,0,0.0254,0)    // inches = 0.0254 m
PHYSICAL_UNIT "m/s",          (0, 0,0, 1,0,-1, 0,0,0,0,1,0)         // Velocity: Preferred SI unit is meter/second
PHYSICAL_UNIT "mph",          (0, 0,0, 1,0,-1, 0,0,0,0,0.44704,0)   // miles per hour = 0.44704 m/s
PHYSICAL_UNIT "fps",          (0, 0,0, 1,0,-1, 0,0,0,0,0.3048,0)    // feet per second = 0.3048 m/s
PHYSICAL_UNIT "radian",       (0, 1,0, 0,0, 0, 0,0,0,0,1,0)         // Angular position: base SI unit is radian
PHYSICAL_UNIT "degrees",      (0, 1,0, 0,0, 0, 0,0,0,0,0.0174533,0) // degrees = 0.0174533 radians
PHYSICAL_UNIT "radian/s",     (0, 1,0, 0,0,-1, 0,0,0,0,1,0)         // Angular velocity: preferred SI unit is radian/s
PHYSICAL_UNIT "rpm",          (0, 1,0, 0,0,-1, 0,0,0,0,0.104720,0)  // rev. per minute = 0.104720 radian/s
PHYSICAL_UNIT "Hz",           (0, 0,0, 0,0,-1, 0,0,0,0,1,0)         // Frequency: Hertz = 1/second
PHYSICAL_UNIT "kg/m^3",       (0, 0,0,-3,1, 0, 0,0,0,0,1,0)         // Density: kg/m^3
PHYSICAL_UNIT "g/l",          (0, 0,0,-3,1, 0, 0,0,0,0,1,0)         // gram per liter equals kg/m^3
PHYSICAL_UNIT "mole/m^3",     (0, 0,0,-3,0, 0, 0,0,1,0,1,0)         // Molar concentration: preferred SI unit is mole/m^3
PHYSICAL_UNIT "mole/l",       (0, 0,0,-3,0, 0, 0,0,1,0,1000,0)      // mole per liter = 1000 mole/m^3
PHYSICAL_UNIT "m^3/m^3",      (1, 0,0, 3,0, 0, 0,0,0,0,1,0)         // Volumetric concentration: base SI unit is m^3/m^3
PHYSICAL_UNIT "l/l",          (1, 0,0, 3,0, 0, 0,0,0,0,1,0)         // liter per liter equals m^3/m^3
PHYSICAL_UNIT "kg/s",         (0, 0,0, 0,1,-1, 0,0,0,0,1,0)         // Mass Flow:  preferred SI unit is kg/s
PHYSICAL_UNIT "m^3/s",        (0, 0,0, 3,0,-1, 0,0,0,0,1,0)         // Volumetric Flow: preferred SI unit is m^3/s
PHYSICAL_UNIT "m^3/hr",       (0, 0,0, 3,0,-1, 0,0,0,0,3600,0)      // m^3 per hour = 3600 m^3/s
PHYSICAL_UNIT "gpm",          (0, 0,0, 3,0,-1, 0,0,0,0,6.30902e-5,0)// gallons(US) per minute = 6.30902e-5 m^3/s
PHYSICAL_UNIT "cfm",          (0, 0,0, 3,0,-1, 0,0,0,0,4.71947e-4,0)// cubic feet per minute = 4.71947e-4 m^3/s
PHYSICAL_UNIT "l/min",        (0, 0,0, 3,0,-1, 0,0,0,0,1.66667e-5,0)// liters per minute = 1.66667e-5 m^3/s
PHYSICAL_UNIT "RH",           (1, 0,0,-3,1, 0, 0,0,0,0,1,0)         // Relative Humidity: = (kg/m^3)/(kg/m^3)
PHYSICAL_UNIT "%",            (1, 0,0, 0,0, 0, 0,0,0,0,100,0)       // Dimensionless Ratio: percent
PHYSICAL_UNIT "V/V",          (1, 0,0, 2,1,-3,-1,0,0,0,1,0)         // Voltage ratio: [m^2•kg/(sec^3•A)]/[m^2•kg/(sec^3•A)]
PHYSICAL_UNIT "Ohm",          (0, 0,0, 2,1,-3,-2,0,0,0,1,0)         // Resistance: Ohms equals m^2•kg/(s^3•A^2)
PHYSICAL_UNIT "V",            (0, 0,0, 2,1,-3,-1,0,0,0,1,0)         // Voltage: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "V rms",        (0, 0,0, 2,1,-3,-1,0,0,0,1,0)         // Voltage RMS: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "A",            (0, 0,0, 0,0, 0, 1,0,0,0,1,0)         // Current: base SI unit is Ampere
PHYSICAL_UNIT "A rms",        (0, 0,0, 0,0, 0, 1,0,0,0,1,0)         // Current RMS: base SI unit is Ampere
PHYSICAL_UNIT "W",            (0, 0,0, 2,1,-3, 0,0,0,0,1,0)         // Watts equals m^2•kg/(sec^3)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
PHYSICAL_UNIT "sec",          (0, 0,0, 0,0, 1, 0,0,0,0,1,0)          // Time: base SI unit is seconds
PHYSICAL_UNIT "days",         (0,0,0,0,0,1,0,0,0,0,86400,0)          // Time: Days = 86400 seconds

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Bridge Sensor"

SELECTCASE "Physical Measurand", ID, 6
  CASE "Temperature (kelvin)", 0
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "K"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "K"
  ENDCASE
  CASE "Temperature (Celsius)", 1
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
  ENDCASE
  CASE "Strain", 2
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000p", "strain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000p", "strain"
  ENDCASE
  CASE "microstrain", 3
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
  ENDCASE
  CASE "Force/Weight (Newton)", 4
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000p", "N"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000p", "N"
  ENDCASE
  CASE "Force/Weight (pounds)", 5
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "lb"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000E+0", "lb"
  ENDCASE
  CASE "Force/Weight (kilogram-force/kilopond)", 6
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
  ENDCASE
  CASE "Acceleration (m/s^2)", 7
    %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s^2"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s^2"
ENDCASE
CASE "Acceleration (g)", 8
  %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
  %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
ENDCASE
CASE "Torque (Nm/radian)", 9
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
ENDCASE
CASE "Torque (Nm)", 10
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
ENDCASE
CASE "Torque (oz-in)", 11
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
ENDCASE
CASE "Pressure (Pascal)", 12
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
ENDCASE
CASE "Pressure (PSI)", 13
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
ENDCASE
CASE "Mass (kg)", 14
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
ENDCASE
CASE "Mass (g)", 15
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000p", "g"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000p", "g"
ENDCASE
CASE "Distance (m)", 16
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000p", "m"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000p", "m"
ENDCASE
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
CASE "Distance (mm)", 17
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
ENDCASE
CASE "Distance (inches)", 18
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
ENDCASE
CASE "Velocity (m/s)", 19
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
ENDCASE
CASE "Velocity (mph)", 20
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
ENDCASE
CASE "Velocity (fps)", 21
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
ENDCASE
CASE "Angular Position (radian)", 22
  %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
  %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
ENDCASE
CASE "Angular Position (degrees)", 23
  %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
  %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
ENDCASE
CASE "Rotational Velocity (radian/s)", 24
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
ENDCASE
CASE "Rotational Velocity (rpm)", 25
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
ENDCASE
CASE "Frequency", 26
  %MinPhysVal, "Minimum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  %MaxPhysVal, "Maximum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
ENDCASE
CASE "Concentration (gram/liter)", 27
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
ENDCASE
CASE "Concentration (kg/liter)", 28
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
ENDCASE
CASE "Molar Concentration (mole/m3)", 29
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
ENDCASE
CASE "Molar Concentration (mole/l)", 30
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
ENDCASE
CASE "Volumetric Concentration (m3/m3)", 31
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
ENDCASE
CASE "Volumetric Concentration (l/l)", 32
  %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
  %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
ENDCASE
CASE "Mass Flow", 33
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
ENDCASE
CASE "Volumetric Flow (m3/s)", 34
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
ENDCASE
CASE "Volumetric Flow (m3/hr)", 35
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
ENDCASE
CASE "Volumetric Flow (gpm)", 36
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
ENDCASE
CASE "Volumetric Flow (cfm)", 37
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
ENDCASE
CASE "Volumetric Flow (l/min)", 38
  %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
  %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
ENDCASE
CASE "Relative Humidity", 39
  %MinPhysVal, "Minimum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
  %MaxPhysVal, "Maximum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
ENDCASE
CASE "Ratio (percent)", 40
  %MinPhysVal, "Minimum Percentage", CAL, 32, SINGLE, "0.0000", "%"
  %MaxPhysVal, "Maximum Percentage", CAL, 32, SINGLE, "0.0000", "%"
ENDCASE
CASE "Voltage", 41
  %MinPhysVal, "Minimum Voltage", CAL, 32, SINGLE, "0.000p", "V"
  %MaxPhysVal, "Maximum Voltage", CAL, 32, SINGLE, "0.000p", "V"
ENDCASE
CASE "RMS Voltage", 42
  %MinPhysVal, "Minimum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
  %MaxPhysVal, "Maximum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
ENDCASE
CASE "Current", 43
  %MinPhysVal, "Minimum Current", CAL, 32, SINGLE, "0.000p", "A"
  %MaxPhysVal, "Maximum Current", CAL, 32, SINGLE, "0.000p", "A"
ENDCASE
CASE "RMS Current", 44
  %MinPhysVal, "Minimum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
  %MaxPhysVal, "Maximum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
ENDCASE
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  CASE "Power (Watts)", 45
    %MinPhysVal, "Minimum Power", CAL, 32, SINGLE, "0.000p", "W"
    %MaxPhysVal, "Maximum Power", CAL, 32, SINGLE, "0.000p", "W"
  ENDCASE
ENDSELECT

SELECTCASE "Full Scale Electrical Value Precision", CAL, 2
  CASE "mV/V", 0
    %MinElecVal, "Minimum Electrical Value", CAL, 11, ConRes, -1, .001, "0p", "V/V"
    %MaxElecVal, "Maximum Electrical Value", CAL, 11, ConRes, -1, .001, "0p", "V/V"
  ENDCASE
  CASE "uV/V", 1
    %MinElecVal, "Minimum Electrical Value", CAL, 19, ConRes, -0.00655, 0.000000025, "0p", "V/V"
    %MaxElecVal, "Maximum Electrical Value", CAL, 19, ConRes, -0.00655, 0.000000025, "0p", "V/V"
  ENDCASE
  CASE "Full precision", 2
    %MinElecVal, "Minimum Electrical Value", CAL, 32, SINGLE, "0.000p", "V/V"
    %MaxElecVal, "Maximum Electrical Value", CAL, 32, SINGLE, "0.000p", "V/V"
  ENDCASE
ENDSELECT

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)","Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

ENUMERATE BridgeTypeEnum, "Quarter", "Half", "Full"
%BridgeType, "Bridge type", ID, 2, BridgeTypeEnum,"e",""

%SensorImped, "Impedance of each bridge element", ID, 18, ConRes, 1, 0.1, "0.0", "Ohm"


%RespTime, "Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"

%ExciteAmplNom, "Excitation Level (Nominal)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
%ExciteAmplMin, "Excitation Level (Minimum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
%ExciteAmplMax, "Excitation Level (Maximum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT, "0", ""

ENDTEMPLATE

//-----------------------------------------------------------------------------------------------------------

TEMPLATE 0,8,34,"AC Linear/Rotary Variable Differential Transformer (LVDT/RVDT)"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 34 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default LVDT Template
ABSTRACT For AC Linear/Rotary Variable Differential Transforms (LVDT and RVDT)
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)

PHYSICAL_UNIT "m",       (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,0)         // Distance:  base SI unit is meter
PHYSICAL_UNIT "mm",      (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0.001,0)     // millimeter = 0.001 m
PHYSICAL_UNIT "in",      (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0.0254,0)    // inches = 0.0254 m
PHYSICAL_UNIT "radian",  (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,0)         // Angular position: base SI unit is radian
PHYSICAL_UNIT "degrees", (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0.0174533,0) // degrees = 0.0174533 radians
PHYSICAL_UNIT "V/V",     (1, 0, 0, 2, 1,-3,-1, 0, 0, 0, 1,0)         // Voltage ratio: [m2•kg/(sec3•A)]/[m2•kg/(sec3•A)]
PHYSICAL_UNIT "Ohm",     (0, 0, 0, 2, 1,-3,-2, 0, 0, 0, 1,0)         // Resistance: Ohms equals m2•kg/(s3•A2)
PHYSICAL_UNIT "V",       (0, 0, 0, 2, 1,-3,-1, 0, 0, 0, 1,0)         // Voltage: Volts equals m2•kg/(sec3•A)
PHYSICAL_UNIT "sec",     (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,0)         // Time: base SI unit is seconds
PHYSICAL_UNIT "Hz",      (0, 0, 0, 0, 0,-1, 0, 0, 0, 0, 1,0)         // Frequency: Hertz = 1/second
PHYSICAL_UNIT "days",    (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 86400,0)     // Time: Days = 86400 seconds

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "LVDT Sensor"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
SELECTCASE "Units of Displacement", ID, 3
  CASE "meters", 0
    %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000p", "m"
    %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000p", "m"
  ENDCASE
  CASE "millimeters", 1
    %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
    %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
  ENDCASE
  CASE "inches", 2
    %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
    %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
  ENDCASE
  CASE "Angular Position (radian)", 3
    %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
    %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
  ENDCASE
  CASE "Angular Position (degrees)", 4
    %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
    %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
  ENDCASE

ENDSELECT

SELECTCASE "Full Scale Electrical Value Precision", CAL, 1
  CASE "mV/V", 0
    %MinElecVal, "Minimum Electrical Value", CAL, 11, ConRes, -1, .001, "0p", "V/V"
    %MaxElecVal, "Maximum Electrical Value", CAL, 11, ConRes, -1, .001, "0p", "V/V"
  ENDCASE
  CASE "Full precision", 1
    %MinElecVal, "Minimum Electrical Value", CAL, 32, SINGLE, "0.000p", "V/V"
    %MaxElecVal, "Maximum Electrical Value", CAL, 32, SINGLE, "0.000p", "V/V"
  ENDCASE
ENDSELECT

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

%RespTime, "Transducer Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"

%ExciteAmplNom, "Excitation Level (Nominal)", ID, 8, ConRes, 0.1, 0.1, "0.0", "V"
%ExciteAmplMax, "Excitation Level (Maximum)", ID, 8, ConRes, 0.1, 0.1, "0.0", "V"
ENUMERATE ExciteTypeEnum, "DC", "Bipolar DC", "AC (rms)"
%ExciteType, "Excitation Voltage Type", ID, 0, ExciteTypeEnum,"e","" = "AC (rms)"

%ExciteFreqNom, "Excitation Frequency (Nominal)", ID, 12, ConRes, 1, 1, "0", "Hz"
%ExciteFreqMin, "Excitation Frequency (Minimum)", ID, 12, ConRes, 1, 1, "0", "Hz"
%ExciteFreqMax, "Excitation Frequency (Maximum)", ID, 12, ConRes, 1, 1, "0", "Hz"

%SensorImped, "LVDT input impedance @ nominal frequency", ID, 7, ConRelRes, 1, 0.043, "rp", "Ohm"

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID",  USR, 11, UNINT, "0", ""

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------------


TEMPLATE 0,8,35,"Strain Gage"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 35 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Strain Gage Template
ABSTRACT For strain gages used in a bridge circuit
SPACING



//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
PHYSICAL_UNIT "strain",     (1, 0,0, 1,0, 0, 0,0,0,0,1,0)          // Strain: equals meter/meter
PHYSICAL_UNIT "Ohm",        (0, 0,0, 2,1,-3,-2,0,0,0,1,0)          // Resistance: Ohms equals m^2•kg/(s^3•A^2)
PHYSICAL_UNIT "V",          (0, 0,0, 2,1,-3,-1,0,0,0,1,0)          // Voltage: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "V/V",        (1, 0,0, 2,1,-3,-1,0,0,0,1,0)          // Voltage ratio: [m^2•kg/(sec^3•A)]/[m^2•kg/(sec^3•A)]
PHYSICAL_UNIT "Pa",         (0, 0,0,-1,1,-2, 0,0,0,0,1,0)          // Pressure: Pascal equals Newton/m^2 = kg/(m•s^3)
PHYSICAL_UNIT "mm^2",       (0, 0,0, 2,0, 0, 0,0,0,0,1E-6,0)       // square millimeter = 1E-6 square meters
PHYSICAL_UNIT "A",          (0, 0,0, 0,0, 0, 1,0,0,0,1,0)          // Current: base SI unit is Ampere
PHYSICAL_UNIT "sec",        (0, 0,0, 0,0, 1, 0,0,0,0,1,0)          // Time: base SI unit is seconds
PHYSICAL_UNIT "days",       (0,0,0,0,0,1,0,0,0,0,86400,0)          // Time: Days = 86400 seconds

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Bridge Sensor"

%MinPhysVal, "Minimum Strain", CAL, 14, ConRes, -0.4, 50E-6, "0.000p", "strain"
%MaxPhysVal, "Maximum Strain", CAL, 14, ConRes, -0.4, 50E-6, "0.000p", "strain"

%MinElecVal, "Minimum Electrical Value", CAL, 14, ConRes, -1.0, 125E-6, "0.000p", "V/V"
%MaxElecVal, "Maximum Electrical Value", CAL, 14, ConRes, -1.0, 125E-6, "0.000p", "V/V"

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Bridge"

ENUMERATE GageTypeEnum, "Single element", "Two Poisson elements", "Two elements opposite sign", "Two elements same
sign", "Two element chevron", "Four element Poisson same sign", "Four element Poisson opposite sign", "Four uniaxial
elements", "Four element dual chevron", "Tee Rosette grid 1 (0°)", "Tee Rosette grid 2 (90°)", "Delta Rosette grid 1
(0°)", "Delta Rosette grid 2 (60°)", "Delta Rosette grid 3 (120°)", "Rectangular Rosette grid 1 (0°)", "Rectangular
Rosette grid 2 (45°)", "Rectangular Rosette grid 3 (90°)"
%GageType, "Gage Type", ID, 5, GageTypeEnum,"e",""

%GageFactor, "Gage Factor", CAL, 13, ConRelRes, 1.5, 0.000422, "rp", ""
%GageTransSens, "Transverse Sensitivity", CAL, 9, ConRes, -0.05, 0.0002, "0.0000", ""
%GageOffset, "Gage bridge offset", USR, 20, ConRes, -50E-3, 100E-9, "0.000p", "V/V"
%PoissonCoef, "Poisson Coefficient", USR, 14, ConRes, 0, 1E-4, "0.0000", ""
%YoungsMod, "Young's Modulus", USR, 14, ConRes, 0, 100E+6, "0.00p", "Pa"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
%GageArea, "Area of each gage element", ID, 7, ConRelRes, 0.2, 0.04, "r", "mm^2"

ENUMERATE BridgeTypeEnum, "Quarter", "Half", "Full"
%BridgeType, "Bridge circuit type", ID, 2, BridgeTypeEnum,"e",""
%SensorImped, "Resistance of each gage element", ID, 13, ConRelRes, 50, 0.000328, "rp", "Ohm"


%RespTime, "Transducer Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"

%ExciteAmplNom, "Excitation Level (Nominal)", ID, 8, ConRes, 0.1, 0.1, "0.0", "V"
%ExciteAmplMax, "Excitation Level (Maximum)", ID, 8, ConRes, 0.1, 0.1, "0.0", "V"

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT, "0", ""

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------


TEMPLATE 0,8,36,"Thermocouple"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 36 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Thermocouple Template
ABSTRACT For thermocouples and temperature sensors that provide a voltage output conforming to a standard
ABSTRACT thermocouple curve
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "°C",          (0, 0,0, 0,0, 0, 0,1,0,0,1,-273.15)   // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "V",           (0, 0,0, 2,1,-3,-1,0,0,0,1,0)         // Voltage: Volts equals m^2•kg/(sec^3•A)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
PHYSICAL_UNIT "sec",          (0, 0,0, 0,0, 1, 0,0,0,0,1,0)          // Time: base SI unit is seconds

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Voltage Sensor"

%MinPhysVal, "Minimum Temperature", CAL, 11, ConRes, -273, 1, "0", "°C"
%MaxPhysVal, "Maximum Temperature", CAL, 11, ConRes, -273, 1, "0", "°C"
%MinElecVal, "Minimum Electrical Value", CAL, 7, ConRes, -25e-3, 1e-3, "0p", "V"
%MaxElecVal, "Maximum Electrical Value", CAL, 7, ConRes, -25e-3, 1e-3, "0p", "V"

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Thermocouple"

ENUMERATE TCTypeEnum, "B","E","J","K","N","R","S","T","non-standard"
%TCType, "Thermocouple Type", ID, 4, TCTypeEnum, "e", ""

ENUMERATE CJSourceEnum, "CJC not provided by sensor","Sensor compensated for 0°C cold junction"
%CJSource, "Cold Junction Compensation", ID, 1, CJSourceEnum,"e",""

%SensorImped, "Output impedance of the sensor", ID, 12, ConRelRes, 1, 0.00155, "rp", "Ohm"

%RespTime, "Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT,"0", ""

ENDTEMPLATE

//-----------------------------------------------------------------------------------------------------
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
TEMPLATE 0,8,37,"Resistive Temperature Detector"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 37 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default RTD Sensor Template
ABSTRACT For Resistive Temperature Detector (RTD) sensors with the Callendar-Van Dusen behavior
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "°C",          (0,0,0,0,0,0,0,1,0,0,1,-273.15)       // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "Ohm",         (0,0,0,2,1,-3,-2,0,0,0,1,0)           // Resistance: Ohms equals m²•kg/(s³•A²)
PHYSICAL_UNIT "A",           (0,0,0,0,0,0,1,0,0,0,1,0)             // Current: base SI unit is Ampere
PHYSICAL_UNIT "sec",         (0,0,0,0,0,1,0,0,0,0,1,0)             // Time: base SI unit is seconds
PHYSICAL_UNIT "days",        (0,0,0,0,0,1,0,0,0,0,86400,0)         // Time: Days = 86400 seconds

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Resistance Sensor"

%MinPhysVal, "Minimum Temperature", CAL, 11, ConRes, -200, 1, "0", "°C"
%MaxPhysVal, "Maximum Temperature", CAL, 11, ConRes, -200, 1, "0", "°C"
%MinElecVal, "Minimum Electrical Value", CAL, 11, ConRes, 0, 1, "0", "Ohm"
%MaxElecVal, "Maximum Electrical Value", CAL, 13, ConRes, 0, 1, "0", "Ohm"

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "RTD"

SELECTCASE "R0 resistance", ID, 2
  CASE "Std. 100 Ohm", 0
    %RTDCoef_R0, "0 C resistance = 100 Ohm", ID, 0, Single, "000.0", "Ohm" = 100.0
  ENDCASE
  CASE "Std. 120 Ohm", 1
    %RTDCoef_R0, "0 C resistance = 120 Ohm", ID, 0, Single, "000.0", "Ohm" = 120.0
  ENDCASE
  CASE "Std. 1000 Ohm", 2
    %RTDCoef_R0, "0 C resistance = 1000 Ohm", ID, 0, Single, "000.0", "Ohm" = 1000.0
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
  ENDCASE
  CASE "Other Value of R0", 3
    %RTDCoef_R0, "0 C resistance", ID, 20, ConRelRes, 1, 0.0000045, "0.00p", "Ohm"
  ENDCASE
ENDSELECT

SELECTCASE "RTDCurve", ID, 3
  CASE "alpha=.00375", 0
    %RTDCoef_A, "Callendar-Van Dusen coefficient A", ID, 0, Single, "0.0000e-0", "1/C" = 3.8100e-3
    %RTDCoef_B, "Callendar-Van Dusen coefficient B", ID, 0, Single, "0.0000e-0", "1/C^2" = -6.0200e-7
    %RTDCoef_C, "Callendar-Van Dusen coefficient C", ID, 0, Single, "0.0000e-0", "1/C^3" = -6.000e-12
  ENDCASE
  CASE "alpha=.003851", 1
    %RTDCoef_A, "Callendar-Van Dusen coefficient A", ID, 0, Single, "0.0000e-0", "1/C" = 3.9083e-3
    %RTDCoef_B, "Callendar-Van Dusen coefficient B", ID, 0, Single, "0.0000e-0", "1/C^2" = -5.7750e-7
    %RTDCoef_C, "Callendar-Van Dusen coefficient C", ID, 0, Single, "0.0000e-0", "1/C^3" = -4.183e-12
  ENDCASE
  CASE "alpha=.003911", 2
    %RTDCoef_A, "Callendar-Van Dusen coefficient A", ID, 0, Single, "0.0000e-0", "1/C" = 3.9692e-3
    %RTDCoef_B, "Callendar-Van Dusen coefficient B", ID, 0, Single, "0.0000e-0", "1/C^2" = -5.8495e-7
    %RTDCoef_C, "Callendar-Van Dusen coefficient C", ID, 0, Single, "0.0000e-0", "1/C^3" = -4.229e-12
  ENDCASE
  CASE "alpha=.003916", 3
    %RTDCoef_A, "Callendar-Van Dusen coefficient A", ID, 0, Single, "0.0000e-0", "1/C" = 3.9739e-3
    %RTDCoef_B, "Callendar-Van Dusen coefficient B", ID, 0, Single, "0.0000e-0", "1/C^2" = -5.8700e-7
    %RTDCoef_C, "Callendar-Van Dusen coefficient C", ID, 0, Single, "0.0000e-0", "1/C^3" = -4.39e-12
  ENDCASE
  CASE "alpha=.003920", 4
    %RTDCoef_A, "Callendar-Van Dusen coefficient A", ID, 0, Single, "0.0000e-0", "1/C" = 3.9787e-3
    %RTDCoef_B, "Callendar-Van Dusen coefficient B", ID, 0, Single, "0.0000e-0", "1/C^2" = -5.8685e-7
    %RTDCoef_C, "Callendar-Van Dusen coefficient C", ID, 0, Single, "0.0000e-0", "1/C^3" = -4.160e-12
  ENDCASE
  CASE "alpha=.003928", 5
    %RTDCoef_A, "Callendar-Van Dusen coefficient A", ID, 0, Single, "0.0000e-0", "1/C" = 3.9888e-3
    %RTDCoef_B, "Callendar-Van Dusen coefficient B", ID, 0, Single, "0.0000e-0", "1/C^2" = -5.915e-7
    %RTDCoef_C, "Callendar-Van Dusen coefficient C", ID, 0, Single, "0.0000e-0", "1/C^3" = -3.816e-12
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  ENDCASE
  CASE "Other RTD curve (30-bit descriptor)", 6
    %RTDCoef_A, "Callendar-Van Dusen coefficient A", ID, 13, ConRes, 3.8E-3, 2.5E-8, "0.00000e-0", "1/C"
    %RTDCoef_B, "Callendar-Van Dusen coefficient B", ID, 10, ConRes, -6.1E-7, 5E-11, "000.00e-0", "1/C^2"
    %RTDCoef_C, "Callendar-Van Dusen coefficient C", ID, 7, ConRes, -6E-12, 2.3E-14, "0.00e-0", "1/C^3"
  ENDCASE
  CASE "Other RTD curve (96-bit descriptor)", 7
    %RTDCoef_A, "Callendar-Van Dusen coefficient A", ID, 32, SINGLE, "0.00000e-0", "1/C"
    %RTDCoef_B, "Callendar-Van Dusen coefficient B", ID, 32, SINGLE, "000.000e-0", "1/C^2"
    %RTDCoef_C, "Callendar-Van Dusen coefficient C", ID, 32, SINGLE, "0.000e-0", "1/C^3"
  ENDCASE
ENDSELECT

%RespTime, "Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"

%ExciteAmplNom, "Excitation Current (Nominal)", CAL, 8, ConRelRes, 1e-6, 0.02356427402545, "rp", "A"
%ExciteAmplMax, "Excitation Current (Maximum)", ID, 8, ConRelRes, 1e-6, 0.02356427402545, "rp", "A"

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT,"0", ""

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------



TEMPLATE 0,8,38,"Thermistor"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 38 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Thermistor Template
ABSTRACT For Thermistors following the Steinhart equation
SPACING
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "°C",          (0,0,0,0,0,0,0,1,0,0,1,-273.15)    // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "Ohm",         (0,0,0,2,1,-3,-2,0,0,0,1,0)        // Resistance: Ohms equals m^2•kg/(s^3•A^2)
PHYSICAL_UNIT "A",           (0,0,0,0,0,0,1,0,0,0,1,0)          // Current: base SI unit is Ampere
PHYSICAL_UNIT "W/°C",        (0,0,0,2,1,-3,0,-1,0,0,1,0)        // Thermal Conductivity: W/°C equals m^2•kg/(s^3•K)
PHYSICAL_UNIT "sec",         (0,0,0,0,0,1,0,0,0,0,1,0)          // Time: base SI unit is seconds
PHYSICAL_UNIT "days",        (0,0,0,0,0,1,0,0,0,0,86400,0)      // Time: Days = 86400 seconds

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Resistance Sensor"

%MinPhysVal, "Minimum Temperature", CAL, 11, ConRes, -200, 1, "0", "°C"
%MaxPhysVal, "Maximum Temperature", CAL, 11, ConRes, -200, 1, "0", "°C"
%MinElecVal, "Minimum Electrical Value", CAL, 18, ConRes, 0, 1, "0", "Ohm"
%MaxElecVal, "Maximum Electrical Value", CAL, 18, ConRes, 0, 1, "0", "Ohm"

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Thermistor"

%RTDCoef_R0, "0 C resistance", ID, 20, ConRelRes, 10, 0.0000063, "0.00p", "Ohm"

%SteinhartA, "Steinhart-Hart coefficient A", ID, 32, SINGLE, "0.0000p", "1/C"
%SteinhartB, "Steinhart-Hart coefficient B", ID, 32, SINGLE, "0.0000p", "1/C"
%SteinhartC, "Steinhart coefficient C", ID, 32, SINGLE, "0.0000p", "1/C"

%RespTime, "Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"

%ExciteAmplNom, "Excitation current (Nominal)", CAL, 8, ConRelRes, 1e-6, 0.02356427402545, "rp", "A"
%ExciteAmplMax, "Excitation current (Maximum)", ID, 8, ConRelRes, 1e-6, 0.02356427402545, "rp", "A"

%SelfHeating, "Self Heating Constant", ID, 5, ConRelRes, 0.00025, 0.0743491775, "rp", "W/°C"

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT,"0", ""

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------


TEMPLATE 0,8,39,"Potentiometric Voltage Divider"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 39 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Potentiometric Voltage Divider Template
ABSTRACT For potentiometric sensors used in voltage divider configuration
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Ampere, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "K",           (0, 0,0, 0,0, 0, 0,1,0,0,1,0)          // Temperature: base SI unit is kelvin
PHYSICAL_UNIT "°C",          (0, 0,0, 0,0, 0, 0,1,0,0,1,-273.15)    // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "strain",      (1, 0,0, 1,0, 0, 0,0,0,0,1,0)          // strain: equals meter/meter
PHYSICAL_UNIT "microstrain",(1, 0,0, 1,0, 0, 0,0,0,0,1e-6,0)       // microstrain = 10^(-6) x strain
PHYSICAL_UNIT "N",           (0, 0,0, 1,1,-2, 0,0,0,0,1,0)          // Force:  Newton equals m•kg/s^2
PHYSICAL_UNIT "lb",          (0, 0,0, 1,1,-2, 0,0,0,0,4.44822,0)    // pound is 4.44822 x Newton
PHYSICAL_UNIT "kgf",         (0, 0,0, 1,1,-2, 0,0,0,0,9.80665,0)    // kilogram-force (kilopond) is 9.80665 x Newton
PHYSICAL_UNIT "m/s^2",       (0, 0,0, 1,0,-2, 0,0,0,0,1,0)          // Acceleration: m/s^2
PHYSICAL_UNIT "ga",          (0, 0,0, 1,0,-2, 0,0,0,0,9.80665,0)    // ga = 9.80665 m/s^2
PHYSICAL_UNIT "Nm/radian",   (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Torque: N•m/rad equals m^2•kg/(s^2•rad)
PHYSICAL_UNIT "Nm",          (0,-1,0, 2,1,-2, 0,0,0,0,1,0)          // Common usage drops radian from abbreviation
PHYSICAL_UNIT "oz-in",       (0,-1,0, 2,1,-2, 0,0,0,0,0.00706155,0)// oz-in is 0.00706155•Nm
PHYSICAL_UNIT "Pa",          (0, 0,0,-1,1,-2, 0,0,0,0,1,0)          // Pressure: Pascal equals Newton/m^2 = kg/(m•s^2)
PHYSICAL_UNIT "PSI",         (0, 0,0,-1,1,-2, 0,0,0,0,6894.757,0)   // Pounds per Square Inch = 6894.757 Pa
PHYSICAL_UNIT "kg",          (0, 0,0, 0,1, 0, 0,0,0,0,1,0)          // Mass:  base SI unit is kilogram
PHYSICAL_UNIT "g",           (0, 0,0, 0,1, 0, 0,0,0,0,0.001,0)      // gram = 0.001 kg
PHYSICAL_UNIT "m",           (0, 0,0, 1,0, 0, 0,0,0,0,1,0)          // Distance:  base SI unit is meter
PHYSICAL_UNIT "mm",          (0, 0,0, 1,0, 0, 0,0,0,0,0.001,0)      // millimeter = 0.001 m
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
PHYSICAL_UNIT "in",          (0, 0,0, 1,0, 0, 0,0,0,0,0.0254,0)     // inches = 0.0254 m
PHYSICAL_UNIT "m/s",         (0, 0,0, 1,0,-1, 0,0,0,0,1,0)          // Velocity: Preferred SI unit is meter/second
PHYSICAL_UNIT "mph",         (0, 0,0, 1,0,-1, 0,0,0,0,0.44704,0)    // miles per hour = 0.44704 m/s
PHYSICAL_UNIT "fps",         (0, 0,0, 1,0,-1, 0,0,0,0,0.3048,0)     // feet per second = 0.3048 m/s
PHYSICAL_UNIT "radian",      (0, 1,0, 0,0, 0, 0,0,0,0,1,0)          // Angular position: base SI unit is radian
PHYSICAL_UNIT "degrees",     (0, 1,0, 0,0, 0, 0,0,0,0,0.0174533,0) // degrees = 0.0174533 radians
PHYSICAL_UNIT "radian/s",    (0, 1,0, 0,0,-1, 0,0,0,0,1,0)          // Angular velocity: preferred SI unit is radian/s
PHYSICAL_UNIT "rpm",         (0, 1,0, 0,0,-1, 0,0,0,0,0.104720,0)   // rev. per minute = 0.104720 radian/s
PHYSICAL_UNIT "Hz",          (0, 0,0, 0,0,-1, 0,0,0,0,1,0)          // Frequency: Hertz = 1/second
PHYSICAL_UNIT "kg/m^3",      (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // Density: kg/m^3
PHYSICAL_UNIT "g/l",         (0, 0,0,-3,1, 0, 0,0,0,0,1,0)          // gram per liter equals kg/m^3
PHYSICAL_UNIT "mole/m^3",    (0, 0,0,-3,0, 0, 0,0,1,0,1,0)          // Molar concentration: preferred SI unit is mole/m^3
PHYSICAL_UNIT "mole/l",      (0, 0,0,-3,0, 0, 0,0,1,0,1000,0)       // mole per liter = 1000 mole/m^3
PHYSICAL_UNIT "m^3/m^3",     (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // Volumetric concentration: base SI unit is m^3/m^3
PHYSICAL_UNIT "l/l",         (1, 0,0, 3,0, 0, 0,0,0,0,1,0)          // liter per liter equals m^3/m^3
PHYSICAL_UNIT "kg/s",        (0, 0,0, 0,1,-1, 0,0,0,0,1,0)          // Mass Flow:  preferred SI unit is kg/s
PHYSICAL_UNIT "m^3/s",       (0, 0,0, 3,0,-1, 0,0,0,0,1,0)          // Volumetric Flow: preferred SI unit is m^3/s
PHYSICAL_UNIT "m^3/hr",      (0, 0,0, 3,0,-1, 0,0,0,0,3600,0)       // m^3 per hour = 3600 m^3/s
PHYSICAL_UNIT "gpm",         (0, 0,0, 3,0,-1, 0,0,0,0,6.30902e-5,0) // gallons(US) per minute = 6.30902e-5 m^3/s
PHYSICAL_UNIT "cfm",         (0, 0,0, 3,0,-1, 0,0,0,0,4.71947e-4,0) // cubic feet per minute = 4.71947e-4 m^3/s
PHYSICAL_UNIT "l/min",       (0, 0,0, 3,0,-1, 0,0,0,0,1.66667e-5,0)// liters per minute = 1.66667e-5 m^3/s
PHYSICAL_UNIT "RH",          (1, 0,0,-3,1, 0, 0,0,0,0,1,0)          // Relative Humidity: = (kg/m^3)/(kg/m^3)
PHYSICAL_UNIT "%",           (1, 0,0, 0,0, 0, 0,0,0,0,100,0)        // Dimensionless Ratio: percent
PHYSICAL_UNIT "V/V",         (1, 0,0, 2,1,-3,-1,0,0,0,1,0)          // Voltage ratio: [m^2•kg/(sec^3•A)]/[m^2•kg/(sec^3•A)]
PHYSICAL_UNIT "Ohm",         (0, 0,0, 2,1,-3,-2,0,0,0,1,0)          // Resistance: Ohms equals m^2•kg/(s^3•A^2)
PHYSICAL_UNIT "V",           (0, 0,0, 2,1,-3,-1,0,0,0,1,0)          // Voltage: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "V rms",       (0, 0,0, 2,1,-3,-1,0,0,0,1,0)          // Voltage RMS: Volts equals m^2•kg/(sec^3•A)
PHYSICAL_UNIT "A",           (0, 0,0, 0,0, 0, 1,0,0,0,1,0)          // Current: base SI unit is Ampere
PHYSICAL_UNIT "A rms",       (0, 0,0, 0,0, 0, 1,0,0,0,1,0)          // Current RMS: base SI unit is Ampere
PHYSICAL_UNIT "W",           (0, 0,0, 2,1,-3, 0,0,0,0,1,0)          // Watts equals m^2•kg/(sec^3)
PHYSICAL_UNIT "sec",         (0, 0,0, 0,0, 1, 0,0,0,0,1,0)          // Time: base SI unit is seconds
PHYSICAL_UNIT "days",        (0,0,0,0,0,1,0,0,0,0,86400,0)          // Time: Days = 86400 seconds

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Potentiometric Voltage Divider
Sensor"

SELECTCASE "Physical Measurand", ID, 6
  CASE "Temperature (kelvin)", 0
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "K"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "K"
  ENDCASE
  CASE "Temperature (Celsius)", 1
    %MinPhysVal, "Minimum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
    %MaxPhysVal, "Maximum Temperature", CAL, 32, SINGLE, "0.000p", "°C"
  ENDCASE
  CASE "Strain", 2
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000p", "strain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000p", "strain"
  ENDCASE
  CASE "microstrain", 3
    %MinPhysVal, "Minimum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
    %MaxPhysVal, "Maximum Strain", CAL, 32, SINGLE, "0.000E+0", "microstrain"
  ENDCASE
  CASE "Force/Weight (Newton)", 4
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000p", "N"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000p", "N"
  ENDCASE
  CASE "Force/Weight (pounds)", 5
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "lb"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000E+0", "lb"
  ENDCASE
  CASE "Force/Weight (kilogram-force/kilopond)", 6
    %MinPhysVal, "Minimum Force/Weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
    %MaxPhysVal, "Maximum Force/weight", CAL, 32, SINGLE, "0.000E+0", "kgf"
  ENDCASE
  CASE "Acceleration (m/s²)", 7
    %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s²"
    %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "m/s²"
  ENDCASE
  CASE "Acceleration (g)", 8
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  %MinPhysVal, "Minimum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
  %MaxPhysVal, "Maximum Acceleration", CAL, 32, SINGLE, "0.000E+0", "ga"
ENDCASE
CASE "Torque (Nm/radian)", 9
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm/radian"
ENDCASE
CASE "Torque (Nm)", 10
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "Nm"
ENDCASE
CASE "Torque (oz-in)", 11
  %MinPhysVal, "Minimum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
  %MaxPhysVal, "Maximum Torque", CAL, 32, SINGLE, "0.000E+0", "oz-in"
ENDCASE
CASE "Pressure (Pascal)", 12
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000p", "Pa"
ENDCASE
CASE "Pressure (PSI)", 13
  %MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
  %MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"
ENDCASE
CASE "Mass (kg)", 14
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000E+0", "kg"
ENDCASE
CASE "Mass (g)", 15
  %MinPhysVal, "Minimum Mass", CAL, 32, SINGLE, "0.000p", "g"
  %MaxPhysVal, "Maximum Mass", CAL, 32, SINGLE, "0.000p", "g"
ENDCASE
CASE "Distance (m)", 16
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000p", "m"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000p", "m"
ENDCASE
CASE "Distance (mm)", 17
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "mm"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
ENDCASE
CASE "Distance (inches)", 18
  %MinPhysVal, "Minimum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
  %MaxPhysVal, "Maximum Distance", CAL, 32, SINGLE, "0.000E+0", "in"
ENDCASE
CASE "Velocity (m/s)", 19
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000p", "m/s"
ENDCASE
CASE "Velocity (mph)", 20
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "mph"
ENDCASE
CASE "Velocity (fps)", 21
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "fps"
ENDCASE
CASE "Angular Position (radian)", 22
  %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
  %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "radian"
ENDCASE
CASE "Angular Position (degrees)", 23
  %MinPhysVal, "Minimum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
  %MaxPhysVal, "Maximum Position", CAL, 32, SINGLE, "0.000E+0", "degrees"
ENDCASE
CASE "Rotational Velocity (radian/s)", 24
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "radian/s"
ENDCASE
CASE "Rotational Velocity (rpm)", 25
  %MinPhysVal, "Minimum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
  %MaxPhysVal, "Maximum Velocity", CAL, 32, SINGLE, "0.000E+0", "rpm"
ENDCASE
CASE "Frequency", 26
  %MinPhysVal, "Minimum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
  %MaxPhysVal, "Maximum Frequency", CAL, 32, SINGLE, "0.000p", "Hz"
ENDCASE
CASE "Concentration (gram/liter)", 27
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
    %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
    %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "g/l"
  ENDCASE
  CASE "Concentration (kg/liter)", 28
    %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
    %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "kg/m3"
  ENDCASE
  CASE "Molar Concentration (mole/m3)", 29
    %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
    %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/m3"
  ENDCASE
  CASE "Molar Concentration (mole/l)", 30
    %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
    %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "mole/l"
  ENDCASE
  CASE "Volumetric Concentration (m3/m3)", 31
    %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
    %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000E+0", "m3/m3"
  ENDCASE
  CASE "Volumetric Concentration (l/l)", 32
    %MinPhysVal, "Minimum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
    %MaxPhysVal, "Maximum Concentration", CAL, 32, SINGLE, "0.000p", "l/l"
  ENDCASE
  CASE "Mass Flow", 33
    %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
    %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "kg/s"
  ENDCASE
  CASE "Volumetric Flow (m3/s)", 34
    %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
    %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/s"
  ENDCASE
  CASE "Volumetric Flow (m3/hr)", 35
    %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
    %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "m3/hr"
  ENDCASE
  CASE "Volumetric Flow (gpm)", 36
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
    %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
    %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "gpm"
  ENDCASE
  CASE "Volumetric Flow (cfm)", 37
    %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
    %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000E+0", "cfm"
  ENDCASE
  CASE "Volumetric Flow (l/min)", 38
    %MinPhysVal, "Minimum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
    %MaxPhysVal, "Maximum Flow", CAL, 32, SINGLE, "0.000p", "l/min"
  ENDCASE
  CASE "Relative Humidity", 39
    %MinPhysVal, "Minimum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
    %MaxPhysVal, "Maximum Relative Humidity", CAL, 32, SINGLE, "0.0000", "RH"
  ENDCASE
  CASE "Ratio (percent)", 40
    %MinPhysVal, "Minimum Percentage", CAL, 32, SINGLE, "0.0000", "%"
    %MaxPhysVal, "Maximum Percentage", CAL, 32, SINGLE, "0.0000", "%"
  ENDCASE
  CASE "Voltage", 41
    %MinPhysVal, "Minimum Voltage", CAL, 32, SINGLE, "0.000p", "V"
    %MaxPhysVal, "Maximum Voltage", CAL, 32, SINGLE, "0.000p", "V"
  ENDCASE
  CASE "RMS Voltage", 42
    %MinPhysVal, "Minimum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
    %MaxPhysVal, "Maximum Voltage RMS", CAL, 32, SINGLE, "0.000p", "V rms"
  ENDCASE
  CASE "Current", 43
    %MinPhysVal, "Minimum Current", CAL, 32, SINGLE, "0.000p", "A"
    %MaxPhysVal, "Maximum Current", CAL, 32, SINGLE, "0.000p", "A"
  ENDCASE
  CASE "RMS Current", 44
    %MinPhysVal, "Minimum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
    %MaxPhysVal, "Maximum Current RMS", CAL, 32, SINGLE, "0.000p", "A rms"
  ENDCASE
  CASE "Power (Watts)", 45
    %MinPhysVal, "Minimum Power", CAL, 32, SINGLE, "0.000p", "W"
    %MaxPhysVal, "Maximum Power", CAL, 32, SINGLE, "0.000p", "W"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  ENDCASE
ENDSELECT

SELECTCASE "Full Scale Electrical Value Precision", CAL, 1
  CASE "Standard full scale", 0
    %MinElecVal, "Minimum Electrical Value", CAL, 0, SINGLE, "0", "V/V" = 0.0
    %MaxElecVal, "Maximum Electrical Value", CAL, 0, SINGLE, "0", "V/V" = 1.0
  ENDCASE
  CASE "Full precision", 1
    %MinElecVal, "Minimum Electrical Value", CAL, 20, CONRES, 0.0, 0.000001, "0.000p", "V/V"
    %MaxElecVal, "Maximum Electrical Value", CAL, 20, CONRES, 0.0, 0.000001, "0.000p", "V/V"
  ENDCASE
ENDSELECT

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

%SensorImped, "Sensor Impedance", ID, 12, ConRelRes, 1, 0.001706, "rp", "Ohm"

%RespTime, "Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"

%ExciteAmplNom, "Excitation Level (Nominal)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
%ExciteAmplMin, "Excitation Level (Minimum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
%ExciteAmplMax, "Excitation Level (Maximum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
ENUMERATE ExciteTypeEnum, "DC", "Bipolar DC", "AC (rms)"
%ExciteType, "Excitation Voltage Type", ID, 2, ExciteTypeEnum,"e",""

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID", USR, 11, UNINT, "0", ""

ENDTEMPLATE

//--------------------------------------------------------------------------
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
TEMPLATE 0,8,40,"Calibration Table"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 40 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Calibration Table Template
ABSTRACT Defines points of calibration as deviations from nominal
SPACING

PHYSICAL_UNIT "%",      (1,0,0,0,0,0,0,0,0,0,100,0)        // Dimensionless Ratio: percent

ENUMERATE CalSetEnum, "Electrical", "Physical"
%CalTable_Domain, "Domain parameter of the Calibration Table", CAL, 1, CalSetEnum, "e", ""

STRUCTARRAY CalTable, "Calibration table", CAL, 7
  %CalPoint_DomainValue, "Domain Calibration Point (% of full span)", CAL, 16, ConRes, 0, 0.00153, "0.00", "%"
  %CalPoint_RangeValue, "Range Calibration Deviation (% of full span)", CAL, 21, ConRes, -100, 0.0001, "0.0000", "%"
ENDSTRUCTARRAY

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------


TEMPLATE 0,8,41,"Calibration Curve"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 41 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Calibration Curve Template
ABSTRACT Defines transfer function deviation from nominal as a piecewise polynomial
SPACING

PHYSICAL_UNIT "%",      (1,0,0,0,0,0,0,0,0,0,100,0)        // Dimensionless Ratio: percent

ENUMERATE CalSetEnum, "Electrical", "Physical"
%CalCurve_Domain, "Domain parameter of the Calibration Curve", CAL, 1, CalSetEnum, "e", ""
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
STRUCTARRAY CalCurve, "Calibration curve segments", CAL, 8
  %CalCurve_PieceStart, "Starting domain value of segment (% of full span)", CAL, 13, ConRes, 0, 0.0123, "0.00", "%"
  STRUCTARRAY CalCurve_Poly, "Calibration curve polynomial", CAL, 7
    %CalCurve_Power, "Power of domain value", CAL, 7, ConRes, -32, 0.5, "0.0", ""
    %CalCurve_Coef, "Polynomial coefficient", CAL, 32, Single, "0.000E+0", ""
  ENDSTRUCTARRAY
ENDSTRUCTARRAY

ENDTEMPLATE

//-------------------------------------------------------------------------------------------------------------


TEMPLATE 0,8,42,"Frequency Response Table"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
  //the sensor to get the template ID, the 25 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Calibration Table Template
ABSTRACT Defines points of calibration as deviations from nominal
SPACING

PHYSICAL_UNIT "Hz",       (0, 0,0, 0,0,-1, 0,0,0,0,1,0)      // Frequency: Hertz = 1/second
PHYSICAL_UNIT "%",        (1,0,0,0,0,0,0,0,0,0,100,0)        // Dimensionless Ratio: percent

STRUCTARRAY TF_Table, "Calibration table", CAL, 7
  %TF_Table_Freq, "Frequency", CAL, 15, ConRelRes, 1, 0.000215, "0.00p", "Hz"
  %TF_Table_Ampl, "Amplitude", CAL, 21, ConRes, -100, 0.0001, "0.0000", "%"
ENDSTRUCTARRAY

ENDTEMPLATE

//-------------------------------------------------------------------------


TEMPLATE 0,8,43,"Charge Amplifier (optionally with Attached Force Transducer)"
  //The first 0 in the Template field indicates IEEE defined template, the 8 is the number of bits to read from
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

© IEEE 2004 – All rights reserved


```
  //the sensor to get the template ID, the 43 is the decimal value of this template ID.
TDL_VERSION_NUMBER 2  //Version 2 refers to the final IEEE 1451.4 version 1.0 TDL specification
ABSTRACT IEEE 1451.4 Default Charge Amplifier (with Attached Force Transducer) Template
ABSTRACT For charge amplifier devices, with attached piezoelectric force transducer
SPACING

//Physical Base Units: (ratio, radian, steradian, meter, kg, sec, Amp, kelvin, mole, candela, scaling, offset)
PHYSICAL_UNIT "Hz",          (0,0,0,0,0,-1,0,0,0,0,1,0)           // Frequency: Hertz = 1/second
PHYSICAL_UNIT "V/C",         (0,0,0,2,1,-2,0,0,0,0,1,0)           //Volt/Columb equals m^2*kg/(sec^2)
PHYSICAL_UNIT "V/N",         (0,0,0,1,0,-1,-1,0,0,0,1,0)          // Volts per Newton: V/(m•kg/s^2) = m/(s*A)
PHYSICAL_UNIT "Hz",          (0,0,0,0,0,-1,0,0,0,0,1,0)           // Frequency: Hertz = 1/second
PHYSICAL_UNIT "degrees",     (0,1,0,0,0,0,0,0,0,0,0.0174533,0)    // degrees = 0.0174533 radians
PHYSICAL_UNIT "°C",          (0,0,0,0,0,0,0,1,0,0,1,-273.15)      // Celsius is (kelvin - 273.15 K)
PHYSICAL_UNIT "%/decade",    (6,0,0,0,0,-1,0,0,0,0,0.01,0)        // Flatness over frequency: 0.01 / (log10(Hz/Hz))
PHYSICAL_UNIT "%/°C",        (0,0,0,0,0,0,0,-1,0,0,0.01,0)        // Temperature drift: %/°C = 0.01 x (1/Kelvin)
PHYSICAL_UNIT "N/m",         (0,0,0,0,1,-2, 0,0,0,0,1,0)          // Newton/m equals (m•kg/s^2)/m = kg/s^2
PHYSICAL_UNIT "days",        (0,0,0,0,0,1,0,0,0,0,86400,0)        // Time: Days = 86400 seconds

SELECTCASE "Attached Transducer", ID, 1

  CASE "Not specified", 0
    SELECTCASE "Extended Functionality", ID, 1
      CASE "None", 0
         UGID "I26-0-0-0", "Charge Amplifier"
        %Gain, "Gain", CAL, 12, ConRelRes, 5E7, 0.001,  "rp", "V/C"
        %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
        %TF_SP, "Low pass cut-off frequency (F lp)", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"
      ENDCASE
      CASE "Programmable Gain", 1
         UGID "I26-0-1-0", "Charge Amplifier, Programmable"
        %passive [Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
        %passive [CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "", "" = "11"
        %passive [ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
        %passive [FunctionType], "Passive control type", ID, 2, UNINT, "","" = 0//checkmark
        %passive [Function], "Passive mode", USR, 10, BitBin, "", "" =  "xx,00"

        %gain [Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

```
      %gain [CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "","" = "11"
      %gain [ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
      %gain [FunctionType], "Gain control type", ID, 2, UNINT, "","" = 1//One Exactly
      %gain [Function], %gain["10"], USR, 4, BitBin, "", "" = "10" //High gain
      %gain [Function], %gain["01"], USR, 4, BitBin, "", "" = "01" //Low gain

      %defaultFR, "Default setting", ID, 2, UNINT, "",""
      %Passive, "Supports multiplexer mode", ID, 1, UNINT, "", ""
      %Gain ["01"], "Low gain", CAL, 12, ConRelRes, 5E6, 0.001, "rp", "V/C"
      %Gain ["10"], "High gain", CAL, 12, ConRelRes, 5E7, 0.001, "rp", "V/C"
      %TF_HP_S ["01"], "Low-gain high pass cut-off frequency",  CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
      %TF_HP_S ["10"], "High-gain high pass cut-off frequency",  CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
      %TF_SP, "Low pass cut-off frequency", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"
    ENDCASE
  ENDSELECT
ENDCASE

CASE "Force Transducer", 1
  %Attached_MfgrID, "Manufacturer of attached transducer", CAL, 14, UNINT, "", ""
  %Attached_ModelNum, "Model number of attached transducer", CAL, 15, UNINT, "", ""
  %Attached_VersionLetter, "Version letter of attached transducer", CAL, 5, CHR5, "s", ""
  %Attached_VersionNum, "Version number of attached transducer", CAL, 6, UNINT, "", ""
  %Attached_SerialNum, "Serial number of attached transducer", CAL, 24, UNINT, "", ""
  SELECTCASE "Extended Functionality", ID, 1
    CASE "None", 0
       UGID "I26-0-0-0", "Charge Amplifier"
      %Sens@Ref, "Total sensitivity @ ref. condition", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/N"
      %TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
      %TF_SP, "Low pass cut-off frequency (F lp)", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"
    ENDCASE
    CASE "Programmable sensitivity", 1
       UGID "I26-0-1-0", "Charge Amplifier, Programmable"
      %passive [Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
      %passive [CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "","" = "11"
      %passive [ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
      %passive [FunctionType], "Passive control type", ID, 2, UNINT, "","" = 0//checkmark
      %passive [Function], "Passive mode", USR, 10, BitBin, "", "" =  "xx,00"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
        %sens [Initialize], "Initialize not needed", ID, 1, UNINT, "","" = 0
        %sens [CtrlFunctionMask], "Control Function Mask",ID, 4, BitBin, "","" = "11"
        %sens [ReadWrite], "Write only", ID, 2, UNINT, "","" = 3
        %sens [FunctionType], "Sensitivity control type", ID, 2, UNINT, "","" = 1//One Exactly
        %sens [Function], %Sens@Ref["10"], USR, 4, BitBin, "", "" = "10" //High sensitivity
        %sens [Function], %Sens@Ref["01"], USR, 4, BitBin, "", "" = "01" //Low sensitivity

        %defaultFR, "Default setting", ID, 2, UNINT, "",""
        %Passive, "Supports multiplexer mode", ID, 1, UNINT, "", ""
        %Sens@Ref ["01"], "Low sensitivity @ Fref. (total)", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/N"
        %Sens@Ref ["10"], "High sensitivity @ Fref. (total)", CAL, 16, ConRelRes, 5E-7, 0.00015, "rp", "V/N"
        %TF_HP_S ["01"], "Low sensitivity high pass cut-off frequency", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
        %TF_HP_S ["10"], "High sensitivity high pass cut-off frequency", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"
        %TF_SP, "Low pass cut-off frequency (F lp)", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"

      ENDCASE
    ENDSELECT

    ENUMERATE DirectionEnum,"x","y","z"
    %Direction, "Sensitivity direction (x,y,z)", CAL, 2, DirectionEnum, "e", ""

    SELECTCASE "Transfer Function", ID, 1
      CASE "Not specified", 0
      ENDCASE
      CASE "Specified", 1
        %TF_KPr, "Resonance frequency (F res)", CAL, 9, ConRelRes, 100, 0.01, "rp", "Hz"
        %TF_KPq, "Quality factor @ F res (Q)", CAL, 9, ConRelRes, 0.4, 0.01,  "rp", ""
        %TF_SL, "Amplitude slope (a)", CAL, 7, ConRes, -6.3, 0.1, "0.0", "%/decade"
        %TempCoef, "Temperature coefficient (b)", CAL, 6, ConRes, -0.8, 0.025,"0.000","%/°C"
        %Stiffness, "Stiffness of transducer", CAL, 6, ConRelRes, 1E6, 0.10, "rp", "N/m"
        %Mass_below, "Mass below gage", CAL, 6, ConRelRes, 0.1, 0.1, "rp", "g"
      ENDCASE
    ENDSELECT
  ENDCASE

ENDSELECT
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
ENUMERATE SignEnum,"Positive","Negative"
%Sign, "Polarity (Sign)", CAL, 1, SignEnum, "e", ""

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse
1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"
%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge Sensor","LVDT
Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Electrical Signal Type", ID, 0, ElecSigTypeEnum,"e","" = "Voltage Sensor"

ENUMERATE ACDCCouplingEnum,"DC","AC"
%ACDCCoupling, "AC or DC Coupling", ID, 0, ACDCCouplingEnum,"e","" = "AC"

%PhaseCorrection, "Phase correction @ F ref", CAL, 6, CONRES, -3.2, 0.1, "rp", "degrees"
%Reffreq, "Reference frequency (F ref)", CAL, 8, ConRelRes, 0.35, 0.0175, "0p","Hz"
%RefTemp, "Reference temperature (T ref)", CAL, 5, ConRes, 15, 0.5, "0.0","°C"

%CalDate, "Calibration Date", CAL, 16, DATE,"d-mmm-yyyy",""
%CalInitials, "Calibration Initials", CAL, 15, CHR5,"s",""
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"

%MeasID, "Measurement location ID",  USR, 11, UNINT,"0", ""

ENDTEMPLATE

//-----------------------------------------------------------------------------------------------------

Validation_Keycode xxx
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Annex B

(normative)

Property definitions

B.1 Reserved properties

This annex contains the listing and definitions of properties used in templates. Table B.1 includes a listing of all parameters, organized into functional groups.

Table B.1—Reserved properties

Properties		Description
Sensitivity and mapping properties		
General	%Sens	Sensitivity of transducer
	%Sens@Ref	Sensitivity of transducer at reference conditions
	%Reffreq	Reference frequency (f ref)
	%RefTemp	Reference temperature (T ref)
	%Sign	Phase inversion (0° or 180°)
	%Direction	Direction, or axis, of sensitivity (x, y, or z)
	%MapMeth	Mapping Method of physical to electrical units
	%MinPhysVal	Minimum value of physical measurement/control range
	%MaxPhysVal	Maximum value of physical measurement/control range
	%MinElecVal	Minimum value of electrical signal range
	%MaxElecVal	Maximum value of electrical signal range
Strain/Bridge	%GageType	Topology and rosette orientation of gage
	%BridgeType	Type of bridge (quarter, half, or full)
	%GageFactor	Sensitivity of strain gage
	%GageTransSens	Transverse sensitivity of strain gage
	%GageOffset	Zero offset of gage circuit after installation
	%PoissonCoef	Poisson Coefficient of strain gage
	%YoungsMod	Youngs Modulus of material to which gage is attached
	%GageArea	Area of gage element

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.1—Reserved properties *(continued)*

Properties		Description
Sensitivity and mapping properties		
RTD and thermistor	%RTDCoef_R0	RTD or thermistor resistance at 0 °C
	%RTDCoef_A	Coefficient A of Callendar Van-Dusen equation for RTDs
	%RTDCoef_B	Coefficient B of Callendar Van-Dusen equation for RTDs
	%RTDCoef_C	Coefficient C of Callendar Van-Dusen equation for RTDs
	%SteinhartA	Coefficient A of Steinhart-Hart equation for thermistors
	%SteinhartB	Coefficient B of Steinhart-Hart equation for thermistors
	%SteinhartC	Coefficient C of Steinhart-Hart equation for thermistors
	%SelfHeating	Coefficient of self-heating, intended for thermistors
TC	%TCType	Thermocouple calibration type (J, K, T, etc.)
	%CJSource	Cold-junction compensation method
Electrical signal properties		
General	%ElecSigType	Type of electrical signal (enumerated)
	%RespTime	Response Time
	%ACDCCoupling	Coupling of electrical signal (AC or DC)
	%SensorImped	Electrical impedance of sensor (of each element in case of bridge)
	%DiscSigType	Discrete signal type
	%DiscSigAmpl	Discrete signal voltage amplitude
	%PulseMeasType	Pulse signal measurement type (frequency, period, count, etc.)
	%Gain	Gain of preamplifier
	%Filter	Indicates selectable filter
	%TempCoef	Temperature coefficient

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.1—Reserved properties *(continued)*

Properties		Description
Sensitivity and mapping properties		
Mic/Preamp	%Prepolarized	Prepolarized (yes or no)
	%RefPol	Polarization voltage
	%Rin	Input resistance of amplifier
	%Rout	Output resistance of amplifier
	%Cin	Input capacitance of amplifier
	%Cmic	Microphone capacitance
	%Cstray	Microphone stray capacitance
	%Rleakage	Microphone leakage resistance
	%MicType	Microphone type
	%MicSize	Microphone size
	%Resp_Type	Frequency response type
	%RefPress	Reference pressure
	%Equi_Vol	Equivalent microphone volume
	%Gate	Gate present
Excitation and power	%ExciteAmplNom	Excitation or power-supply level, nominal
	%ExciteAmplMin	Excitation or power-supply level, minimum
	%ExciteAmplMax	Excitation or power-supply level, maximum
	%ExciteType	Type of excitation or power (DC, AC, or bipolar DC)
	%ExciteCurrentDraw	Maximum current required to power/excite transducer
	%ExciteFreqNom	Excitation signal frequency, nominal
	%ExciteFreqMin	Excitation signal frequency, minimum
	%ExciteFreqMax	Excitation signal frequency, maximum
	%LoopSupplyMin	Supply for current loop transducers, minimum
	%LoopSupplyMax	Supply for current loop transducers, maximum
Calibration properties		
Mgt.	%CalDate	The date of the last calibration
	%CalInitials	Calibration initials
	%CalPeriod	Amount of time recommended between calibrations

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.1—Reserved properties *(continued)*

Properties		Description
Sensitivity and mapping properties		
Calibration table and curves	%CalTable_Domain	Indicates calibration table domain as electrical or physical
	%CalPoint_DomainValue	Domain calibration value
	%CalPoint_RangeValue	Range calibration deviation
	%CalCurve_Domain	Indicates calibration curve domain as electrical or physical
	%CalCurve_PieceStart	Start of calibration curve segment
	%CalCurve_Power	Power of domain value
	%CalCurve_Coef	Coefficient of polynomial
Transfer function	%TF_SZ	Single zero
	%TF_SP	Single pole (low-pass filter of first order)
	%TF_KZr	Complex zero
	%TF_KZq	Quality factor parameter Qz of a complex zero
	%TF_KPr	Complex pole at Fpres (F mounted resonance)
	%TF_KPq	Quality factor Qp for the complex pole (mounted quality factor)
	%TF_HP_S	Single zero at 0 and a single pole (high-pass filter)
	%TF_SL	Constant relative slope
	%TF_SZm	Single zero dependent on previous property
	%TF_SPm	Single pole dependent on previous property
	%PhaseCorrection	Phase correction at the reference condition
	%TF_Table_Freq	Frequency point value for tabular transfer function
	%TF_Table_Ampl	Amplitude point value for tabular transfer function
Miscellaneous properties		
Attached transducer	%Attached_MfgrID	Manufacturer ID of transducer attached to amplifier
	%Attached_ModelNum	Model number of transducer attached to amplifier
	%Attached_VersionLetter	Version letter of transducer attached to amplifier
	%Attached_VersionNum	Version number of transducer attached to amplifier
	%Attached_SerialNum	Serial number of transducer attached to amplifier
	%System_MfgrID	Manufacturer ID of system
	%System_ModelNum	Model number of system
	%System_VersionLetter	Version letter of system
	%System_VersionNum	Version number of system
	%System_SerialNum	Serial number of system

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.1—Reserved properties *(continued)*

Properties		Description
Sensitivity and mapping properties		
Miscellaneous	%Stiffness	Stiffness of transducer
	%Mass_below	Mass below gage
	%Weight	Weight of transducer
	%TestGain	Test gain
	%Passive	Indicates support of passive mode
	%PollFreq	The frequency with which the host shall update the FR
	%MeasID	Measurand ID
	%Ccable	Capacitance of cable
	%CableLen	Length of cable
	%Appended_TEDS	Indicates if an Appended TEDS exists
	%Appended_TEDS_location	Indicates location of the Appended TEDS if it exists
	%EMBTPL	Indicates that the next portion of the TEDS is in Embedded Template format
	%DefaultFR	Defines the default setting of the FR. Cannot coexist with subproperty Default.
	%XML	XML format
	%MDEF	Prefix for manufacturer-defined parameters
	%TDL_CHKSUM	User template validation checksum
	%user	Freeform TEDS format
Grouping properties		
	%PhysicalParameterType	Describes the parameter type
	%MemberIndex	Indicate the order in which XdcrChannels are grouped within a PublicXdcr

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

B.2 Grouping properties

These grouping signal parameters are used to describe the grouping of the transducer. The information in these parameters can be used by a measurement system in configuring signal conditioning and grouping XdcrChannels. In addition to these parameters, the parameters Minimum Electrical Value and Maximum Electrical Value listed under the Mapping Parameters heading are used to describe the useful range of the transducer.

B.2.1 Group type

Property name: **GroupType**

See Table B.2.

© IEEE 2004 – All rights reserved

Table B.2—GroupTypedescriptions

GroupTypevalue	PublicXdcr:Type enumeration
0	SCALAR
1	VECTOR
2	NULL

B.2.2 CoordinateSystem

Property name: **CoordinateSystem**

See Table B.3.

Table B.3—CoordinateSystem enumeration

CoordinateSystem value	ParameterType:MetaData: CoordinateSystem
0	CC_ARBITRARY
1	CC_CARTESIAN
2	CC_CYLINDRICAL
3	CC_SPHERICAL
4	CC_PLANETARY
5	CC_ORIENTATION

B.2.3 Member Index

Property name: **MemberIndex**

This property shall indicate the order in which XdcrChannels are grouped within a PublicXdcr. For example, a XdcrChannel containing the MemberIndex property in its property list with a value of 0 will be the first element of a vector given that a GroupType property has a value of VECTOR. A second XdcrChannel with MemberIndex value of 1 will be the second element, etc.

B.3 Sensitivity and mapping properties

Sensitivity and mapping properties specify the general mapping of the physical to electrical (or electrical to physical) operation performed within the transducer. Generally, a system will use this information to scale from electrical signal units to engineering units, and vice versa.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

B.3.1 General

B.3.1.1 Transducer sensitivity

Property name: **Sens**

This property is the sensitivity of the transducer, expressed as the output parameter per input physical unit; also used with subproperty `[Function]` to select sensitivity.

Example:

%Sens, “Transducer Sensitivity”, CAL, 32, SINGLE, “0.000p”, “V/(m/s^2)”

B.3.1.2 Sensitivity at reference conditions

Property name: **Sens@Ref**

This property gives the magnitude of the complex sensitivity of the transducer at reference conditions. The unit is the output parameter per input physical unit (e.g., m/s^2, Pa). The reference conditions are the chosen parameters and environmental conditions, e.g., for accelerometers calibrated by primary means given in ISO 16063-11:1999 [B7] as:

When used with accelerometers, recommended reference conditions are as follows:

— Frequency in hertz: 160, 80, 40, 16, or 8 (or radian frequency ω = 1000, 500, 250, 100, or 50 radians /s)

— Acceleration in meters per second squared (acceleration amplitude or r.m.s. value): 100, 50, 20, 10, 5, 2, or 1

— Room temperature: (23 ± 3) ^{o}C

— Relative humidity: Max. 75%

A selection of these conditions may be included in the TEDS information. If not, the manufacturer’s given data can be consulted.

Example:

%Sens@Ref, “Sensitivity @ F ref”, CAL, 16, CONRELRES, 100E-6, 0.0001, “0.000p”, “V/(m/s^2)”

B.3.1.3 Reference frequency

Property name: **Reffreq**

This property gives the frequency in Hz or radians at which the Sens@Ref property was measured.

Example:

%Reffreq, "F ref", CAL, 8, ConRelRes, 10.17501895022, 0.015, "0p","Hz"

B.3.1.4 Reference temperature

Property name: **RefTemp**

This property gives the temperature in °C at which the Sens@Ref property was measured.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Example:

%RefTemp, “Reference temperature (T ref)”, CAL, 5, ConRes, 15, 0.5, “0.0”,”°C”

B.3.1.5 Phase sign

Property name: **Sign**

This property gives the sign (i.e., + for in phase or – for 180° out of phase) of the output signal with respect to the input signal with both properly defined by the manufacturer. For an amplifier, this is also used to indicate inverting or non-inverting operation.

This property shall use an enumerated data type: `SignEnum`, with the values shown in Table B.4:

Table B.4—Sign property description

%Sign Value	Enumeration index	Description
"Positive"	0	Positive or 0°
"Negative"	1	Negative or 180°

NOTE— Additional smaller phase shifts may be given as a phase shift at the reference or as a part of a transfer function definition.

Example:

ENUMERATE SignEnum,"Positive","Negative"
%Sign, "Polarity (Sign)", CAL, 1, SignEnum, "e", ""

B.3.1.6 Axis of sensitivity

Property name: **Direction**

This property gives the direction of sensitivity (i.e., x, y, or z). This property shall be an enumerated data type with the values shown in Table B.5:

Table B.5—Direction property description

%Direction Value	Enumeration index	Description
"x"	0	x-axis
"y"	1	y-axis
"z"	2	z-axis
"n/a"	3	n/a

Example:

ENUMERATE DirectionEnum,"x","y","z","n/a"
%Direction, "Sensitivity direction (x,y,z)", CAL, 2, DirectionEnum, "e", ""

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

B.3.1.7 Mapping Method

Property name: **MapMeth**

The Mapping Method is an enumerated list describing the mathematical relationship between the physical signal and the electrical signal of the transducer. The MapMeth parameter shall be an enumerated data type with the values shown in Table B.6:

Table B.6—MapMeth property description

%MapMeth Value	Enumeration index	Description
″Linear″	0	Linear: (Phys Value) = m*(Electrical Value) + b
″Inverse m/(x+b)″	1	Inverse I: (Phys Value) = m/[(Electrical Value) + b]
″Inverse b+m/x″	2	Inverse II: (Phys Value) = b + m/(Electrical Value)
″Inverse 1/(b+m/x)″	3	Inverse III: (Phys Value)$^{-1}$ = b + m/ (Electrical Value)
″Thermocouple″	4	Standard Thermocouple temperature/voltage mapping
″Thermistor″	5	Callandar Van-Dusen (RTD) temperature/resistance mapping
″RTD″	6	Steinhart (thermistor) temperature/resistance mapping
″Bridge″	7	Bridge mapping (using strain-gage properties)

Example:

ENUMERATE MapMethEnum, "Linear","Inverse m/(x+b)","Inverse (b+m/x)", "Inverse 1/(b+m/x)","Thermocouple","Thermistor","RTD","Bridge"

%MapMeth, "Mapping Method", ID, 0, MapMethEnum,"e","" = "Linear"

B.3.1.8 Minimum and Maximum Physical Values

Property name: **MinPhysVal, MaxPhysVal**

Together, these parameters describe the limits of the useful or specified operating range of the transducer. When used with transducers having linear or inverse Mapping Methods, these parameters may be used with the Minimum and Maximum Electrical Values to define the two points needed to represent the calibrated mathematical relationship between the physical and electrical signals of the transducer. In addition, the units assigned to these parameters indicate the specific measurand for the sensor.

Example:

%MinPhysVal, "Minimum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"

%MaxPhysVal, "Maximum Pressure", CAL, 32, SINGLE, "0.000E+0", "PSI"

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

B.3.1.9 Minimum and Maximum Electrical Values

Property name: **MinElecVal, MaxElecVal**

Together, these parameters describe the limits of the electrical signal. For sensors, these parameters define the range of the electrical output signal (i.e., –10 to 10 V). When used with transducers having linear or inverse Mapping Methods, these parameters may be used with the Minimum and Maximum Physical Values to define the two points needed to represent the calibrated mathematical relationship between the physical and electrical signals of the transducer. When used in templates having alternate descriptions of specifying the electrical-to-physical calibration, such as strain gage, thermocouple, RTD, and thermistor templates, then the MinElecVal and MaxElecVal are intended primarily as a convenient way to indicate the magnitude of the electrical signal for proper configuration of external signal conditioning and measurement equipment.

Example:

```
%MinElecVal, "Minimum Electrical Value", CAL, 11, ConRes, -20.48, 0.02, "0.00", "V"
%MaxElecVal, "Maximum Electrical Value", CAL, 11, ConRes, -20.48, 0.02, "0.00", "V"
```

B.3.2 Bridge mapping parameters

A bridge circuit maps a physical measurement (such as strain or force) to an electrical signal that is proportional to an electrical excitation. Describing a bridge circuit requires information about the bridge type and, in the case of strain gages, the bridge topology. The Bridge Type property defines the form of the electrical circuit—either quarter, half, or full bridge—and indicates what, if any, bridge completion needs to be provided by the measuring system. The Gage Type property describes the bridge topology by indicating which of the arms of the bridge are active elements and by describing their alignment with the principal strain axis. The Gage Type property additionally defines the relative orientation of individual strain-gage sensors in a rosette configuration where multiple axes are measured with multiple strain-gage sensors. Additional parameters such as Gage Factor and Gage offset are also available to fully describe the sensor.

B.3.2.1 Gage Type

Property name: **GageType**

This parameter describes which of the resistive elements of a strain-gage bridge are active elements; their orientation with the principal strain axis; and, in the case of a multi-axial rosette gage, the relative orientations of the individual gages. This information, along with the Gage Factor and possibly Poisson ratio parameters, describe the mathematical relationship between the strain being measured and the electrical signal when the Mapping Method is equal to 6 (bridge mapping). Although these bridge topologies may result in Mapping Methods that could be describable by either Mapping Method 0 (linear) or Mapping Method 3 (inverse III), the use of this explicit method of describing strain-gage topologies is more aligned with traditional methods of describing a strain gage.

The first nine of these properties describe a single sensor measuring strain along a single axis. The various options allow for different ways of mounting one or more strain-gage elements and configuring them in a bridge to comprise the total strain-gage sensor. The remaining eight properties are used to describe rosette configurations. In a rosette configuration, each strain-gage element of the rosette is independently measured to read the strain in a particular axial orientation; therefore, each strain gage of the rosette is considered to be its own sensor with its own TEDS. The TEDS of each sensor in the rosette will have a complementary value assigned to the GageType property. For example, in a Delta rosette there will be three sensors—the TEDS of one will have a GageType of Delta, 0°, another will be assigned Delta, 60°, and the third will be assigned Delta, 120°.

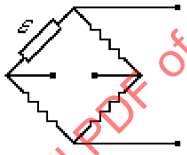
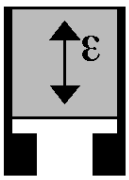
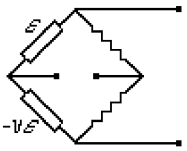
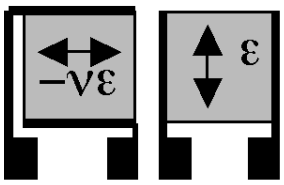
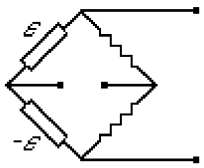
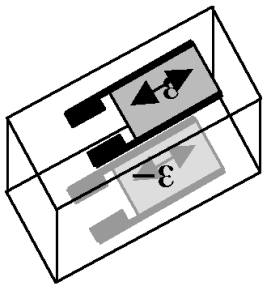
IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

The Gage Types and the resulting mapping functions from strain to mV/V are depicted in Table B.7. The active elements oriented with the principal axis are shown with a $+\varepsilon$ or a $-\varepsilon$, and the "transverse" or "compensating" elements that vary by the Poisson ratio are shown with a $+\nu\varepsilon$ or a $-\nu\varepsilon$. In these equations, the physical value (strain) being measured is expressed as ε, the measured electrical signal is expressed as V_r, (a ratio of a measured voltage to an excitation voltage), the Gage Factor = F, and Poisson's ratio = ν. Since the ordering (numbering or lettering) of the elements in a rosette is critical in establishing their relative orientations, the table also shows how the rosette orientations are related to each other. Notice that, in keeping with the practice of denoting counterclockwise rotation as increasing angular measurement, the numbering of these rosettes increases with counterclockwise rotation.

Gage Type is an enumerated data type with the defined values shown in Table B.7:

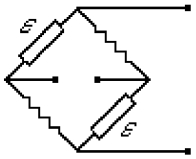
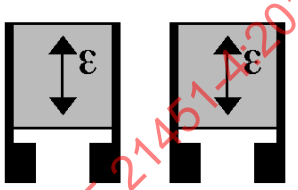
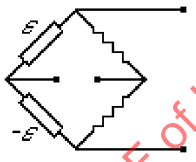
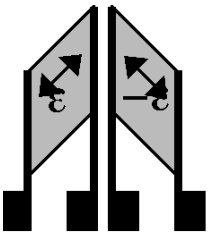
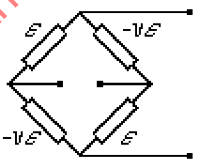
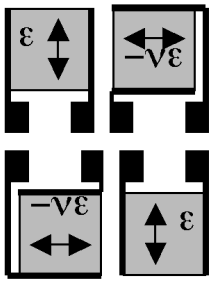
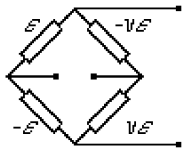
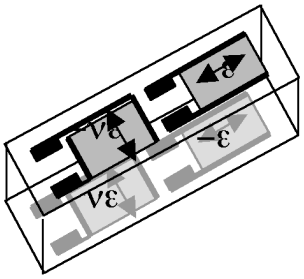
Table B.7—GageType property description

%GageType Value	Enum. index	Description		
Uniaxial (single sensor) Gage Types				
″Single element″	0	Single element gage	 $\varepsilon = \frac{4V_r}{F[1-2V_r]}$	
″Two Poisson element″	1	Two elements with Poisson arrangement	 $\varepsilon = \frac{4V_r}{F[(1+\nu)-2V_r(1-\nu)]}$	
″Two elements, opposite sign″	2	Two elements, opposite sign (adjacent arms)	 $\varepsilon = \frac{2V_r}{F}$	

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

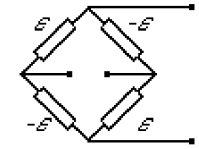
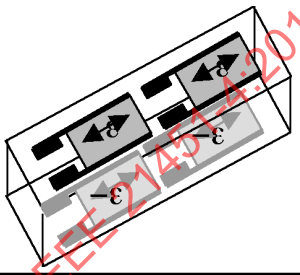
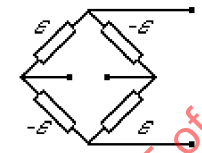
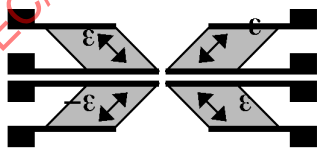
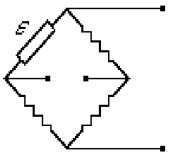
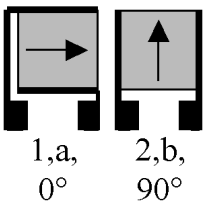
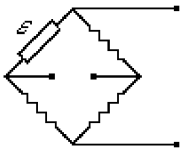
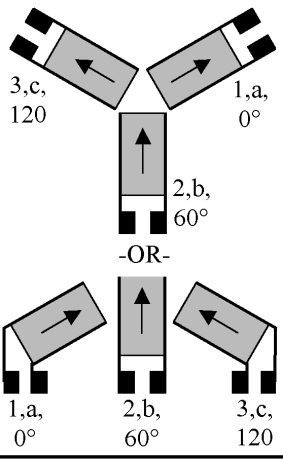
Table B.7—GageType property description *(continued)*

%GageType Value	Enum. index	Description		
Uniaxial (single sensor) Gage Types				
″Two elements, same sign″	3	Two elements, same sign (opposite arms)	 $\varepsilon = \frac{2V_r}{F[1-V_r]}$	
″Two elements, chevron orientation″	4	Two elements, 45° Chevron (torque or shear) arrangement	 $\varepsilon = \frac{2V_r}{F}$	
″Four Poisson elements, same sign″	5	Four elements, Poisson strains of same sign in opposite arms	 $\varepsilon = \frac{2V_r}{F[(1+v)-V_r(1-v)]}$	
″Four Poisson elements, opposite sign″	6	Four elements, Poisson strains of opposite sign in adjacent arms	 $\varepsilon = \frac{2V_r}{F[1+v]}$	

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

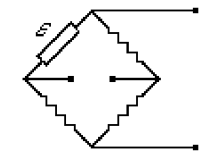
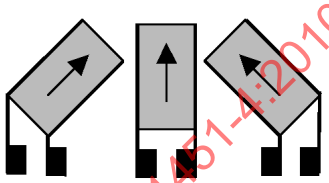
Table B.7—GageType property description *(continued)*

%GageType Value	Enum. index	Description		
Uniaxial (single sensor) Gage Types				
″Four uniaxial elements″	7	Four elements, equal strains of opposite sign in adjacent arms	 $\varepsilon = \frac{V_r}{F}$	
″Four elements, dual chevron″	8	Four elements, 45° Chevron (torque or shear) arrangement	 $\varepsilon = \frac{V_r}{F}$	
Multi-axial (multi-sensor Rosette) Gage Types				
″Tee Rosette grid 1″	9	Tee Rosette grid 1 or a (0°)	 $\varepsilon = \frac{4V_r}{F[1-2V_r]}$	 1,a, 0° 2,b, 90°
″Tee Rosette grid 2″	10	Tee Rosette grid 2 or b (90°)		
″Delta Rosette grid 1″	11	Delta Rosette grid 1 or a (0°)	 $\varepsilon = \frac{4V_r}{F[1-2V_r]}$	 3,c, 120 1,a, 0° 2,b, 60° -OR- 1,a, 0° 2,b, 60° 3,c, 120
″Delta Rosette grid 2″	12	Delta Rosette grid 2 or b (60°)		
″Delta Rosette grid 3″	13	Delta Rosette grid 3 or c (120°)		

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.7—GageType property description *(continued)*

%GageType Value	Enum. index	Description		
Uniaxial (single sensor) Gage Types				
″Rectangular Rosette grid 1″	14	Rectangular Rosette grid 1 or a (0°)	 $\varepsilon = \frac{4V_r}{F[1-2V_r]}$	 1,a, 0° 2,b, 45° 3,c, 90°
″Rectangular Rosette grid 2″	15	Rectangular Rosette grid 2 or a (45°)		
″Rectangular Rosette grid 3″	16	Rectangular Rosette grid 3 or a (90°)		

Example:

ENUMERATE GageTypeEnum, "single element", "two Poisson element", "two elements, opposite sign", "two elements, same sign", "two elements, chevron orientation", "four Poisson elements, same sign", "four Poisson elements, opposite sign", "four uniaxial elements", "four elements, dual chevron", "Tee rosette grid 1", "Tee rosette grid 2", "Delta rosette grid 1", "Delta rosette grid 2", "Delta rosette grid 3", "Rectangular rosette grid 1", "Rectangular rosette grid 2", "Rectangular rosette grid 3"

%GageType,"Gage Type", ID, 5, GageTypeEnum, "e", ""

B.3.2.2 Bridge Type

Property name: **BridgeType**

The Bridge Type is an enumerated list describing the number of resistive elements in the bridge that are provided by the sensor. A measurement system may use this information to configure bridge completion circuitry, if needed. This property does not describe which of these resistive elements are active elements-such information is provided by the GageType property. The BridgeType property simply describes the electrical circuit presented by the sensor to the measurement system. The defined Bridge Types and their corresponding values of this parameter are shown in Table B.8:

Table B.8—BridgeType property description

%BridgeType value	Enumeration index	Description
“Quarter”	0	Quarter bridge (single resistive element)
“Half”	1	Half bridge (two resistive elements in series)
“Full”	3	Full bridge (complete four element bridge)

Example:

ENUMERATE BridgeTypeEnum, "Quarter", "Half", "Full"

%BridgeType, "Bridge type", ID, 2, BridgeTypeEnum,"e",""

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

B.3.2.3 Gage Factor

Property name: **GageFactor**

The Gage Factor of the strain gage indicates the sensitivity of the gage and is used in conjunction with the GageType property to define the mapping relation between the physical measurement of strain and the electrical signal of the strain gage. The Gage Factor should be specified independently of transverse sensitivity effects, as there is a separate property to specify the transverse sensitivity of the gage elements.

Example:

%GageFactor, "Gage Factor", CAL, 13, ConRelRes, 1.5, 0.000422, "0.00000", ""

B.3.2.4 Transverse sensitivity

Property name: **GageTransSens**

The transverse sensitivity of the strain gage indicates the sensitivity of the individual active elements to strains along the transverse axis of those elements. This can be used for further correction of the measured strain values, particularly in rosette applications. The equations given in Table B.7 can be modified to incorporate the effects of a gage's transverse sensitivity by modifying the GageFactor and PoissonCoef values as shown in Equation (B.1):

$$F = \hat{F}(1 - \hat{\nu}K_t) \qquad \nu = \frac{(\hat{\nu} - K_t)}{(1 - \hat{\nu}K_t)} \tag{B.1}$$

where

$\hat{F}$ and $\hat{\nu}$ are the GageFactor and PoissonCoef properties in the TEDS,

F and ν are the modified values to be used in the equations of Table 46,

K_t is the GageTransSens (transverse sensitivity) value.

Example:

%GageTransSens, "Transverse Sensitivity", CAL, 9, ConRes, -0.05, 0.0002, "0.00000", ""

B.3.2.5 Gage offset

Property name: **GageOffset**

This parameter is the zero-strain offset of the strain-gage bridge upon installation and mounting of the gage. Therefore, this value will typically be set to a value of 0.0 by the manufacturer and the modified by the end user after the gage is installed.

Example:

%GageOffset, "Gage offset", USR, 20, ConRes, -50E-3, 100E-9, "0.00p", "V/V"

B.3.2.6 Poisson Coefficient

Property name: **PoissonCoef**

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

The Poisson Coefficient of the strain gage is used, along with the GageFactor and GageTransSens properties, to define the mapping relation between the physical measurement of strain and the electrical signal of the strain gage. The equations shown in Table B.7 indicate how and whether the Poisson Coefficient is to be used in this mapping. Additionally, the Poisson Coefficient may be used along with the YoungsMod property converting rosette gage strain vales to multiaxial stress vectors.

Since the Poisson Coefficient is a function of a mounted sensor, it may not be possible for a manufacturer of unmounted gages to set the vale for this property; rather, it may need to be set by the installer of the gage.

Example:

%PoissonCoef, "Poisson Coefficient", USR, 14, ConRes, 0, 1E-4, "0.000000", ""

B.3.2.7 Youngs Modulus

Property name: **YoungsMod**

This property defines the Youngs Modulus of the material to which the strain gage is attached; it can be used to infer stress from the measured strain.

Example:

%YoungsMod, "Youngs Modulus", USR, 14, ConRes, 0.0, 100, "0", "MPa"

B.3.2.8 Gage Area

Property name: **GageArea**

This property defines the area of the strain gage. If there is more than one active element, this is the area of each individual element. This parameter may be used to infer the degree of self-heating of the gage.

Example:

%GageArea, "Gage Area", USR, 6, ConRelRes, 1, 0.06101, "0", "mm^2"

B.3.3 RTD and thermistor parameters

RTDs (resistance temperature detectors) and thermistors convert a physical measurement of temperature to an electrical value of resistance.

The mapping between temperature and RTDs is described by the Callendar-Van Dusen equation:

For $T < 0$ °C $\quad R = R_0 \times (1 + A \times T + B \times T^2 + (T - 100\ °C) \times C \times T^3)$

For $T > 0$ °C $\quad R = R_0 \times (1 + A \times T + B \times T^2)$

where

T = temperature in °C,
R = resistance in Ω.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Thermistors with a negative temperature coefficient have a mapping between physical measurements of temperature and electrical values of resistance that follow the Steinhart-Hart model. This model follows the Steinhart equation, which is defined by three coefficients:

$1/T = A + B \ln(R) + C \ln(R)^3$

where

T = temperature in °C,
R = resistance in Ω.

B.3.3.1 Reference resistance (R_0)

Property name: **RTDCoef_R0**

The R_0 coefficient of the Callendar-Van Dusen equation is equal to the resistance of an RTD at 0 °C. This parameter is also used with thermistors to indicate the resistance of the thermistor at 0 °C.

Example:

```
%RTDCoef_R0, "0 C resistance", ID, 20, ConRelRes, 1, 0.0000063, "0.00p", "Ohm"
```

B.3.3.2 RTD Callendar Van-Dusen coefficients

Property Names: **RTDCoef_A, RTDCoef_B, RTDCoef_C**

These parameters give the A, B, and C coefficients for the Callendar Van-Dusen equation for RTDs. RTDCoef_A is the first-order polynomial coefficient; RTDCoef_B is the second-order polynomial coefficient; and RTDCoef_C is the higher-order polynomial coefficient.

Example:

```
CASE "alpha=.00375", 0
  %RTDCoef_A, "Callendar-Van Dusen coefficient A", ID, 0, Single, "0.0000e-0", "1/C" = 3.8100e-3
  %RTDCoef_B, "Callendar-Van Dusen coefficient B", ID, 0, Single, "0.0000e-0", "1/C2" = -6.0200e-7
  %RTDCoef_C, "Callendar-Van Dusen coefficient C", ID, 0, Single, "0.000e-00", "1/C3" = -6.000e-12
ENDCASE
```

B.3.3.3 Steinhart-Hart coefficients

Property Names: **SteinhartA, SteinhartB, SteinhartC**

These parameters give the A, B, and C coefficients for the Steinhart-Hart equation for specifying thermistor transfer function curves.

Example:

```
%SteinhartA, "Steinhart coefficient A", ID, 32, SINGLE, "0.0000p", "1/C"
%SteinhartB, "Steinhart coefficient B", ID, 32, SINGLE, "0.0000p", "1/C"
%SteinhartC, "Steinhart coefficient C", ID, 32, SINGLE, "0.0000p", "1/C"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

B.3.3.4 Coefficient of self-heating

Property Names: **SelfHeating**

This parameter specifies the coefficient of self-heating generated by current flowing through the transducer.

Example:

%SelfHeating, "Self Heating Constant", ID, 5, ConRelRes, 0.00025, 0.0743491775, "rp", "W/°C"

B.3.4 Thermocouple parameters

Thermocouples have complex but well defined mappings between electrical values (dimensions of voltage) and physical values (dimensions of temperature). If a thermocouple mapping is chosen, additional fields are required to further define the mapping. These fields describe the thermocouple type (which defines the temperature/voltage relationship) and the cold-junction compensation method required.

B.3.4.1 Thermocouple calibration type

Property name: **TCType**

This is an enumerated list that indicates which of the 8 NIST defined thermocouple types the Mapping Method is derived from. The defined thermocouple types and their corresponding values of this parameter are shown in Table B.9:

Table B.9—TCType property description

%TCType Value	Enumeration index	Description
″B″	0	Type B
″E″	1	Type E
″J″	2	Type J
″K″	3	Type K
″N″	4	Type N
″R″	5	Type R
″S″	6	Type S
″T″	7	Type T
″non-standard″	8	Non-standard thermocouple curve

Example:

ENUMERATE TCTypeEnum, "B","E","J","K","N","R","S","T","non-standard"

%TCType, "Thermocouple Type", ID, 4, TCTypeEnum, "e", ""

B.3.4.2 Cold-junction compensation method

Property Names: **CJSource**

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

This property indicates whether or not the sensor requires cold-junction compensation. Actual thermocouples require the cold junction of the thermocouple to be measured and taken into account in the conversion between voltage and temperature. If the sensor provides actual thermocouple wires to the measurement device, this parameter should have a value of 0. If the sensor internally provides compensation and presents the measurement device with a voltage that corresponds to a thermocouple with a cold junction of 0 °C, then this parameter should have a value of 1. See Table B.10.

Table B.10—CJSource property description

%CJSource Value	Enumeration index	Description
“CJC not provided by sensor”	0	Cold-junction compensation is not provided by the sensor.
“Sensor compensated cold junction”	1	Sensor internally compensates for a 0 °C cold junction.

Example:

```
ENUMERATE CJSourceEnum, "CJC not provided by sensor","Sensor compensated cold junction"
%CJSource, "Cold Junction Compensation", ID, 1, CJSourceEnum,"e",""
```

B.4 Electrical signal properties

These electrical signal parameters are used to describe the electrical interface of the transducer. The information in these parameters may be used by a measurement system in configuring signal conditioning parameters such as gain, filtering, excitation, and bridge completion. In addition to these parameters, the parameters Minimum Electrical Value and Maximum Electrical Value listed under the Mapping Parameters heading are used to describe the useful range of the transducer.

B.4.1 Electrical Signal Type

Property Names: **ElecSigType**

The Electrical Signal Type is an enumerated list that broadly describes the electrical interface of the transducer. It is important to note that this parameter describes the electrical interface and not the internal operation of the transducer. For example, an LVDT transducer that internally provides signal conditioning such that it provides a simple 0–10 V signal to the measurement device would be classified as a voltage sensor, not an LVDT sensor. The defined Electrical Signal Types and their corresponding values of this parameter are shown in Table B.11.

Note that the electrical properties of some of these signal types, notably the bridge and pulse types, are further defined by additional properties.

Example:

```
ENUMERATE ElecSigTypeEnum,"Voltage Sensor","Current Sensor","Resistance Sensor","Bridge
Sensor","LVDT Sensor","Potentiometric Voltage Divider Sensor","Pulse Sensor","Voltage
Actuator","Current Actuator","Pulse Actuator"
%ElecSigType, "Transducer Signal Type", ID, 4, ElecSigTypeEnum,"e",""
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.11—ElecSigType property description

%ElecSigType Value	Enumeration index	Description
“Voltage Sensor”	0	Voltage sensor
“Current Sensor”	1	Current sensor
“Resistance Sensor”	2	Resistance sensor
“Bridge Sensor”	3	Bridge sensor
“LVDT Sensor”	4	LVDT sensor
“Potentiometric Voltage Divider Sensor”	5	Potentiometric voltage divider
“Pulse Sensor”	6	Pulse Sensor
“Voltage Actuator”	7	Voltage actuator
“Current Actuator”	8	Current Actuator
“Pulse Actuator”	9	Pulse Actuator

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

B.4.1.1 Response Time

Property Names: **RespTime**

The Response Time of the transducer describes the approximate rate at which the sensor output signal changes, or the rate at which an actuator can keep up with changes presented to its electrical signal. Measurement or control devices may use this information to configure filter settings or conversion rates. This parameter is meant to be a description of the bandwidth of the electrical signal; it may not necessarily represent the time it takes a physical phenomena change to be represented by a corresponding change in the electrical signal.

Example:

```
%RespTime, "Response Time", ID, 6, ConRelRes, 1E-6, 0.146, "rp", "sec"
```

B.4.1.2 Electrical signal coupling

Property Names: **ACDCCoupling**

This parameter is an enumerated list describing the signal coupling required for this transducer. If the transducer requires AC coupling, this value of this parameter will be a 1; if it does not, the parameter will have a value of 0. See Table B.12.

© IEEE 2004 – All rights reserved

Table B.12—ACDCCoupling property description

%ACDCCoupling Value	Enumeration index	Description
“DC”	0	DC coupling
“AC”	1	AC coupling

Example:

```
ENUMERATE ACDCCouplingEnum,"DC","AC"
%ACDCCoupling, "AC or DC Coupling", ID, 1, ACDCCouplingEnum,"e",""
```

B.4.1.3 Transducer impedance

Property Names: **SensorImped**

This parameter describes the impedance of a sensor at the reference or nominal levels of frequency and excitation. In the case of a bridge type transducer, this parameter should indicate the impedance of the individual resistive elements of the bridge (a quarter bridge consisting of a single 120 Ω resistive element and a half bridge of two 120 Ω resistive elements in series should both have a value of 120 Ω for this parameter).

Example:

```
%SensorImped, "Output Impedance of the Sensor", ID, 7, ConRelRes, 1, 0.043, "rp", "Ohm"
```

B.4.1.4 Discrete Signal Type

Property Names: **DiscSigType**

The Discrete Signal Type describes the basic function of the electrical interface of a discrete (also known as Boolean, 2-state, or logic) signal. This parameter may be used to describe in more detail a Pulse Sensor or actuator. The description of a Discrete Signal Type may be further refined by the Discrete Signal Amplitude parameter. The defined types of discrete signals and their corresponding values of this parameter are shown in Table B.13.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.13—DiscSigType property description

%DiscSigType Value	Enumeration index	Description
“Contact to Gnd”	0	Contact-to-ground (also called sinking, NPN, open-collector, or open-drain)
“Contact to Power”	1	Contact-to-power (also called sourcing, PNP, or open-emitter)
“Active Low”	2	Active low voltage (voltage levels may be further defined using the Discrete Signal Amplitude parameter)
“Active High”	3	Active high voltage (voltage levels may be further defined using the Discrete Signal Amplitude parameter)
“Bipolar”	4	Bipolar voltage (voltage levels may be further defined using the Discrete Signal Amplitude parameter)

These descriptions are written in terms of the device generating the output. Both a Pulse Sensor that provided as an electrical signal a contact-to-ground output and a Pulse Actuator that requires a controlling device to provide a contact-to-ground output would have a value of 0 for this parameter.

Example:

```
ENUMERATE DiscSigEnum,"Contact to Gnd","Contact to Power","Active Low","Active High","Bipolar"
%DiscSigType, "Discrete Signal Type", ID, 3, DiscSigEnum,"e",""
```

B.4.1.5 Discrete Signal Amplitude

Property Names: **DiscSigAmpl**

The Discrete Signal Amplitude describes the nominal voltage levels of a discrete signal. For example, a transducer whose electrical interface was a 0 to 5 V logic signal pulse train would have a Discrete Signal Type of 2 or 3 (active low or active high voltage) and a Discrete Signal Amplitude of 5 V. As another example, a transducer that provided a frequency output in the form of a +/–100 mV sine wave would have a Discrete Signal Type of 4 (bipolar voltage) and Discrete Signal Amplitude of 0.1 V.

Example:

```
%DiscSigAmpl, "Signal Level", ID, 5, ConRelRes, 1E-3, 0.52, "rp","V"
```

B.4.1.6 Pulse Measurement Type

Property Names: **PulseMeasType**

The Pulse Measurement Type property is an enumerated list that describes the type of measurement to be taken from a Pulse Sensor, or the type of pulses to be provided to a Pulse Actuator. The Pulse Measurement types and their corresponding values of this parameter are shown in Table B.14:

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.14—PulseMeasType property description

%PulseMeasType value	Enumeration index	Description
“Frequency”	0	Frequency (Hz)
“Period”	1	Period (seconds)
“On time”	2	On-time (seconds)
“Off time”	3	Off-time (seconds)
“Duty cycle”	4	Duty Cycle (percentage)
“Count”	5	Count (number of pulses)

On-time is defined as the amount of time during which the contact is closed (if Discrete Signal Type is a contact-to-ground or contact-to-power type), during which the signal is active (in the case of an active low or active high type), or during which the signal is greater than 0 V (in the case of a bipolar type). Likewise, off-time is defined as the amount of time during which the contact is open, the signal is inactive or negative.

Example:

```
ENUMERATE PulseTypeEnum, "Frequency", “Period", "On time", "Off time", "Duty cycle", "Count"
%PulseMeasType, "Pulse Measurement Type", ID, 3, PulseTypeEnum,"e",""
```

B.4.1.7 Gain

Property Names: **Gain**

This property contains the amplification gain of the device described by the TEDS. Also used with subproperty `[Function]` to select gain.

Example:

```
%Gain, "Gain", Cal, 12, ConRelRes, 5E7, 0.001,  "0.000p", "V/C"
```

B.4.1.8 Filter

Property name: **Filter**

This property is used with subproperty `[Function]` to select filter mode.

Example:

```
%Filter, %TF_HP_S["00x1"], ID, 8, BitBin,"", ""= "0001"
```

B.4.1.9 Temperature coefficient

Property name: **TempCoef**

This property gives the temperature coefficient of the transfer function of the transducer, as described in the following formula:

$H(T) = (1 + TempCoef\ (T - RefTemp))$

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

implying that TempCoef is the absolute factor needed for multiplication by *(T–RefTemp)* to describe the change imposed by temperature. It is the coefficient found at the reference temperature.

Example:

%TempCoef, "Temperature coefficient (b)", CAL, 9, ConRes, -0.5, 0.002,"0.000","%/°C"

B.4.2 Microphone and preamplifier parameters

B.4.2.1 Polarization voltage

Property name: **RefPol**

This parameter specifies the polarization voltage that was used to obtain the sensitivity stated. See Table B.15.

Table B.15—Polarization voltage property description

%RefPol Value	Enumeration index	Description
“Prepolarized”	0	Prepolarized
“28 V”	1	28 V
“200 V”	2	200 V

Example:

ENUMERATE PolarizationEnum, "Pre-polarized", "28 V", "200 V"

%RefPol, "Polarization Voltage", CAL, 2, PolarizationEnum, "e", ""

B.4.2.2 Amplifier resistance and capacitance

Property Names: **Rin, Cin, Rout**

The parameters Rin and Rout specify the amplifier input and output resistance. The parameter Cin specifies the input capacitance.

Example:

%Rin, "Rin", CAL, 14, ConRelRes, 1, 0.001, "0.000p", "ohm"

%Cin, "Cin", CAL, 6, ConRes, 0, 10E-15, "0.00p", "F"

%Rout, "Rout", CAL, 5, ConRelRes,5, 0.10, "0.0p", "ohm"

B.4.2.3 Stiffness of transducer

Property name: **Stiffness**

This parameter specifies the total stiffness of a force transducer.

Example:

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

%Stiffness, "Stiffness of transducer", CAL, 6, ConRelRes, 1E6, 0.10, "rp", "N/m"

B.4.2.4 Mass below gage

Property name: **Mass_below**

This parameter specifies the mass below the sensitive element in a force transducer (so-called unwanted mass).

Example:

%Mass_below, "Mass below gage", CAL, 6, ConRelRes, 0.1, 0.1, "rp", "g"

B.4.2.5 Weight of transducer

Property name: **Weight**

This parameter specifies the total weight of a transducer.

Example:

%Weight, "Transducer weight", CAL, 6, CONRELRES, 0.1, 0.1, "rp", "g"

B.4.2.6 System test available in amplifier

Property name: **TestGain**

This parameter specifies that a system test is available. A voltage can be applied, and the TestGain specifies the output of the system in dB below the applied voltage.

Example:

%TestGain, "Test Gain", CAL, 10, ConRes, 0, 0.1, "000.0", "dB"

B.4.2.7 Microphone capacitance

Property name: **Cmic**

This parameter specifies the active capacitance of the microphone.

Example:

%Cmic, "Microphone capacitance", CAL, 9, ConRes, 5E-12, 0.3E-12, "0.000p", "F"

B.4.2.8 Microphone stray capacitance

Property name: **Cstray**

This parameter specifies the stray capacitance of the microphone. See Figure B.1.

Example:

%Cstray, "Passive capacitance", CAL, 5, ConRelRes, 0.15E-12, 0.1, "0.000p", "F"

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

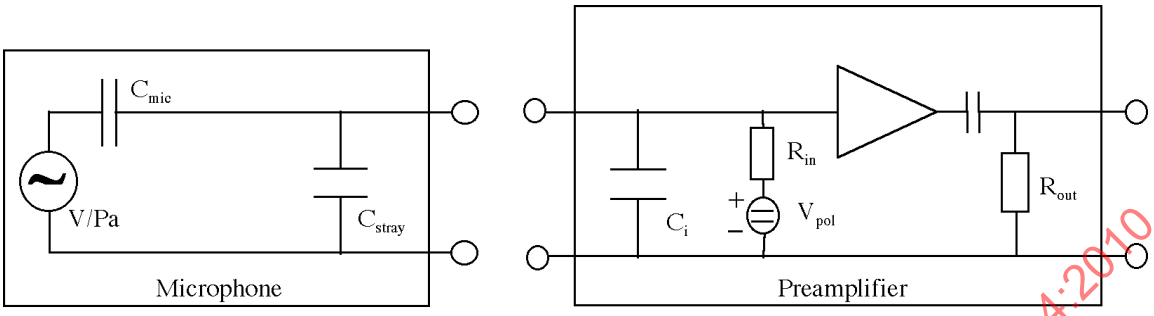


Figure B.1—Schematic of microphone and preamplifier electrical properties

B.4.2.9 Microphone type

Property name: **MicType**

This parameter specifies the type of microphone, according to IEC 61094-1:2000 [B5] and IEC 61094-4:1995 [B6]. See Table B.16.

Table B.16—MicType property description

% MicType value	Enumeration index	Description
“Free field”	0	Free field
“Pressure”	1	Pressure
“Random incidence”	2	Random incidence
“Other”	3	Other

Example:

ENUMERATE MicTypeEnum, "Free field", "Pressure", "Random incidence", "Other"

%MicType, "Microphone Type", CAL, 2, MicTypeEnum, "e", ""

B.4.2.10 Microphone size

Property name: **MicSize**

This parameter specifies the size of microphone, according to IEC 61094-1:2000 [B5] and IEC 61094-4:1995 [B6]. See Table B.17.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.17—MicSize property description

%MicSize value	Enumeration index	Description
“1 inch”	0	1 inch
“1/2 inch”	1	1/2 inch
“1/4 inch”	2	1/4 inch
“1/8 inch”	3	1/8 inch

Example:

```
ENUMERATE MicSizeEnum, "1 inch", "1/2 inch", "1/4 inch", "1/8 inch"
%MicSize, "Microphone Size", CAL, 2, MicSizeEnum, "e", ""
```

B.4.2.11 Frequency response type

Property Names: **Resp_Type**

This parameter specifies the type of response given by the transfer function, either actuator or corrected response, as shown in Table B.18.

Table B.18—Resp_Type property description

%ACDCCoupling value	Enumeration index	Description
“Actuator”	0	Actuator
“Corrected”	1	Corrected

Example:

```
ENUMERATE Resp_TypeEnum, "Actuator", "Corrected"
%Resp_Type, "Response type",  CAL, 1, Resp_TypeEnum, "e", ""
```

B.4.2.12 Reference pressure

Property Names: **RefPress**

This parameter specifies the pressure at which the measurement data was obtained.

Example:

```
%RefPress, "Reference pressure",  CAL, 6, ConRes, 80000, 500, "0", "Pa"
```

B.4.2.13 Equivalent microphone volume

Property Names: **Equi_Vol**

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

This parameter specifies the equivalent volume.

Example:

%Equi_Vol, "Equivalent volume", CAL, 8, ConRes, 0, 10E-6, "0.0000p", "m^3"

B.4.2.14 Gate present

Property Names: **Gate**

This parameter specifies if there is a gate that can open for more nodes. This can be used at another level during the discovery of the network topology.

Example:

ENUMERATE YesNoEnum, "No","Yes"
%Gate, "Gate present", ID, 1, YesNoEnum, "e",""

YesNoEnum has the values "No" and "Yes" for the indexes 0 and 1.

B.4.3 Excitation or power level specification

The properties in this subclause specify the requirements for external power, or excitation, for the transducer. This information, along with the electrical signal information may be used.

B.4.3.1 Excitation or power-supply levels

Property Names: **ExciteAmplNom, ExciteAmplMin, ExciteAmlMax**

These three parameters specify the amplitude of the excitation or power supply required to operate the transducer. ExciteAmplNom specifies the nominal excitation or power level, while ExciteAmplMin and ExciteAmplMax specify the minimum and maximum levels, respectively.

Example:

%ExciteAmplNom, "Excitation Level (Nominal)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
%ExciteAmplMin, "Excitation Level (Minimum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
%ExciteAmplMax, "Excitation Level (Maximum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"

B.4.3.2 Excitation Type

Property Names: **ExciteType**

The Excitation Type property is an enumerated list that indicates whether the excitation amplitude properties (ExciteAmplNom, ExciteAmplMin, ExciteAmplMax) describe a single DC level, a pair of bipolar DC levels, or an AC (rms) level. The values of this property, shown in Table B.19, are defined for these Excitation Types.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table B.19—ExciteType property description

%ExciteType value	Enumeration index	Description
“DC”	0	DC power or excitation
“Bipolar DC”	1	Bipolar, or dual, DC supply (i.e., +15 VDC and –15 VDC)
“AC (rms)"	2	AC (rms)

For example, an excitation amplitude (ExciteAmplNom) of 10 V would imply (a) a 10 VDC excitation if the Excitation Type were 0, (b) both +10 V and –10 V if the Excitation Type were 1, and (c) 10 Vrms excitation if the Excitation Type were 2.

Example:

ENUMERATE ExciteTypeEnum, "DC", "Bipolar DC", "AC (rms)"

%ExciteType, "Excitation Voltage Type", ID, 2, ExciteTypeEnum, "e", ""

B.4.3.3 Excitation current draw

Property Names: **ExciteCurrentDraw**

This parameter specifies the amount of current that shall be supplied by a excitation or power-supply voltage source. This parameter should specify the maximum current load at the nominal excitation level (ExciteAmplNom) under standard operating conditions.

Example:

%ExciteCurrentDraw, "Maximum current draw at nominal excitation level", ID, 6, ConRelRes, 1e-6, 0.129462705898, "rp", "A"

B.4.3.4 Excitation frequency

Property Names: **ExciteFreqNom, ExciteFreqMin, ExciteFreqMax**

If the Excitation Type is set to a value of 2 (AC excitation), then these parameters specify the range, and nominal value, of the frequency of the excitation signal. ExciteFreqNom specifies the nominal, or reference, frequency, while ExciteFreqMin and ExciteFreqMax specify the allowable range of frequency of the excitation source.

Example:

%ExciteFreqNom, "Excitation Frequency (Nominal)", ID, 12, ConRes, 1, 1, "0", "Hz"

%ExciteFreqMin, "Excitation Frequency (Minimum)", ID, 12, ConRes, 1, 1, "0", "Hz"

%ExciteFreqMax, "Excitation Frequency (Maximum)", ID, 12, ConRes, 1, 1, "0", "Hz"

B.4.3.5 Loop supply parameters

Property Names: **LoopSupplyMin, LoopSupplyMax**

IECNORM.COM: Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

These parameters together specify the allowable range of power-supply voltage for loop-powered current loop devices. LoopSupplyMin specifies the minimum allowable supply voltage, while LoopSupplyMax specifies the maximum loop supply voltage.

Example:

```
%LoopSupplyMin, "Loop Supply (Minimum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
%LoopSupplyMax, "Loop Supply (Maximum)", ID, 9, ConRes, 0.1, 0.1, "0.0", "V"
```

B.5 Calibration parameters

The properties included in this subclause provide either actual calibration data or calibration management information, such as date of calibration and calibration period.

B.5.1 Calibration management information

B.5.1.1 Calibration Date

Property name: **CalDate**

This parameter specifies the date on which the last calibration was performed on the transducer.

Example:

```
%CalDate, "Calibration Date", CAL, 16, DATE, "d-mmm-yyyy",""
```

B.5.1.2 Calibration initials

Property name: **CalInitials**

CalInitials contains the initials of the individual who performed the last calibration on the Calibration Date.

Example:

```
%CalInitials, "Calibration Initials", CAL, 15, CHR5, "s", ""
```

B.5.1.3 Calibration period

Property name: **CalPeriod**

This parameter specifies the recommended time period between subsequent calibrations of the transducer. This property may be added to the Calibration Date to determine the next calibration due date.

Example:

```
%CalPeriod, "Calibration Period (Days)", CAL, 12, UNINT, "0", "days"
```

B.5.2 Calibration table and curve specification

A calibration table or calibration curve that specifies the complete input-output transfer function of the transducer may be described using the calibration table properties or calibration curve properties.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

B.5.2.1 Calibration table properties

Property Names: **CalTable_Domain, CalPoint_DomainValue, CalPoint_RangeValue**

These properties are used to specify a calibration look-up table to describe the electrical-to-physical transfer function of the transducer. These parameters would typically be used with the STRUCTARRAY command to create the look-up table values.

CalTable_Domain is an enumerated value used to select the domain parameter for the calibration table as either the electrical or physical parameter. A value of 0 specifies the domain as electrical; a value of 1 specifies the physical phenomenon as the domain.

CalPoint_DomainValue is the value of the domain parameter for which the calibrated value is specified. CalPoint_DomainValue may be specified as a percentage of the full scale.

CalPoint_RangeValue specifies the deviation of the range value from the linear transfer function.

The description of a calibration table will require multiple sets of values of CalPoint_DomainValue and CalPoint_RangeValue, generated using the STRUCTARRAY command.

Example:

```
ENUMERATE CalSetEnum, "Electrical", "Physical"

%CalTable_Domain, "Domain parameter of the Calibration Table", CAL, 1,  CalSetEnum, "e", ""

STRUCTARRAY CalTable, "Calibration table", CAL, 7
   %CalPoint_DomainValue, "Cal Point (% of full span)", CAL, 16, ConRes, 0, 0.00153, "0.00", "%"
   %CalPoint_RangeValue, "Cal Deviation (% of full span)", CAL, 21, ConRes, -100, 1E-4, "0.0000", "%"
ENDSTRUCTARRAY
```

Figure B.2 illustrates the use of CalPoint_DomainValue and CalPoint_RangeValue properties to specify the transducer mapping. In this figure, the i^{th} value read for %CalPoint_DomainValue is designated as $DomainVal_i$, while the i^{th} value read for %CalPoint_RangeValue is designated as $RangeVal_i$.

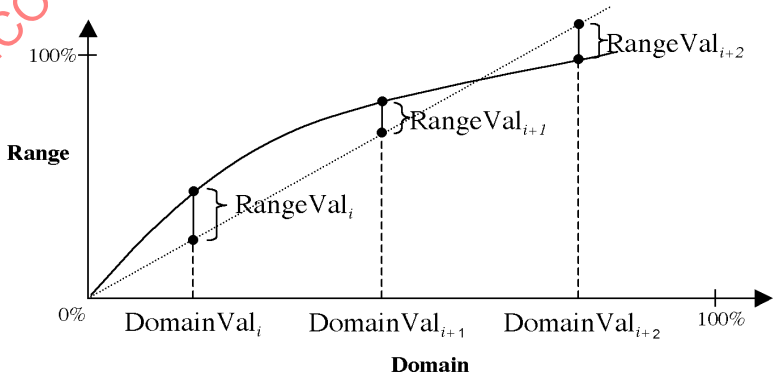


Figure B.2—Calibration table properties

B.5.2.2 Calibration curve properties

Property Names: **CalCurve_Domain, CalCurve_PieceStart, CalCurve_Power, CalCurve_Coef**

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

These properties are used to specify a multi-segment polynomial calibration curve to describe the electrical-to-physical transfer function of the transducer.

CalCurve_Domain is an enumerated value used to select the domain parameter for the calibration table as either the electrical or physical parameter. A value of 0 specifies the domain as electrical; a value of 1 specifies the physical phenomenon as the domain.

CalCurve_PieceStart specifies the starting value of each piece segment of the domain parameter. Each segment is therefore bounded by sequential values of CalCurve_PieceStart. For example, segment k is bounded by CalCurve_PieceStart$_k$ and CalCurve_PieceStart$_{k+1}$. Each segment shall be closed below and open above (i.e., the segment k interval includes CalCurve_PieceStart$_k$ but excludes CalCurve_PieceStart$_{k+1}$).

For each segment, CalCurve_Power and CalCurve_Coef are arrays specifying a polynomial equation.

Example:

```
ENUMERATE CalSetEnum, "Electrical", "Physical"

%CalCurve_Domain, "Domain parameter of the Calibration Curve", CAL, 1, CalSetEnum, "e", ""

STRUCTARRAY CalCurve, "Calibration curve segments", CAL, 8
  %CalCurve_PieceStart, "Start value of segment (% of FS)", CAL, 13, ConRes, 0, 100, "0.00", "%"
  STRUCTARRAY CalCurve_Poly, "Calibration curve polynomial", CAL, 7
    %CalCurve_Power, "Power of domain value", CAL, 7, ConRes, -32, 0.5,  "0.0", ""
    %CalCurve_Coef, "Polynomial coefficient", CAL, 32, Single,  "0.000E+0", ""
  ENDSTRUCTARRAY
ENDSTRUCTARRAY
```

Figure B.3 illustrates the use of the calibration curve properties. PieceStart$_k$ designates the k^{th} value read for the %CalCurve_PieceStart property. For segment k, defined by PieceStart$_k$ and PieceStart$_{k+1}$, the multinomial curve for this segment is shown where:

n_k = the number of coefficients for segment k, read from the TEDS as CalCurve_Poly,

$C_{k,i}$ = the i^{th} polynomial coefficient for segment k, read from the TEDS as CalCurve_Coef,

$P_{k,i}$ = the power to which domain variable x is raised for coefficient $C_{k,i}$ in segment k.

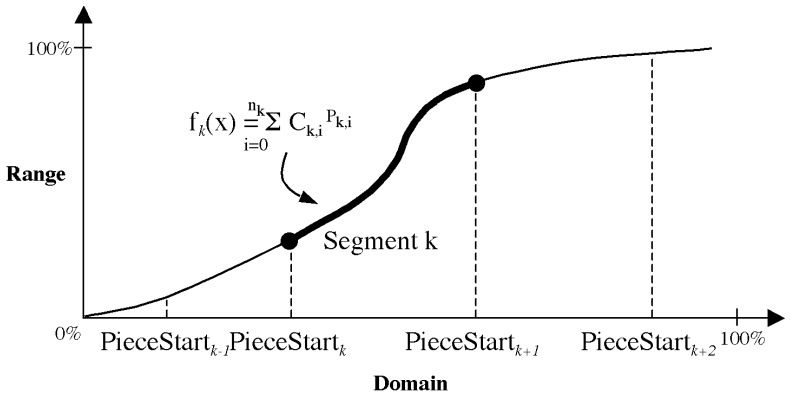


Figure B.3—Calibration curve properties

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

B.5.3 Transfer function

The following properties specify the frequency response transfer function of the transducer. Two methods of specifying the transfer function are defined. First, a transfer function can be compactly defined using poles and zeroes. Alternatively, the transfer function can be specified as a look-up table of frequency-amplitude pairs.

In the TDL, the basic functions are named by properties, and the next number of bits represents the value read in from the node in the IEEE 1451.4 Transducer. The exact number of bits is given by a decimal value as shown in the examples following each of the properties.

Multiplying any number of the following basic functions composes the transfer function.

B.5.3.1 Single zero

Property name: **TF_SZ**

The value picked up from the node at the specified place in the memory is used as Fsz in the transfer function shown in Equation (B.2).

$$H(f, \mathrm{Fsz}) = \left(1 + \frac{j \cdot f}{\mathrm{Fsz}}\right) \tag{B.2}$$

B.5.3.2 Single pole

Property name: **TF_SP**

The value picked up from the node at the specified place in the memory is used as Fsp in the transfer function shown in Equation (B.3), which directly describes a low-pass filter of first order with the –3dB point at Fsp.

$$H(f, \mathrm{Fsp}) = \frac{1}{1 + \frac{j \cdot f}{\mathrm{Fsp}}} \tag{B.3}$$

Example:

```
%TF_SP, "Low pass cut-off frequency (F lp)", CAL, 7, ConRelRes, 10, 0.05, "rp", "Hz"
```

B.5.3.3 Complex zero

Property name: **TF_KZr**

The value picked up from the node at the specified place in the memory is used as Fzres in the transfer function as shown in Equation (B.4), where the parameter Qz is given in the next property.

$$H(f, \mathrm{Fzres}, \mathrm{Qz}) = \left(1 + \frac{j \cdot f}{\mathrm{Qz} \cdot \mathrm{Fzres}} + \left(\frac{j \cdot f}{\mathrm{Fzres}}\right)^2\right) \tag{B.4}$$

B.5.3.4 Quality factor (zero)

Property name: **TF_KZq**

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

This property gives the parameter Qz for the transfer function of a complex zero in the previous property. This is typically used to describe the behavior of a single degree of freedom system such as the spring mass system in an accelerometer or the membrane with air damping in a microphone. The parameters are the resonance frequency and Q or quality factor of the response curve.

B.5.3.5 Complex pole

Property name: **TF_KPr**

The value picked up from the node at the specified place in the memory is used as Fpres in the transfer function shown in Equation (B.5), where the parameter Qp is given in the next property.

$$H(f, \mathrm{Fzres}, \mathrm{Qp}) = \frac{1}{\left(1 + \frac{j \cdot f}{\mathrm{Qp} \cdot \mathrm{Fpres}} + \left(\frac{j \cdot f}{\mathrm{Fpres}}\right)^2\right)} \quad \text{(B.5)}$$

Example:

%TF_KPr, "Resonance frequency (F res)", CAL, 9, ConRelRes, 100, 0.01, "rp", "Hz"

B.5.3.6 Quality factor (pole)

Property name: **TF_KPq**

This property gives the parameter Qp for the transfer function of a complex pole in the previous property.

Example:

%TF_KPq, "Quality factor @ F res (Q)", CAL, 9, ConRelRes, 0.4, 0.01, "rp",""

B.5.3.7 Single zero at zero and a single pole (high-pass function)

Property name: **TF_HP_S**

The value picked up from the node at the specified place in the memory is used as Fhp in the transfer function shown in Equation (B.6), which directly describes a high-pass filter of first order with the –3dB point at Fhp.

$$H(f, \mathrm{Fhp}) = \frac{\frac{j \cdot f}{\mathrm{Fhp}}}{\left(1 + \frac{j \cdot f}{\mathrm{Fhp}}\right)} \quad \text{(B.6)}$$

Example:

%TF_HP_S, "High pass cut-off frequency (F hp)", CAL, 8, ConRelRes, 0.005, 0.03, "rp", "Hz"

B.5.3.8 Constant relative slope

Property name: TF_SL

The value picked up from the node at the specified place in the memory is used as Fref in the transfer function shown in Equation (B.7).

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

$$H(f, \mathrm{a}, \mathrm{Fref}) = \left(\frac{j \cdot f}{\mathrm{Fref}}\right)^{\frac{a}{\ln(10)}} \tag{B.7}$$

Example:

%TF_SL, "Amplitude slope (a)", CAL, 7, ConRes, -6.3, 0.1, "0.0", "%/decade"

B.5.3.9 Single zero dependent on previous property

Property name: **TF_SZm**

The value picked up from the node at the specified place in the memory is used as x in the transfer function shown in Equation (B.8).

$$H(f, \mathrm{x}, \mathrm{v}) = \left(1 + \frac{j \cdot f}{\mathrm{x} \cdot \mathrm{v}}\right) \tag{B.8}$$

Example:

%TF_SZm, "F high/ F low lift", CAL, 8, ConRelRes, 1, 0.0015, "rp", ""

B.5.3.10 Single pole dependent on previous property

Property name: **TF_SPm**

The value picked up from the node at the specified place in the memory is used as x in following transfer function, as shown in Equation (B.9):

$$H(f, \mathrm{x}, \mathrm{v}) = \frac{1}{\left(1 + \frac{j \cdot f}{\mathrm{x} \cdot \mathrm{v}}\right)} \tag{B.9}$$

B.5.3.11 Phase correction

Property name: **PhaseCorrection**

This property specifies phase correction to compensate a phase shift at the reference frequency.

Example:

%PhaseCorrection, "Phase correction at F ref", CAL, 11, ConRes, -90.0, 0.1, "0.0", "degrees"

B.5.3.12 Alternate transfer function table

Property Names: **TF_Table_Freq, TF_Table_Ampl**

The frequency response transfer function of a transducer can be specified using a look-up table of frequency-amplitude data pairs. Similar to the CalTable and CalCurve properties, TF_Table_Freq and TF_Table_Ampl define data points in the amplitude-versus-frequency transfer function.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

The TF_Table_Freq and TF_Table_Ampl are typically used within a STRUCTARRRAY structure. For each element of the array structure generated by the STRUCTARRAY command, TF_Table_Ampl specifies the transfer function amplitude, in percentage, at a given frequency specified by TF_TableFreq.

Example:

```
STRUCTARRAY TF_Table, "Calibration table", CAL, 7
  %TF_Table_Freq, "Frequency", CAL, 15, ConRelRes, 1, 0.000215, "0.00p", "Hz"
  %TF_Table_Ampl, "Amplitude", CAL, 21, ConRes, -100, 0.0001, "0.0000", "%"
ENDSTRUCTARRAY
```

Figure B.4 illustrates the use of TF_Table_Freq and TF_Table_Ampl properties to specify the transducer transfer function. In this figure, the i^{th} value read for %TF_Table_Freq is designated as $TF\_Table\_Freq_i$, while the i^{th} value read for %TF_Table_Ampl is designated as $TF\_Table\_Ampll_i$.

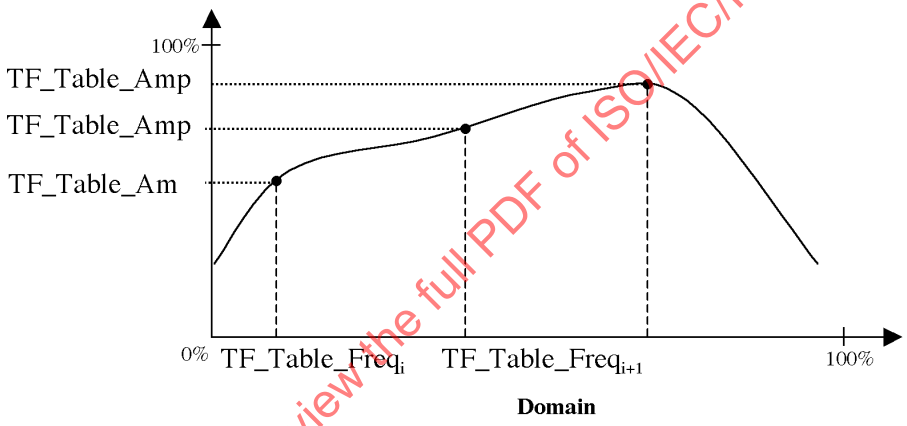


Figure B.4—Use of a table to specify the transducer transfer function

B.6 Miscellaneous parameters

B.6.1 Attached transducer information

Many transducer configurations include the use of an IEEE 1451.4-compatible signal amplifier, preamplifier, or general signal conditioner that is separate from the actual transducer. For example, the standard template collection in Annex A includes templates for a charge amplifier and a microphone preamplifier. As the measurement system ideally requires TEDS information for both the amplifier device as well as the transducer that is connected to the device, these templates provide options for including properties for the attached transducer. This subclause includes the properties for identification of the attached transducer which are equivalent in format to the Basic TEDS information.

B.6.1.1 Basic TEDS information for transducers attached to IEEE 1451.4 amplifiers

Property name: **Attached_MfgrID, Attached_ModeNum, Attached_VersionLetter, Attached_VersionNum, Attached_SerialNum**

These properties provide the equivalent of Basic TEDS identification information for transducers that are attached to an IEEE 1451.4 amplifier, preamplifier, or signal conditioner. For example, the standard IEEE

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

charge amplifier and microphone preamplifier templates include the option to describe the combined characteristics of the amplifier and of the transducer that is attached.

These five properties follow the same data format as specified for the 64-bit Basic TEDS described 5.1. In particular, the Attached_MfgrID will also use the enumerated list of manufacturers contained in a file named ManufactureList.xdl. Note that the 64-bit Basic TEDS, typically contained in the OTP register of the memory device, will contain the identification information for the amplifier, while these properties, contained in the template, contain the identification information for the attached transducer.

Example:

```
%Attached_MfgrID, "Manufacturer of attached transducer", CAL, 14, UNINT, "", ""
%Attached_ModelNum, "Model number of attached transducer", CAL, 15, UNINT, "", ""
%Attached_VersionLetter, "Version letter of attached transducer", CAL, 5, CHR5, "s", ""
%Attached_VersionNum, "Version number of attached transducer", CAL, 6, UNINT, "", ""
%Attached_SerialNum, "Serial number of attached transducer", CAL, 24, UNINT, "", ""
```

B.6.1.2 Basic TEDS information for systems with transducers attached to IEEE 1451.4 amplifiers

Property name: **System_MfgrID, System_ModeNum, System_VersionLetter, System_VersionNum, System_SerialNum**

These properties provide the equivalent of Basic TEDS identification information for systems with transducers that are attached to an IEEE 1451.4 amplifier, preamplifier, or signal conditioner. For example, the standard IEEE charge amplifier and microphone preamplifier templates include the option to describe the combined characteristics of the amplifier and of the transducer that is attached and give it a new system identity in the form of System Basic TEDS.

These five properties follow the same data format as specified for the 64-bit Basic TEDS described 5.1. In particular, the System_MfgrID will also use the enumerated list of manufacturers contained in a file named ManufactureList.xdl. Note that the 64-bit Basic TEDS, typically contained in the OTP register of the memory device, will contain the identification information for the amplifier, while these properties, contained in the template, contain the identification information for the attached transducer or the System.

Example:

```
%System_MfgrID, "Manufacturer of System", CAL, 14, UNINT, "", ""
%System_ModelNum, "Model number of System", CAL, 15, UNINT, "", ""
%System_VersionLetter, "Version letter of System", CAL, 5, CHR5, "s", ""
%System_VersionNum, "Version number of System", CAL, 6, UNINT, "", ""
%System_SerialNum, "Serial number of System", CAL, 24, UNINT, "", ""
```

B.6.2 Miscellaneous properties

B.6.2.1 Passive mode

Property name: **Passive**

This property indicates whether the transducer supports passive mode. This is used to configure multiplexed configurations; also used with subproperty `[Function]` to select passive mode.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Example:

```
%Passive, "Passive Mode", CAL, 1, UNINT,"", ""
```

B.6.2.2 Measurement location ID

Property name: **MeasID**

This property serves as a user-specified location ID for the transducer.

Example:

```
%MeasID, "Measurement location ID",  USR, 11, UNINT,"0", ""
```

B.6.2.3 Cable capacitance

Property name: **Ccable**

This property specifies the capacitance of the cable.

Example:

```
%Ccable, "Cable capacitance", Cal, 6, ConRes, 0, 10E-15, "rp", "F"
```

B.6.2.4 Cable length

Property name: **CableLen**

This property specifies the length of the cable attached to the transducer.

Example:

```
%CableLen, "Cable Length", CAL, 0, SINGLE, "0.0p", "mm"
```

B.6.2.5 Appended TEDS

Property name: **Appended_TEDS, Appended_TEDS_location**

Appended_TEDS indicates whether an Appended TEDS exists, while Appended_TEDS_location indicates its location. See 7.4.10 for more information.

B.6.2.6 EMBTPL

Property name: **EMBTPL**

This property indicates that the data following this property is in ASCII-based template format containing legal TDL commands. See 7.4.11 for more information.

B.6.2.7 XML

Property name: **XML**

This property indicates that the data following this property is in ASCII-based XML format. See 7.4.10 for more information.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

B.6.2.8 Manufacturer-defined properties

Property name: **MDEF**

Properties that are defined by manufacturers and not IEEE standard properties may begin with the prefix MDEF. See 7.4.10 for more information.

B.6.2.9 User template validation checksum

Property name: **TDL_CHKSUM**

This property specifies the checksum of the template file. User-defined templates are not unique. It is the users responsibility to ensure that the template used is the same as the one used when the data in the transducer was written. The TDL_CHKSUM is used for this purpose. The user stores the checksum for the specific user-defined template file in the transducer. This is then validated each time the transducer's data are being read and interpreted. See 6.5.1 for more information.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Annex C

(informative)

TDL formal grammar

This annex provides a formal description of the TDL syntax. It does so using token and grammar definition conventions used by the compiler compilers flex and bison. (These are variations of lex and yacc, see Levine et al. [B8] and Mellor and Balcer [B10].) These two programs use the token and grammar definition input files to generate a lexical and semantical parsing program (i.e., a compiler).

In addition to the formal grammar, code for implementing semantic actions is also provided. The purpose of this is to provide potential users of this standard with a program that can be used as a syntax checker, or that could be used as the foundation for a full template and TEDS parser. In order to use this code, flex and bison (or other variants) must be used to generate the tokenizing and parsing code itself.

C.1 TDL token definition (flex input file)

This subclause lists the flex input file that defines the tokens.

```
%{

//------------------------------------------------------------------
// TDL.l
//
// This file is a standard input file to flex (which is a
// variant of lex).  It contains token definitions for the
// IEEE 1451.4 Template Description Language (TDL). This file
// should be used in conjunction with bison and its TDL input
// file tdl.y.  To use this file execute the command line:
//
//          flex -i tdl.l
//
// or the command line
//
//          os2 /p flex.exe /c flex -i tdl.l
//
// After executing this, then
//
//          rename lexyy.c lexyy.cpp
//
// Then execute bison on the tdl.y file and then compile the main
// TDL compiler using a C++ compiler.
//
// Note: It has been reported that some versions of flex generate
// code which includes a #include <unistd.h> statement.  If such a
// file does not exist, then create a blank file of this name in
// the working directory.
//
// Updated 10 September 2003
//
//------------------------------------------------------------------

#include "TDL_tab.h"
#include "tdlparser.h"

extern int statement_count;
extern void print_statement_num ();
extern void print_token (char*);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
%}

%%

TDL_VERSION_NUMBER      {print_token ("TDL VERSION NUMBER");
                        return(TDL_VERSION_NUMBER);}
VALIDATION_KEYCODE      {print_token ("VALIDATION_KEYCODE");
                        return(VALIDATION_KEYCODE);}
UGID                {print_token ("UGID"); return(UGID);}
SPACING             {print_token ("SPACING"); return(SPACING);}
SELECTCASE          {print_token ("SELECTCASE"); return(SELECTCASE);}
ENDSELECT           {print_token ("ENDSELECT"); return(ENDSELECT);}
CASE                {print_token ("CASE"); return(CASE);}
ENDCASE             {print_token ("ENDCASE"); return(ENDCASE);}
STRUCTARRAY         {print_token ("STRUCTARRAY"); return(STRUCTARRAY);}
ENDSTRUCTARRAY      {print_token ("ENDSTRUCTARRAY");
                    return(ENDSTRUCTARRAY);}
ENUMERATE           {print_token ("ENUMERATE"); return(ENUMERATE);}
PHYSICAL_UNIT       {print_token ("PHYSICAL_UNIT");
                    return(PHYSICAL_UNIT);}

ID          {print_token ("ID"); return(ID);}
CAL         {print_token ("CAL"); return(CAL);}
USR         {print_token ("USR"); return(USR);}
ALIGN       {print_token ("ALIGN"); return(ALIGN);}
UNINT       {print_token ("UNINT"); return(UNINT);}
CHR5        {print_token ("CHR5"); return(CHR5);}
STRING5     {print_token ("STRING5"); return(STRING5);}
STRING7     {print_token ("STRING7"); return(STRING7);}
STRING16    {print_token ("STRING16"); return(STRING16);}
DATE        {print_token ("DATE"); return(DATE);}
ConRelRes   {print_token ("ConRelRes"); return(CONRELRES);}
ConRes      {print_token ("ConRes"); return(CONRES);}
ASCII       {print_token ("ASCII"); return(ASCII);}
Single      {print_token ("Single"); return(SINGLE);}
Unicode     {print_token ("Unicode"); return(UNICOD);} // UNICODE
conflicts with a C++ variable

DEFAULT     {print_token ("DEFAULT  "); return(DEFAULT      );}
INITIALIZE  {print_token ("INITIALIZE"); return(INITIALIZE);}
READWRITE   {print_token ("READWRITE"); return(READWRITE);}
CTRLFUNCTIONMASK  { print_token ("CTRLFUNCTIONMASK");
                  return(CTRLFUNCTIONMASK);}
FUNCTIONTYPE      {print_token ("FUNCTIONTYPE");
                  return(FUNCTIONTYPE);}
FUNCTION          {print_token ("FUNCTION"); return(FUNCTION);}

ENDTEMPLATE       {print_token ("ENDTEMPLATE"); return(ENDTEMPLATE);}
ABSTRACT[^\n]*    {yylval.str=strdup(yytext); print_token (yylval.str);
                  return(ABSTRACT);}
TEMPLATE          {yylval.str=strdup(yytext); print_token (yylval.str);
                  return(TEMPLATE);}

\%[A-Za-z][A-Za-z0-9\^$_!@#&]*      { yylval.str=strdup(yytext);
                                    print_token (yylval.str);
                                    return(PROPERTYNAME);}
\/\/[^\n]* { yylval.str=strdup(yytext); print_token (yylval.str);
           return(COMMENT);}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
[A-Za-z][A-Za-z0-9\^$_!@#&]*  { yylval.str=strdup(yytext);
                                print_token (yylval.str);
                                return(VARIABLE_NAME);}

[\n]          {return (END_OF_LINE); }
[\t ]*        {}

\"[^"\n]*\" |
\'[^'\n]*\'         {yylval.str=strdup(yytext); print_token (yylval.str);
                    return(QUOTED_STRING);}

\[            |
\]            |
\=            |
\,            |
\(            |
\)                  {print_token (&yytext[0]); return(yytext[0]);}

[-+]?[0-9]+         {yylval.str=strdup(yytext); print_token (yylval.str);
                    return(INTEGER);}
0[xX][0-9A-Fa-f]+ {yylval.str=strdup(yytext); print_token (yylval.str);
                    return (HEX_INTEGER); }
0[bB][0-1]+         {yylval.str=strdup(yytext); print_token (yylval.str);
                    return (BINARY_INTEGER); }
[-+]?([0-9]+([eE][-+]?[0-9]+)|([0-9]+\.[0-9]*([eE][-+]?[0-9]+)?))|(\.[0-
9]+([eE][-+]?[0-9]+)?)  { yylval.str=strdup(yytext);
                          print_token (yylval.str);
                          return(REAL);}

.                   {return(-1);}

%%
// This routine is the standard error trap if a lexical error occurs.

int yyerror(char* s)
{
      fprintf(stderr, "ERROR: %s at '%s' in statement %d.\n",
                    s,yytext, statement_count);
      fprintf(messages, "ERROR: %s at '%s' in statement %d.\n",
                    s,yytext, statement_count);
    return 0;
}
```

C.2 Formal grammar without semantic actions (partial bison input file)

This subclause lists the bison input file defining the formal grammar. It has been stripped of semantic action routines to allow an easy reference of just the syntax.

```
%token <str> REAL
%token <str> INTEGER
%token <str> QUOTED_STRING
%token <str> HEX_INTEGER
%token <str> BINARY_INTEGER
%token <str> PROPERTYNAME
%token <str> VARIABLE_NAME
%token <str> TEMPLATE
%token <str> COMMENT
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
%token <str> ABSTRACT

%type <str> size boundary template_statement
%type <str> numeric case_bit_value description
%type <str> selectcase_number_of_bits startvalue tolerance
%type <str> manufacturer_code id_number_of_bits template_id

%token VALIDATION_KEYCODE TDL_VERSION_NUMBER UGID SPACING
%token ENDTEMPLATE SELECTCASE ENDSELECT CASE ENDCASE
%token STRUCTARRAY ENDSTRUCTARRAY
%token ID CAL USR ALIGN
%token UNINT CHR5 DATE CONRELRES CONRES
%token STRING5 STRING7 STRING16
%token ASCII SINGLE UNICOD // UNICODE conflicts with a C++ variable
%token END_OF_LINE
%token DEFAULT INITIALIZE READWRITE CTRLFUNCTIONMASK
%token FUNCTIONTYPE FUNCTION
%token ENUMERATE PHYSICAL_UNIT

%%

template_list : template between_templates_list template_list
              | VALIDATION_KEYCODE_statement
between_templates_list : | between_templates between_templates_list
between_templates : comment_statement | eol

template : template_statement statement_list ENDTEMPLATE
template_statement : TEMPLATE ManufacturerID comma
                   id_number_of_bits comma template_id ','
                   optional_description eol optional_eol

manufacturer_code : INTEGER
id_number_of_bits : INTEGER
template_id : INTEGER
optional_description : | description

statement_list : | statement optional_eol statement_list
statement :  abstract_statement eol
             | align_statement
             | comment_statement
             | enumerate_statement
             | physical_unit_statement
             | property_definition_command
             | selectcase_statement
             | SPACING
             | structarray_statement
             | TDL_VERSION_NUMBER_statement
             | UGID_statement

abstract_statement      : ABSTRACT
align_statement         : ALIGN boundary
comment_statement       : COMMENT

enumerate_statement     : ENUMERATE VARIABLE_NAME enumerate_list
enumerate_list          : comma enumerate_item optional_enumerate_item
optional_enumerate_item : | comma enumerate_item optional_enumerate_item
enumerate_item          : QUOTED_STRING
                        | VARIABLE_NAME
                        | numeric
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
physical_unit_statement : PHYSICAL_UNIT QUOTED_STRING comma dot2_unit

property_definition_command : property comma optional_prop_description
                              comma accesslevel comma size comma
                              datatype comma format ','
                              physical_unit optional_assignment

property : PROPERTYNAME optional_subproperty

optional_subproperty :  | '[' subproperty ']' optional_eol
subproperty : DEFAULT
            | INITIALIZE
            | READWRITE
            | CTRLFUNCTIONMASK
            | QUOTED_STRING
            | FUNCTIONTYPE
            | FUNCTION

optional_prop_description :
                  | description
                  | PROPERTYNAME '[' QUOTED_STRING ']'

accesslevel :     | ID
                  | CAL
                  | USR

datatype :   UNINT
            | CHR5
            | DATE
            | CONRELRES comma startvalue comma tolerance
            | CONRES comma startvalue comma tolerance
            | ASCII
            | STRING5
            | STRING7
            | STRING16
            | SINGLE
            | UNICOD
            | VARIABLE_NAME

format : | QUOTED_STRING

physical_unit :   | QUOTED_STRING
                  | dot2_unit
dot2_unit : left_paren numeric
            comma numeric
            comma numeric
            comma numeric
            comma numeric
            comma numeric
            comma numeric
            comma numeric
            comma numeric
            comma numeric
            comma numeric
            comma numeric
            right_paren

optional_assignment :| equals assignment
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
assignment : HEX_INTEGER
           | BINARY_INTEGER
           | VARIABLE_NAME
           | numeric
           | QUOTED_STRING

selectcase_statement : SELECTCASE description comma accesslevel comma
                       selectcase_number_of_bits
                       case_statement_list ENDSELECT
case_statement_list : | case_statement case_statement_list
case_statement : CASE description comma case_bit_value
                 optional_eol statement_list
                 ENDCASE optional_eol

structarray_statement : STRUCTARRAY VARIABLE_NAME
                        comma description comma accesslevel comma size
                        optional_eol
                        struct_block ENDSTRUCTARRAY

struct_block : | struct_block_statement struct_block
struct_block_statement : property_definition_command optional_eol
                       | structarray_statement optional_eol

TDL_VERSION_NUMBER_statement  : TDL_VERSION_NUMBER INTEGER
UGID_statement : UGID QUOTED_STRING comma QUOTED_STRING

VALIDATION_KEYCODE_statement : VALIDATION_KEYCODE INTEGER

description : QUOTED_STRING

case_bit_value :              INTEGER
selectcase_number_of_bits :   INTEGER
size :                        INTEGER
boundary :                    INTEGER

startvalue : numeric
tolerance : numeric

numeric : REAL | INTEGER

comma       : ',' optional_eol
equals      : '=' optional_eol
left_paren  : '(' optional_eol
right_paren : optional_eol ')'

eol : END_OF_LINE
optional_eol : | eol optional_eol
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

C.3 Formal grammar with semantic actions (bison input file)

This subclause lists the bison input file defining the formal grammar. It includes semantic actions as C++ calls using the bison convention (Stroustrup [B13]). A simple set of semantic actions is listed in C.4.

```
%{

// ------------------------------------------------------------------
// TDL.y
//
```

© IEEE 2004 – All rights reserved

```
// This file is a standard input to the compiler compiler bison
// (which is a variant of yacc). It provides a formal grammar for
// the IEEE 1451.4 Template Description Language (TDL).  To use
// this file, you must first create the file lexyy.cpp using the
// file tdl.l (which contains the token definitions).  You then
// run this file through bison by executing the command line:
//
//          bison -d tdl.y
//
// After executing this command do:
//
//          rename tdl_tab.c tdl_tab.cpp.
//
// You can then compile the main TDL compiler using a C++ compiler.
//
// Note that the stack generated by bison in tdl_tab.c may not
// be big enough.  The sympton of this is that the parser will
// appear to terminate properly but will not have completed
// parsing the input file. It might be necessary, inside
// tdl_tab.cpp, to change
//
//    #define YYINITDEPTH 200
//
// to
//
//    #define YYINITDEPTH 500
//
// One aspect of the design of this grammar that could probably be
// done better is the use of end_of_line.  The need to worry about this
// as something other than white space is a consequence of both the
// comment and abstract commands that use end of line as a delimiter.
//
// Updated 10 September 2003
//
// ---------------------------------------------------------------------

#include "tdlparser.h"  // Main header file for TDL compiler.

// ---------------------------------------------------------------------
// lex and yacc stuff
// ---------------------------------------------------------------------

extern int yyerror (char*);
extern int yylex ();

// ---------------------------------------------------------------------
// debug variables
// ---------------------------------------------------------------------

extern int statement_count;
extern void print_statement_num ();

// ---------------------------------------------------------------------
// semantic action routine prototypes
// ---------------------------------------------------------------------

extern void initialize_parser ();
extern void finalize_parser ();
extern void initialize_template ();
extern void finalize_template ();
extern void process_abstract_statement();
extern void process_align ();
extern void process_case ();
extern void process_case_end ();
extern void process_comment();
extern void process_enumerate ();
extern void process_enumerate_item ();
extern void process_physical_unit ();
extern void process_property_definition ();
extern void process_selectcase ();
extern void process_endselect ();
extern void process_spacing ();
extern void process_structarray_statement();
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
extern void process_endstructarray_statement();
extern void process_subproperty ();
extern void process_template_statement();
extern void process_tdl_version_number ();
extern void process_ugid ();
extern void process_validation_keycode ();

%}

// ------------------------------------------------------------------------
// All tokens sent on to the action routines are captured
// as strings.  It is up to the action routines to convert
// to appropriate data types.
// ------------------------------------------------------------------------

%union {
  double real;
  char* str;
  int integer;
}

%token <str> REAL
%token <str> INTEGER
%token <str> QUOTED_STRING
%token <str> HEX_INTEGER
%token <str> BINARY_INTEGER
%token <str> PROPERTYNAME
%token <str> VARIABLE_NAME
%token <str> TEMPLATE
%token <str> COMMENT
%token <str> ABSTRACT

%type <str> size boundary template_statement
%type <str> numeric case_bit_value description
%type <str> selectcase_number_of_bits startvalue tolerance
%type <str> manufacturer_code id_number_of_bits template_id

%token VALIDATION_KEYCODE TDL_VERSION_NUMBER UGID SPACING
%token ENDTEMPLATE SELECTCASE ENDSELECT CASE ENDCASE
%token STRUCTARRAY ENDSTRUCTARRAY
%token ID CAL USR ALIGN
%token UNINT CHR5 DATE CONRELRES CONRES
%token STRING5 STRING7 STRING16
%token ASCII SINGLE UNICOD // UNICODE conflicts with a C++ variable
%token END_OF_LINE
%token DEFAULT INITIALIZE READWRITE CTRLFUNCTIONMASK
%token FUNCTIONTYPE FUNCTION
%token ENUMERATE PHYSICAL_UNIT

%%

template_list : {initialize_parser ();} template between_templates_list
                template_list
              | VALIDATION_KEYCODE_statement
              {process_validation_keycode (); finalize_parser ();}
between_templates_list : | between_templates between_templates_list
between_templates : comment_statement | eol

template : {initialize_template ();} template_statement statement_list
           ENDTEMPLATE {finalize_template ();}
template_statement : TEMPLATE manufacturer_code comma
                     id_number_of_bits comma template_id ','
                     optional_description {process_template_statement();}
                     eol optional_eol

manufacturer_code : INTEGER {strcpy (manufacturer_code, $1);}
id_number_of_bits : INTEGER {strcpy (id_number_of_bits, $1);}
template_id : INTEGER {strcpy (template_id, $1);}
optional_description : {description = NULL;} | description {}

statement_list : | statement optional_eol statement_list
statement :  abstract_statement {process_abstract_statement();} eol
           | align_statement {process_align ();}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
                | comment_statement
                | enumerate_statement
                | physical_unit_statement
                | property_definition_command
                    {process_property_definition ();}
                | selectcase_statement
                | SPACING {process_spacing ();}
                | structarray_statement
                | TDL_VERSION_NUMBER_statement
                    {process_tdl_version_number ();}
                | UGID_statement {process_ugid ();}

abstract_statement      : ABSTRACT {abstract = $1;}
align_statement         : ALIGN boundary
comment_statement       : COMMENT {comment = $1; process_comment();}

enumerate_statement     : ENUMERATE VARIABLE_NAME {variable_name = $2;
                            process_enumerate();} enumerate_list
enumerate_list          : comma enumerate_item optional_enumerate_item
optional_enumerate_item : | comma enumerate_item optional_enumerate_item
enumerate_item          : QUOTED_STRING {variable_name = $1;
                            strcpy          (numeric,           "");
process_enumerate_item ();}
                        | VARIABLE_NAME {variable_name = $1;
                            strcpy          (numeric,           "");
process_enumerate_item ();}
                        | numeric {variable_name = NULL;
                            strcpy          (numeric,           $1);
process_enumerate_item ();}

physical_unit_statement : PHYSICAL_UNIT QUOTED_STRING comma dot2_unit
                        {variable_name = $2; process_physical_unit ();}

property_definition_command : property comma optional_prop_description
                    comma accesslevel comma size comma datatype comma
                    format ',' physical_unit optional_assignment

property : PROPERTYNAME {propertyname = $1;} optional_subproperty {}

optional_subproperty : {strcpy(subproperty, ""); register_mask = NULL;}
                        | '[' subproperty ']'
                        {process_subproperty();} optional_eol
subproperty : DEFAULT {strcpy(subproperty, "DEFAULT"); }
              | INITIALIZE {strcpy(subproperty, "INITIALIZE");}
              | READWRITE {strcpy(subproperty, "READWRITE");}
              | CTRLFUNCTIONMASK {strcpy(subproperty,
                  "CTRLFUNCTIONMASK");}
              | QUOTED_STRING {register_mask = $1;}
              | FUNCTIONTYPE {strcpy(subproperty, "FUNCTIONTYPE");}
              | FUNCTION {strcpy(subproperty, "FUNCTION");}

optional_prop_description : {description = NULL;
                             function_propertyname = NULL;
                             function_register_mask = NULL}
                    | description {function_propertyname = NULL;
                        function_register_mask = NULL}
                    | PROPERTYNAME {description = NULL;
                        function_propertyname = $1;} '[' QUOTED_STRING
                        {function_register_mask = $4;} ']'

accesslevel : {strcpy(access_level,"");}
                    | ID {strcpy(access_level,"ID");}
                    | CAL {strcpy(access_level,"CAL");}
                    | USR {strcpy(access_level, "USR");}

datatype :   UNINT      {strcpy(datatype, "UNINT");}
             | CHR5     {strcpy(datatype, "CHR5");}
             | DATE     {strcpy(datatype, "DATE");}
             | CONRELRES comma startvalue comma tolerance
                        {strcpy(datatype, "ConRelRes");}
             | CONRES comma startvalue comma tolerance
                        {strcpy(datatype, "ConRes");}
             | ASCII    {strcpy(datatype, "ASCII");}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
                | STRING5    {strcpy(datatype, "String5");}
                | STRING7    {strcpy(datatype, "String7");}
                | STRING16   {strcpy(datatype, "String16");}
                | SINGLE     {strcpy(datatype, "Single");}
                | UNICOD     {strcpy(datatype, "Unicode");}
                | VARIABLE_NAME {strcpy(datatype, "Enumeration");}

format : {format = NULL;} | QUOTED_STRING {format = $1;}

physical_unit : {physical_unit = NULL;}
                | QUOTED_STRING {physical_unit=$1;}
                | dot2_unit {physical_unit = NULL}
dot2_unit : left_paren numeric {strcpy(dot2_1, $2);}
            comma numeric {strcpy(dot2_2, $5);}
            comma numeric {strcpy(dot2_3, $8);}
            comma numeric {strcpy(dot2_4, $11);}
            comma numeric {strcpy(dot2_5, $14);}
            comma numeric {strcpy(dot2_6, $17);}
            comma numeric {strcpy(dot2_7, $20);}
            comma numeric {strcpy(dot2_8, $23);}
            comma numeric {strcpy(dot2_9, $26);}
            comma numeric {strcpy(dot2_10, $29);}
            comma numeric {strcpy(dot2_11, $32);}
            comma numeric {strcpy(dot2_12, $35);}
            right_paren

optional_assignment : {assignment_found = false;}
                    | equals {assignment_found = true;} assignment
assignment : HEX_INTEGER {hex_integer = $1;}
             | BINARY_INTEGER {binary_integer = $1;}
             | VARIABLE_NAME {variable_name = $1;}
             | numeric {strcpy (numeric,$1);}
             | QUOTED_STRING {assignment_string = $1;}

selectcase_statement : SELECTCASE description comma accesslevel comma
                       selectcase_number_of_bits
                       {process_selectcase ();} optional_eol
                       case_statement_list ENDSELECT
                       {process_endselect ();}
case_statement_list : | case_statement case_statement_list
case_statement : CASE description comma case_bit_value
                 {process_case ();} optional_eol statement_list
                 ENDCASE {process_case_end();} optional_eol

structarray_statement : STRUCTARRAY VARIABLE_NAME {variable_name = $2;}
                        comma description comma accesslevel comma size
                        {process_structarray_statement ();} optional_eol
                        struct_block ENDSTRUCTARRAY
                        {process_endstructarray_statement ();}
struct_block : | struct_block_statement struct_block
struct_block_statement : property_definition_command optional_eol
                         {process_property_definition ();}
                         | structarray_statement optional_eol

TDL_VERSION_NUMBER_statement  : TDL_VERSION_NUMBER INTEGER
                                {strcpy (tdl_version_number, $2);}


UGID_statement : UGID QUOTED_STRING {ugid=$2;}
                         comma description

VALIDATION_KEYCODE_statement : VALIDATION_KEYCODE INTEGER
                                {strcpy (validation_keycode, $2);}

description : QUOTED_STRING {description=$1;}

case_bit_value : INTEGER {strcpy (case_bit_value, $1);}
selectcase_number_of_bits : INTEGER
                         {strcpy (selectcase_number_of_bits, $1);}
size : INTEGER {strcpy (size,$1);}
boundary : INTEGER {strcpy (boundary,$1);}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
startvalue : numeric {strcpy (startvalue,$1);}
tolerance : numeric {strcpy (tolerance,$1);}

numeric : REAL | INTEGER

comma       : ',' optional_eol
equals      : '=' optional_eol
left_paren  : '(' optional_eol
right_paren : optional_eol ')'

eol : END_OF_LINE {statement_count++; print_statement_num();}
optional_eol : | eol optional_eol

%%
```

C.4 C++ header file

This subclause lists the file that includes common variable and routine definitions needed for implementing the semantic routines.

```
//////////////////////////////////////////////////////
// tdlparser.h - IEEE 1451.4 TDL grammar header file
//
// 27 January 2003
// Updated 10 September 2003
//
// This file declares common variables and other header
// files used by the Template Description Language (TDL)
// parser routines.  It is intended to be used in conjunction
// with tdl.l, tdl.y, tdlmain.cpp, and tdlparser.cpp.
//
//////////////////////////////////////////////////////

#include <stdio.h>
#include <afx.h>
#include <afxcoll.h>
#include <string.h> // Use standard C strings for general compatability.

// Define this to avoid having to do dynamic allocation.

#define MAX_STRING 100
#define MAX_TEMPLATES 1000
#define MAX_STRUCTS 10
#define MAX_SELECTS 10
#define MAX_CASES 50

// Token string values returned to parser routines from bison (yacc).

extern char* abstract;
extern char* assignment_string;
extern char* binary_integer;
extern char* comment;
extern char* description;
extern char* ugid;
extern char* format;
extern char* hex_integer;
extern char* physical_unit;
extern char* propertyname;
extern char* variable_name;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
extern char* register_mask;
extern char* function_propertyname;  // Used with subproperty function.
extern char* function_register_mask; // Used with subproperty function.

// The token numeric values returned by bison can not be just
// by pointer. They must be copied into a string or else they may
// be overwritten before we get to the end of the statement.
// This is not true for the string values above.

extern char value[MAX_STRING];
extern char startvalue[MAX_STRING];
extern char tolerance[MAX_STRING];
extern char boundary[MAX_STRING];
extern char numeric[MAX_STRING];
extern char size[MAX_STRING];
extern char case_bit_value[MAX_STRING];
extern char selectcase_number_of_bits[MAX_STRING];
extern char tdl_version_number[MAX_STRING];
extern char validation_keycode[MAX_STRING];
extern char access_level[MAX_STRING];
extern char datatype[MAX_STRING];
extern char subproperty[MAX_STRING];
extern char manufacturer_code [MAX_STRING];
extern char id_number_of_bits [MAX_STRING];
extern char template_id [MAX_STRING];

extern char dot2_1[MAX_STRING];
extern char dot2_2[MAX_STRING];
extern char dot2_3[MAX_STRING];
extern char dot2_4[MAX_STRING];
extern char dot2_5[MAX_STRING];
extern char dot2_6[MAX_STRING];
extern char dot2_7[MAX_STRING];
extern char dot2_8[MAX_STRING];
extern char dot2_9[MAX_STRING];
extern char dot2_10[MAX_STRING];
extern char dot2_11[MAX_STRING];
extern char dot2_12[MAX_STRING];

// Globals not tokens

extern bool assignment_found;
extern FILE *tokens;
extern FILE *messages;

// debug variables

extern int statement_count;
```

C.5 Main C++ code routines

This subclause lists the main routines for implementing the TDL parser.

```
////////////////////////////////////////////////////////////////////
// tdlmain.cpp
//
// IEEE 1451.4 Template Description Language (TDL) syntax checker.
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
//
// 27 January 2003
// Updated 10 September 2003
//
// This program uses code generated by flex and bison to parse
// TDL templates as defined by the IEEE 1451.4 standard.  The files
// used as input to flex and bison are tdl.l and tdl.y.  Those files
// contain formal token and grammar definitions.
//
// The main objective of this program is to provide a syntax checker
// for templates.  However, it could also be used as a basis for a
// TEDS parser, or as a portion of a T-Block implementation.
//
// The basic design is simply to parse through the template a token
// at a time.  At appropriate places in the parse, usually at the end
// of a template command, a semantic action routine is invoked.  These
// routines mostly just output the line number, the command type, and
// any parameters associated with the command.  An indication of when
// to read bits from the TEDS and some other minor activity suggestions
// are occasionaly provided.
//
// Note that this program does not check the syntax of the
// quoted format field in property commands.
//
// No verification is made that enumerated types used in property
// commands have been previously defined.
//
// Property command names are verified against the initial property
// list defined in the standard.  Any unrecognized properties are
// listed at the end of the message file.
//
// The minimum and maximum number of bits for each template is output
// at the end of the message file.  However, because of the complexity
// of calculating options regarding selectcase statements,
// string types, free form commands, align commands, etc., there is
// no guarentee that these calculations are correct.  Indeed, some
// of the commands do not allow an exact calculation to be made.
//
// Other C++ files needed are:
//
//    tdlparser.h - the main header file declaring global variables
//    tdlparser.cpp - the semantic action routines that implement
//    processing of the template commands.
//
//
// Input:
//
//    The template file name and associated template file.  The default
//    filename is "Default.tdl".
//
// Outputs:
//
//    messages.dat - A file containing descriptions of each statement
//    in the template file.
//
//    tokens.dat - A file containing the parsed tokens.  This is most
//    useful for debugging since it provides a specific indication of
//    what tokens were parsed.
//
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
// Initially programmed by Charles H. Jones March 2001.
//
/////////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <afx.h>
#include <afxcoll.h>

extern int yyparse();          // standard yacc/bison routine
extern FILE *yyin;             // standard yacc/bison input file

FILE *tokens;                  // output file containing tokens
FILE *messages;                // output file with all messages
int statement_count = 1;


/////////////////////////////////////////////////////////////////////
// Main function
/////////////////////////////////////////////////////////////////////
void main (int argc,char **argv)

{
  char filename[100];

  ////////////////////////////////////////
  // Open the messages file
  ////////////////////////////////////////

  strcpy (filename, "message.dat");
  if ( (messages=fopen(filename, "w")) == NULL) {
   printf ("ERROR: DL Parser Main: Can not open file %s\n", filename);
   exit(1);
  };

  printf ("DL Parser program.\n");
  fprintf (messages, "DL Parser program.\n");

  ////////////////////////////////////////
  // Open the token file
  ////////////////////////////////////////

  strcpy (filename, "tokens.dat");
  if ( (tokens=fopen(filename, "w")) == NULL) {
   printf ("ERROR: DL Parser Main: Can not open file %s\n", filename);
   exit(1);
  };

  fprintf (tokens, "DL Parser program.\n");

  ////////////////////////////////////////
  // Open the input file
  ////////////////////////////////////////

  if (argc == 1) {
    strcpy (filename, "default.tdl");
  }else{
    strcpy (filename, argv[1]);
  };
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  printf ("Reading from file '%s'.\n", filename);
  fprintf (messages, "Reading from file '%s'.\n", filename);
  fprintf (tokens, "Reading from file '%s'.\n", filename);

  yyin=fopen(filename, "r");
  if (yyin == NULL) {
    perror(filename);
    exit(1);
  }

  ////////////////////////////////////
  // parse the file
  ////////////////////////////////////

  yyparse();
  fclose(yyin);

  fclose (tokens);
  fprintf (messages, "Done\n");
  fclose (messages);
  printf ("Done\n");

} // end main
```

C.6 C++ code for implementing semantic actions

This subclause lists the file that contains the semantic action routines.

```
////////////////////////////////////////////////////////////////
// tdlparser.cpp
//
// IEEE 1451.4 TDL parser semantic action routines.
//
// 27 January 2003
// Updated 10 September 2003
//
// These routines work in conjuction with tdlmain.cpp, tdlparser.h,
// tdl.l, and tdl.y.  These last two files are inputs to flex and
// bison which generate lexyy.cpp, tdl_tab.h, and tdl_tab.cpp.
// The routines contained in this file are identified in tdl.y and
// are then generated into tdl_tab.cpp.
//
// The intent of these routines is to illustrate the
// basic flow of the parser as it reads through a template.
// Comments describing this flow are output to the message.dat file.
// These comments include indicating how many bits are read
// and what other actions should have taken place at that point
// during the parse.
//
// The basic structure of these routines is for there to be an
// action routine for each TDL command along with some support
// routines for these commands.  The routines mostly identify the
// particular command and output the associated parameters that were
// read from the template.  There are also start and end
// routines for the template file itself and for each template.
//
// Someone intending to use this program as a basis for a larger
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
// program would spend most of their efforts in modifying the
// routines in this file.  All of the information needed for each
// command is available within the routine that processes that command.
// One particular function that would have to be performed would be
// to translate numeric values into numeric variables since everything
// is handled as strings in these routines.
//
// One global function included in these routines is to verify property
// names (this is not done in the grammar itself).  When an unknown
// property is found, it is output to the messages file.  But it is
// also recorded in an array of unknown properties.  This array is
// output to the messages.dat file at the end of the template file.
//
// Note that one parsing functionalility that is not implemented here
// nor in the grammar is checking the proper syntax of format strings.
// (That is, the suggested format for output of a property value.)
//
// Note that the validation_keycode is not calculated.
//
// There is also an attemnpt to calculate the minimum and maximum
// number of bits in a template.  This adds a fair amount of
// complexity to the code in some places.  For example, keeping
// track of mins and maxs as a consquence of selectcase commands
// requires some careful programming.  Similarly, nested
// structarrays cause some difficulty with bit counting.
//
////////////////////////////////////////////////////////////////////////

#include "tdlparser.h"
#include <math.h>

// Token string values.  These are assigned in the grammar.

char* propertyname = NULL;
char* variable_name = NULL;
char* description = NULL;
char* ugid = NULL;
char* format = NULL;
char* physical_unit = NULL;
char* hex_integer = NULL;
char* binary_integer = NULL;
char* abstract = NULL;
char* comment = NULL;
char* assignment_string = NULL;
char* register_mask = NULL;
char* function_propertyname = NULL;  // Used with subproperty function.
char* function_register_mask = NULL; // Used with subproperty function.

// The token numeric values returned by bison (yacc) can not be just
// by pointer.  They must be copied into a string or else they may be
// overwritten before we get to the end of the statement. This is not
// true for the string values above.

char value[MAX_STRING] = "";
char startvalue [MAX_STRING] = "";
char tolerance [MAX_STRING] = "";
char numeric [MAX_STRING] = "";
char boundary [MAX_STRING] = "";
char size [MAX_STRING] = "";
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
char case_bit_value [MAX_STRING] = "";
char selectcase_number_of_bits [MAX_STRING] = "";
char tdl_version_number [MAX_STRING] = "";
char access_level[MAX_STRING] = "";
char datatype[MAX_STRING] = "";
char subproperty[MAX_STRING] = "";
char validation_keycode[MAX_STRING] = "";
char manufacturer_code [MAX_STRING] = "";
char id_number_of_bits [MAX_STRING] = "";
char template_id [MAX_STRING] = "";

char dot2_1[MAX_STRING] = "";
char dot2_2[MAX_STRING] = "";
char dot2_3[MAX_STRING] = "";
char dot2_4[MAX_STRING] = "";
char dot2_5[MAX_STRING] = "";
char dot2_6[MAX_STRING] = "";
char dot2_7[MAX_STRING] = "";
char dot2_8[MAX_STRING] = "";
char dot2_9[MAX_STRING] = "";
char dot2_10[MAX_STRING] = "";
char dot2_11[MAX_STRING] = "";
char dot2_12[MAX_STRING] = "";

// Non token variables.

bool assignment_found = false;

// Stuff to keep track of template statistics.

int min_number_bits[MAX_TEMPLATES];
int max_number_bits[MAX_TEMPLATES];
int template_ids[MAX_TEMPLATES];
int manufacturer_codes[MAX_TEMPLATES];
char template_titles[MAX_TEMPLATES][MAX_STRING];
int template_index = 0;

// Keep track of statements that cause uncertainty in bit counts.

bool free_form_found[MAX_TEMPLATES];
bool string_types_found[MAX_TEMPLATES];
bool align_found[MAX_TEMPLATES];

// Stuff to help figure out how many bits for a structarray.

int structarray_level = -1;
int structarray_max_bits[MAX_STRUCTS];
int structarray_size[MAX_STRUCTS];

// Stuff to help figure out how many bits for a selectcase.

int select_level = -1;
int case_number = 0;
int case_min_bits[MAX_SELECTS][MAX_CASES];
int case_max_bits[MAX_SELECTS][MAX_CASES];
int  select_bits[MAX_SELECTS];  //  the  select_number_of_bits  for  each
level

// Keep track of what properties are not known.
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
int unknown_prop_index = -1;
char unknown_properties[1000][MAX_STRING];

// The property list.

char property_list [][MAX_STRING] = {"%ACDCCoupling",
"%Appended_TEDS", "%Appended_TEDS_location",
"%Attached_MfgrID","%Attached_ModelNum",
"%Attached_SerialNum","%Attached_VersionLetter",
"%Attached_VersionNum","%BridgeTopology","%BridgeType",
"%CableLen","%CalCurve_Coef","%CalCurve_Domain",
"%CalCurve_PieceStart","%CalCurve_Power","%CalDate",
"%CalInitials","%CalPeriod","%CalPoint_DomainValue",
"%CalPoint_RangeValue","%CalTable_Domain","%Ccable",
"%Cin","%CJSource","%Cmic","%Cstray",
"%DefaultFR","%Direction","%DiscSigAmpl","%DiscSigType",
"%ElecSigType","%EmbTpl","%Equi_Vol","%ExciteAmplMax",
"%ExciteAmplMin","%ExciteAmplNom","%ExciteCurrentDraw",
"%ExciteFreqMax","%ExciteFreqMin","%ExciteFreqNom",
"%filter","%ExciteType","%Gain","%GageArea","%GageFactor",
"%GageOffset","%GageTransSens","%GageType", "%Gate",
"%LoopSupplyMax","%LoopSupplyMin","%MapMeth","%Mass_Below",
"%MaxElecVal","%MaxPhysVal","%MeasID","%MemberIndex",
"%MicSize","%MicType","%MinElecVal","%MinPhysVal",
"%Passive","%PhaseCorrection","%PhysicalParameterType",
"%PoissonCoef","%PollFreq","%Prepolarized",
"%PulseMeasType","%Reffreq","%RefPol","%RefPress",
"%RefTemp","%RespTime","%Resp_Type","%Rin","%Rleakage",
"%Rout","%RTDCoef_A","%RTDCoef_B","%RTDCoef_C",
"%RTDCoef_R0","%SelfHeating","%Sens","%Sens@Ref",
"%SensorImped","%Sign","%SteinhartA","%SteinhartB",
"%SteinhartC","%Stiffness","%System_MfgrID",
"%System_ModelNum","%System_VersionLetter",
"%System_VersionNum","%System_SerialNum",
"%TCType","%TDL_Checksum","%TempCoef",
"%TestGain", "%TF_HP_S","%TF_KPq","%TF_KPr",
"%TF_KZq","%TF_KZr","%TF_SL","%TF_SP","%TF_SPm","%TF_SZ",
"%TF_SZm","%TF_Table_Ampl","%TF_Table_Freq","%user",
"%Weight","%XML", "%YoungsMod",
"END_OF_LIST"};

// function prototypes

void print_bit_count (int, int);

////////////////////////////////////////////////////////////////////
// initialize_parser
//
// Initialize the parser.  This is executed prior to
// reading anything from the template file.
//
// global inputs: none
//
// global outputs:
//
//          free_form_found
//          string_types_found
//          align_found
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
//
//////////////////////////////////////////////////////////////////////
void initialize_parser ()

{

free_form_found[template_index] = false;
string_types_found[template_index] = false;
align_found[template_index] = false;

} // end initialize_parser

//////////////////////////////////////////////////////////////////////
// finalize_parser
//
// Finish off anything needed to be done.  This is executed after the
// VALIDATION_KEYCODE.
//
// This outputs template statistics and the unknown propeties.
//
// global inputs: messages, unknown_prop_index, unknown_properties
//
// global outputs: messages and list of unknown
//                 properties to message.dat
//
//////////////////////////////////////////////////////////////////////
void finalize_parser ()

{

  fprintf (messages, "Template file parse completed.\n\n");

  // Output template statistics.

  for (int i=0; i<template_index; i++) {

      fprintf (messages, "ManCode=%d ID=%d MinBits=%d MaxBits=%d ",
                manufacturer_codes[i], template_ids[i],
                min_number_bits[i], max_number_bits[i]);

    if (align_found[i]) {
        fprintf (messages, "Align found. ");
      }

    if (free_form_found[i]) {
      fprintf (messages, "Free form statements found. ");
      }

    if (string_types_found[i]) {
        fprintf (messages, "String types found.");
      }

    fprintf (messages, "\n");

  } // end for


  // Output unknown properties.
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  if (unknown_prop_index > 0) {

      printf ("\nThe following properties were not known:\n");
      fprintf   (messages,   "\nThe   following   properties   were   not
known:\n");

    int index = 0;
      while (index <= unknown_prop_index) {

        printf ("%s\n", unknown_properties[index]);
      fprintf (messages, "%s\n", unknown_properties[index]);
      index++;

      } // end while

  } // end if

} // end finalize_parser

////////////////////////////////////////////////////////////////////
// initialize_template
//
// Set up for a new template.  This is executed prior to
// reading anything from each template.
//
// Just initializes several variables so there's no carry over
// from the last template.
//
// global inputs: none
//
// global outputs:
//
//          hex_integer
//          binary_integer
//          variable_name
//          assignment_string
//          numeric
//
////////////////////////////////////////////////////////////////////
void initialize_template ()

{
  hex_integer = NULL;
  binary_integer = NULL;
  variable_name = NULL;
  assignment_string = NULL;
  strcpy (numeric, "");

} // end initialize_template

////////////////////////////////////////////////////////////////////
// finalize_template
//
// Finish off anything needed to be done for each template.
// This is executed after each ENDTEMPLATE.
//
// global inputs: manufacturere_codes, template_ids, min_number_bits,
//                max_number_bits, align_found, free_form_found
//                string_types_found
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
//
// global outputs: string_types_found, free_form_found, align_found
//                 messages, messages to message.dat, template_index
//
////////////////////////////////////////////////////////////////////
void finalize_template ()

{

  fprintf (messages, "Finished reading template with manufacturer code
%d and ID = %d\n",
                  manufacturer_codes[template_index],
                  template_ids[template_index]);
  fprintf (messages, "Minimum number of bits = %d\n",
                  min_number_bits[template_index]);
  fprintf (messages, "Maximum number of bits = %d\n",
                  max_number_bits[template_index]);

  if (align_found[template_index]) {
      fprintf (messages, "There was an align statement so there is some
error in the above number of bits calculated.\n");
  }

  if (free_form_found[template_index]) {
      fprintf (messages, "There were free form statements so there the
maximum number of bits is indeterminate.\n");
  }

  if (string_types_found[template_index]) {
      fprintf (messages, "There were string types found which increases
the min to max range significantly.\n");
  }

  template_index++;
  free_form_found[template_index] = false;
  string_types_found[template_index] = false;
  align_found[template_index] = false;

} // end finalize_template


////////////////////////////////////////////////////////////////////
// known_property
//
// input: property name
//
// global inputs: property_list
//
// global outputs: none
//
// return value:  true if the property is on the known property list
//                false otherwise
//
////////////////////////////////////////////////////////////////////
bool known_property (char property[MAX_STRING])
{
bool known = 0;
int index = 0;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  while (   stricmp(property_list[index],"END_OF_LIST") != 0
         && !known) {

        if (stricmp(property_list[index], property) == 0) {
              known = 1;
        }
        index++;

  } // end while

return (known);

} // end known_property

////////////////////////////////////////////////////////////////////////
// process_property_definition
//
// A property definition command reads a value from
// the TEDS and assigns it to the specified property.
// Alternatively, the value can be assigned directly.
//
// Syntax:
//
// %<property_tag>,<"description">,<access_level>,<data_type>,
//    <format>,<physical_unit>
// or
// %<property_tag>,<"description">,<access_level>,<data_type>,
//    <format>,<physical_unit> = <value>
// or
// %<property_tag><[subproperty]>,<"description">,<access_level>,
//    <data_type>,<format>,<physical_unit>
// or
// %<property_tag><[subproperty]>,<"description">,<access_level>,
//    <data_type>,<format>,<physical_unit> = <value>
//
// For the purposes of the assignment (whether read from the TEDS or
// given explicitely), the description, access_level, and
// physical_unit are extraneous.  The size tells the number of
// bits to read. The datatype indicates what type of conversion
// of the bits is necessary.
//
// This routine does quite a lot.  However, it is broken down into
// independent sequential activities as indicated by the major
// comment headers. Mostly it just outputs messages indicating
// the values of the properties.  However, it also does some of
// the bit counting stuff.
//
// global Inputs:
//
//    propertyname, description, access_level, size, datatype,
//    physical_unit, hex_integer, binary_integer, messages, format
//    dot2_1 ... dot2_12, variable_name, assignment_string, numeric,
//    structarray_bit_count, unknown_prop_index, unknown_properties
//
// global outputs:
//
//    string_types_found, unknown_prop_index, hex_integer,
//    binary_integer, variable_name, numeric, assignment_string,
//    dot2_1
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
//    comments to the message file
//
//////////////////////////////////////////////////////////////////////
void process_property_definition ()

{
int local_bit_count = 0;

  ////////////////////////////////////////////////////
  // General for all properties
  ////////////////////////////////////////////////////

  fprintf (messages, "The property %s was just defined.\n",
           propertyname);

  if (description != NULL) {
    fprintf (messages, "It was described as:\n %s\n", description);
  }else{
     fprintf (messages, "It has a function subproperty association with:
%s[%s].\n",
      function_propertyname, function_register_mask);
  }

  fprintf (messages, "It has access level %s", access_level);

  if (format != NULL) {
    fprintf (messages, " and format %s,\n", format);
  }else{
    fprintf (messages, " and no format specified,\n");
  }

  if (physical_unit != NULL) {
    fprintf (messages, " and units %s.\n", physical_unit);
  }else if (stricmp(dot2_1, "") != 0) {
                    fprintf      (messages,      "      and      units
(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s).\n",
      dot2_1, dot2_2, dot2_3, dot2_4, dot2_5, dot2_6, dot2_7, dot2_8,
      dot2_9, dot2_10, dot2_11, dot2_12);
  }else{
    fprintf (messages, " and no units specified.\n");
  }


  ////////////////////////////////////////////////////
  // Different datatypes need to be handled differently.
  ////////////////////////////////////////////////////

  if ( (stricmp (datatype, "ConRelRes") == 0)
        || (stricmp (datatype, "ConRes") == 0) ) {

      fprintf (messages, "The data type is %s with start value %s and
tolerance %s.\n", datatype, startvalue, tolerance);

  }else if (stricmp (datatype, "Enumeration") == 0) {

      fprintf (messages, "The data type is an enumeration of %s.\n",
           variable_name);

  }else{
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
      fprintf (messages, "The data type is %s.\n", datatype);

  }

  ////////////////////////////////////////////////////
  // Is it an assignment or do you read from the TEDS?
  // Also increment min and max bits read variable.
  ////////////////////////////////////////////////////

  int integer_size = atoi(size);
  fprintf (messages, "The size parameter is %d\n", integer_size);

  if (!assignment_found) {

    // It's not an assignment. Read the bits from the TEDS.

    fprintf (messages, "The value of %s should be read from the TEDS.\n",
propertyname, size);

    ////////////////////////////////////////////////////
    // If it's a string type, have to multply the size by char
    // length to get max bits read. The min bits read is 0.
    ////////////////////////////////////////////////////

    if (stricmp (datatype, "string5") == 0) {

      string_types_found[template_index] = true;
      local_bit_count = (pow(2, integer_size) - 1.0)*5;

    }else if (stricmp (datatype, "string7") == 0) {

      string_types_found[template_index] = true;
      local_bit_count = (pow(2, integer_size) - 1.0)*7;

    }else if (stricmp (datatype, "string16") == 0) {

      string_types_found[template_index] = true;
      local_bit_count = (pow(2, integer_size) - 1.0)*16;

    }else{ // not a string type

      local_bit_count = integer_size;

    } // end if-then-else string type


    if (structarray_level > -1) {
      // we're in a structarray
      structarray_max_bits[structarray_level] += local_bit_count;
    }else{
      // we're not in a structarray
      print_bit_count (local_bit_count, local_bit_count);
    } // end if structarray_level

  }else{ // It's an assignment.

      if (hex_integer != NULL) {
         fprintf (messages, "The value of %s is stated as the hex value
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
%s.\n", propertyname, hex_integer);
      }else if (binary_integer != NULL) {
        fprintf (messages, "The value of %s is stated as the binary
value %s.\n", propertyname, binary_integer);
      }else if (variable_name != NULL) {
        fprintf (messages, "The value of %s is stated as the string
value %s.\n", propertyname, variable_name);
      }else if (assignment_string != NULL) {
        fprintf (messages, "The value of %s is stated as the string
value %s.\n", propertyname, assignment_string);
      }else{
         fprintf (messages, "The value of %s is stated as the decimal
value %s.\n", propertyname, numeric);
      } // end if hex_integer

  } // end if !assignment

  ////////////////////////////////////////////////////
  // Free form properties are special.
  ////////////////////////////////////////////////////

  if (   (   stricmp(propertyname,"%user")      == 0
          || stricmp(propertyname,"%asciiteds") == 0
          || stricmp(propertyname,"%xml")       == 0)
       && stricmp(size,"0") == 0) {

    fprintf (messages, "Since the bit size for this free form command is
0, the rest of the TEDS is user defined.\n");
    fprintf (messages, "The bits read from the TEDS should be treated as
%s.\n", datatype);
    free_form_found[template_index] = true;

  }else if (!known_property(propertyname)) {

      fprintf (messages, "WARNING: Property %s is not on the known
property list!\n", propertyname);
      unknown_prop_index++;
      strcpy (unknown_properties[unknown_prop_index], propertyname);

  } // end if %user

  ////////////////////////////////////////////////////
  // cleanup
  ////////////////////////////////////////////////////

  hex_integer = NULL;
  binary_integer = NULL;
  variable_name = NULL;
  strcpy (numeric, "");
  assignment_string = NULL;
  strcpy (dot2_1, "");

} // end process_property_definition

//////////////////////////////////////////////////////////////////////
// process_template_statement
//
// Syntax:
//
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
//    TEMPLATE <manufacturer code>, <ID number of bits>,
//          <Template ID>,<title>
//          <commands>
//    ENDTEMPLATE
//
// global inputs: messages, template_string, manufacturer_code
//                template_id, description, id_number_of_bits,
//                template_index
//
// global outputs: messages to message.dat, manufacturer_code,
//                 template_ids, template_titles
//
////////////////////////////////////////////////////////////////////
void process_template_statement ()

{

  fprintf (messages, "\nJust read Template statement.\n%");
  fprintf (messages, "Manufacturer code = %s\n", manufacturer_code);
  fprintf (messages, "Template ID = %s\n", template_id);
  fprintf (messages, "Title = %s\n", description);
  print_bit_count (atoi(id_number_of_bits), atoi(id_number_of_bits));

  manufacturer_codes[template_index] = atoi(manufacturer_code);
  template_ids[template_index] = atoi(template_id);
  strcpy(template_titles[template_index], description);

} // end process_template_statement

////////////////////////////////////////////////////////////////////
// process_abstract_statement
//
// Syntax:
//
//    ABSTRACT <description>
//
// global inputs: messages, abstract
//
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////
void process_abstract_statement ()

{

  fprintf (messages, "Just read abstract (no bits read):\n%s\n",
           abstract);

} // end process_abstract_statement

////////////////////////////////////////////////////////////////////
// process_align
//
// Syntax:
//
//    ALIGN <word_size>
//
// The tricky part about this is calculating the bits to skip.
// Since this is dependent on how many bits have already been read,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
// this is can not be determined without actually reading a TEDS.
// Thus, the bit counts have an error for any template containing
// an align command.
//
// global inputs: messages, boundary
//
// global outputs: messages to message.dat, align_found
//
////////////////////////////////////////////////////////////////////
void process_align ()

{

int memory_size = 255;
int boundary_int = atoi(boundary);
// current position, X, is the next bit to be read so need to
// add 1 to total_bits_read
int bits_to_skip = (memory_size - (min_number_bits[template_index]+1) +
1) % boundary_int;

  fprintf (messages, "Time to align the next bits to be read as if we
had %d sized bytes.\n", boundary_int);
  fprintf (messages, "As an illustration, assume the total memory size,
Y, is %d\n", memory_size);
  fprintf (messages, "We've already read %d bits (call this X).\n",
           min_number_bits);
  fprintf (messages, "So calculate (Y-X+1) MOD# = (%d-%d+1)mod%d and
skip %d bits.\n",
           memory_size, min_number_bits[template_index]+1,
           boundary_int, bits_to_skip);

  print_bit_count (bits_to_skip, bits_to_skip);
  align_found[template_index] = true;

} // end process_align

////////////////////////////////////////////////////////////////////
// process_comment
//
// Syntax:
//
//    // <description>
//
// global inputs: messages, comment
//
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////
void process_comment ()

{
  fprintf (messages, "Just read comment (no bits read):\n%s\n",
           comment);

} // end process_comment

////////////////////////////////////////////////////////////////////
// process_enumerate
//
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
// Syntax:
//
//    Enumerate <defined_type>, <enum_list>
//
// global inputs: messages, variable_name
//
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////
void process_enumerate ()

{
  fprintf (messages, "The Enumerate class %s is defined as:\n",
           variable_name);

} // end process_enumerate

////////////////////////////////////////////////////////////////////
// process_enumerate_item
//
// global inputs: messages, numeric, variable_name
//
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////
void process_enumerate_item ()

{
  if (variable_name != NULL) {
    fprintf (messages, "%s\n", variable_name);
  }else{
    fprintf (messages, "%s\n", numeric);
  }

} // end process_enumerate_item

////////////////////////////////////////////////////////////////////
// process_physical_unit
//
// global inputs: messages, variable_name, dot2_1 ... dot2_12
//
// global outputs: messages to message.dat, dot2_1
//
////////////////////////////////////////////////////////////////////
void process_physical_unit ()

{
  fprintf (messages, "Physical_unit %s has been associated with",
                  variable_name);
  fprintf (messages, " (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s).\n",
           dot2_1, dot2_2, dot2_3, dot2_4, dot2_5, dot2_6, dot2_7,
           dot2_8, dot2_9, dot2_10, dot2_11, dot2_12);

  strcpy (dot2_1, "");

} // end process_physical_unit


////////////////////////////////////////////////////////////////////
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
// process_selectcase
//
// Syntax:
//
//    SELECTCASE <"description">, <access_level>, <number_of_bits>
//          CASE <"description">, <bit value>
//                <command list>
//          ENDCASE
//          ...
//          CASE <"description">, <bit value>
//                <command list>
//          ENDCASE
//    ENDSELECT
//
// global inputs: messages, description, access_level,
//                              selectcase_number_of_bits, select_level
//
// global outputs: messages to message.dat, select_bits, case_number,
//                              select_level
//
////////////////////////////////////////////////////////////////////////
void process_selectcase ()

{

  select_level++;

   fprintf (messages, "A selectcase statement with description %s\n and
access level %s has been read.\n",
                  description, access_level);
  fprintf (messages, "The next %s bits from the TEDS are the selectcase
value for the following cases.\n",
                  selectcase_number_of_bits);
  select_bits[select_level] = atoi(selectcase_number_of_bits);
  case_number = -1;

} // end process_selectcase

////////////////////////////////////////////////////////////////////////
// process_endselect
//
// The complexity here is that you have to figure out min and max
// bits based on all possible combinations of cases and nested
// selectcase statements.  The basic approach is to keep track
// of mins and maxs for each case and for each selectcase level.
// This is essentially a recursive routine implemented with arrays.
//
// global inputs: messages, case_min_bits, case_nax_bits, select_level
//
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////////
void process_endselect ()

{

  fprintf (messages, "End of selectcase statement.\n");

  // Find minimum bits out of all cases.
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  int min_bits = 100000;
  for (int i=0; i<=case_number; i++) {
      if (min_bits > case_min_bits[select_level][i]) {
        min_bits = case_min_bits[select_level][i];
      }
  } // end for

  // Find maximum bits out of all cases.

  int max_bits = 0;
  for (i=0; i<=case_number; i++) {
      if (max_bits < case_max_bits[select_level][i]) {
        max_bits = case_max_bits[select_level][i];
      }
  } // end for

  // Select_level is decremented because print_bit_count uses it to
  // decide whether it's the top level select or not.
  // But we still have to add in the number of bits read in
  // the selectcase.
  // So the select_bits index has to be upped by 1.

  select_level--;
  print_bit_count(min_bits+select_bits[select_level+1],
      max_bits+select_bits[select_level+1]);

} // end process_endselect

////////////////////////////////////////////////////////////////
// process_case
//
// global inputs: messages, description, case_bit_value, case_number,
//                          select_level
//
// global outputs: messages to message.dat, case_number, case_min_bits,
//                          case_max_bits
//
////////////////////////////////////////////////////////////////
void process_case ()

{

  fprintf (messages, "Case %s or bit value %s. No bits read.\n",
           description, case_bit_value);

  case_number++;
  case_min_bits[select_level][case_number] = 0;
  case_max_bits[select_level][case_number] = 0;

} // end process_case

////////////////////////////////////////////////////////////////
// process_case_end
//
// global inputs: messages
//
// global outputs: messages to message.dat
//
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
////////////////////////////////////////////////////////////////////
void process_case_end ()

{

  fprintf (messages, "End of case statement.\n");

} // end process_case_end

////////////////////////////////////////////////////////////////////
// process_spacing
//
// Syntax:
//
//    spacing
//
// global inputs: messages
//
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////
void process_spacing ()

{

  fprintf (messages, "Spacing statement (no bits read).\n");

} // end process_spacing

////////////////////////////////////////////////////////////////////
// process_structarray_statement
//
// This is slightly complicated by having to track number of bits.
//
// Syntax:
//
//    STRUCTARRAY <name>, <"description">, <access_level>,
//          <number_of_bits>
//          <property_list>
//    ENDSTRUCTARRAY
//
// global inputs: messages, variable_name, description, access_level
//                size, structarray_level
//
// global outputs: messages to message.dat, structarray_level
//                 stryct_array_max_bits, structarray_size
//
////////////////////////////////////////////////////////////////////
void process_structarray_statement ()

{

  fprintf (messages, "Structarray %s. with description \n%s\n",
            variable_name, description);
  fprintf (messages, "and Access Level %s\n", access_level);

  structarray_level++;
  structarray_max_bits[structarray_level] = 0;
  structarray_size[structarray_level] = atoi(size);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
} // end process_structarray_statemnt

////////////////////////////////////////////////////////////////////
// structarray_bits
//
// Calculates the maximum number of bits a structarray can use.
// Since the structarray command indicates the number of bits to be
// read to determine how many elements are in the array, the max bits
// is dependent on raising the 2 to the number of bits.
//
// global inputs: structarray_size, structarray_max_bits
//
// global outputs: none
//
// Returns max number of bits for the structarray.
//
////////////////////////////////////////////////////////////////////
int structarray_bits (int level)
{

return ((pow(2,
             structarray_size[level])- 1.0)*structarray_max_bits[level]
             + structarray_size[level]);

} // end structarray_bits

////////////////////////////////////////////////////////////////////
// process_endstructarray_statement
//
// Complicated by calculating number of bits and nested structarrays.
//
// global inputs:  messages, structarray_size, structarray_bit_count,
//                           structarray_level
//
// global outputs: messages to message.dat, structarray_max_bits,
//                           structarray_level
//
////////////////////////////////////////////////////////////////////
void process_endstructarray_statement ()

{

  fprintf (messages, "EndStructarray\n");

  // Have to increment max bits by number of possible array elements.
  if (structarray_level != 0) {

    // We're in a nested structarray.
      structarray_max_bits[structarray_level-1] +=
            structarray_bits(structarray_level);

  }else if (structarray_level == 0) {

      // This is top level structarray.
      print_bit_count (structarray_size[0], structarray_bits(0));

  }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
  structarray_level--;

} // end process_endstructarray_statement

////////////////////////////////////////////////////////////////////
// process_subproperty
//
// This is part of the property command.
//
// global inputs: messages, binary_integer, hex_integer, subproperty
//
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////
void process_subproperty ()

{

  if (register_mask == 0) {
    fprintf (messages, "Subproperty %s\n", subproperty);
  }else{
    fprintf (messages, "Subproperty register mask %s\n",
            register_mask);
  }

} // end process_subproperty

////////////////////////////////////////////////////////////////////
//
//    TDL_VERSION_NUMBER <version number>
//
// Syntax:
//
//    TDL_VERSION_NUMBER <version number>
//
// global inputs: messages,   tdl_version_number
//
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////
void process_tdl_version_number ()

{

  fprintf (messages, "The TDL Version Number is %s.  No bits read.\n",
    tdl_version_number);

} // end tdl version number

////////////////////////////////////////////////////////////////////
// process_ugid
//
// Syntax:
//
//    UGID <“identifier”>, <“description”>
//
// global inputs: messages, ugid
//
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////
void process_ugid ()

{



   fprintf (messages, "The UGID %s is described by No bits read.\n",
                  ugid, description);

} // end process_ugid

////////////////////////////////////////////////////////////////////
// process_validation_keycode
//
// Syntax:
//
//    VALIDATION_KEYCODE <checksum>
//
//
// global inputs: messages, validation_keycode
//
// global outputs: messages to message.dat
//
////////////////////////////////////////////////////////////////////
void process_validation_keycode ()

{

  fprintf (messages, "The validation keycode is %s.  No bits read.\n",
      validation_keycode);

} // end validation_keycode

////////////////////////////////////////////////////////////////////
// print_bit_count (a debug routine)
//
// This not only prints the bit count but also does some of the
// logic to count min and max bits.  In particular, it helps with
// nested selectcases.  It also updates the template bit count array
// so they can be output at the end of the parse.
//
// global inputs: messages, min_bits_read, max_bits_read,
//                case_min_bits, case_max_bits, template_index,
//                min_number_bits, max_number_bits
//
// global outputs: messages to message.dat, min_number_bits,
//                 max_number_bits, case_min_bits, case_max_bits,
//                 min_number_bits, max_number_bits
//
////////////////////////////////////////////////////////////////////
void print_bit_count (int min_bits_read, int max_bits_read)
{

  if (select_level < 0) {
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
    // Not in a selectcase so update bits read.

    min_number_bits[template_index] += min_bits_read;
    max_number_bits[template_index] += max_bits_read;

    fprintf (messages, "Just read between %d and %d bits.\n",
                  min_bits_read, max_bits_read);
    fprintf (messages, "Total bits read between %d and %d\n",
                  min_number_bits[template_index],
                  max_number_bits[template_index]);

  }else{

      // In a selectcase so update case bit counts.

      fprintf (messages, "Potentially read between %d and %d bits.\n",
                  min_bits_read, max_bits_read);
      case_min_bits[select_level][case_number] += min_bits_read;
      case_max_bits[select_level][case_number] += max_bits_read;

  }

} // end print_bit_count

////////////////////////////////////////////////////////////////////
// print_token (a debug routine)
//
// global inputs: tokens, token
//
// global outputs: token printed to tokens.dat
//
////////////////////////////////////////////////////////////////////
void print_token (char* token)
{

  fprintf (tokens, "%s\n", token);
  fflush (tokens);

} // end print_token

////////////////////////////////////////////////////////////////////
// print_statement_num (a debug routine)
//
// global inputs: messages, tokens, statement_count
//
// global outputs: messages to tokens.dat and messages.dat
//
////////////////////////////////////////////////////////////////////
void print_statement_num ()
{
  printf ("Line: %d\n", statement_count);
  fprintf (tokens, "Line : %d\n", statement_count);
  fprintf (messages, "\nLine: %d\n", statement_count);

} // end print_statement_num
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Annex D

(informative)

Template file checksum example

D.1 C++ code

The following C++ code implements calculating and verifying the template checksum as defined in 7.2.5.

```
#include <ctype.h>
#include <stdio.h>

static FILE* file;
static int   ch;
static int   checksumAtLastNewline = 0;

static void getCh()
{
   static int checksum = 0;
   int c = getc (file);

   checksum += c;
   if (c == '\n' || c == '\r') checksumAtLastNewline = checksum;
   ch = toupper (c);
} // getCh

static int calculateChecksum ()
{
   getCh();

   for (;;)
   {
      if (isalpha(ch))
      {
         const char  stopText[] = "VALIDATION_KEYCODE";
         const char* pBegin     = stopText;
         const char* pEnd       = stopText + sizeof(stopText) - 1;
         const char* pCur       = pBegin;

         do
         {
            if (pCur) if (ch == *pCur) pCur++; else pCur = 0;
            getCh();
         } while (isalpha (ch) || isdigit(ch) || ch == '_');

         if (pCur == pEnd) return 0;

      } else if (isdigit(ch) || ch == '+' || ch == '-' || ch == '.')
         do { getCh();}
         while (isdigit(ch) || ch == '+' || ch == '-' || ch == '.' || ch
== 'E');

      else
         switch (ch)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
            {
               default: getCh();

               break; case EOF:
                  return -1;

               break; case '%':
                  do { getCh(); }
                   while (isalpha(ch) || isdigit(ch) || ch == '_' || ch ==
'@');

               break; case '/':
                  getCh();
                  if (ch == '/')
                     do { getCh(); } while (ch != '\n');
                  else return -4;

               break; case '\'': case '\"':
               {
                  const int quote = ch;
                  do { getCh();} while (ch != quote && ch != '\n');

                  if (ch == quote)
                     getCh();
                  else
                     return -3;
               }

         } // switch

      } // for
   } // calculateChecksum

   static int validateFile(const char* filename)
   {
      int calculatedChecksum;
      int res;

      file = fopen (filename, "r");
      if (file == 0) return -6;
      res  = calculateChecksum ();
      if (res != 0) return res;

      calculatedChecksum = checksumAtLastNewline;

      while (isspace (ch)) getCh();

      if (isdigit(ch))
      {
         int foundChecksum = 0;

         do
         {
            foundChecksum = 10 * foundChecksum + ch - '0';
            getCh();
         } while (isdigit(ch));

         if (calculatedChecksum == foundChecksum)
            return 1;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
      else
         return calculatedChecksum;

  } else return -2;
} // validateFile

int main(int argc, char* argv[])
{
   int res = validateFile (argv[1]);

   if (res < 0)
      printf ("Error %d\n", res);
   else if (res == 1)
      printf("The checksum was correct");
   else
      printf("Checksum should be %d\n", res);

   return 0;
} // main
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Annex E

(informative)

Family Codes

The Family Code embedded in the URN of each device signifies a specific device type. Since each device type has different features and commands, it is imperative the bus master knows how to translate this Family Code into the correct commands. Unfortunately, since the Family Code is only an 8-bit value, it is impossible to encode all of the features and commands in it. Instead the bus master must make this association by different means. One method is to hardcode this association in the source code of the master. It can then be updated by rewriting the source code to accommodate new devices. This method is expensive and in some cases impossible, thus relegating some bus masters to only deal with legacy devices.

This annex presents methods to dynamically configure the bus master to correctly communicate with a previously unknown device type.

The first method is used for describing the devices by Family Code in general; the only exception is devices with familycode=0xFD, which are described by a configuration page only. The means is to provide the bus master with a configuration file that describes how to communicate with a particular device type. The bus master could be updated with the latest devices by providing a new configuration file. This annex describes the format of one such configuration file utilizing XML.

The second method is to provide the means for the generic memory family (familycode=0xFD), which is a device described by a configuration page that resides on the device and instructs the bus master on the size and commands for memory access, including special memory, which contains the function bit entities (see 7.4.9.1.2).

New devices might have a configuration page, eliminating the need for updating the XML description.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

E.1 XML configuration file

The XML configuration file format with command sequence notation that can be readily parsed by a bus master to "learn" how to communicate with specific devices. The contents of the configuration file can be updated as new devices are created. While this standard only covers three types of devices (Memory, Switch, and Temperature), the techniques and notation used here could be expanded to cover any type.

E.1.1 Command notation

It is assumed that each bus master must come with the ability to search for devices and read the URN number associated with each device. From this number, the 8-bit Family Code can be extracted. The bus master will then perform operations as defined by this configuration file based on the Family Code. By examining all device operations, a minimum set of commands was derived. The commands are described in Table E.1 along with a suggested notation. Table E.2 describes additional commands that add verification to the command sequences.

© IEEE 2004 – All rights reserved

Table E.1—Core Commands

Notation	Command description
XX	Send the following hex byte value to the bus. If this hex byte is within a CRC block, then calculate the CRC on the result of the operation (see verification commands).
{L, delay}	Delay for L milliseconds.
{M}	Select the device with a reset, Match ROM Command, and device ROM.
{P}	Prime power delivery (strong pullup) to occur after the next byte.
{N}	Restore normal pullup.
{U}	Issued a 12-volt pulse (used in EPROM programming).
{Ax}	Supply a memory address where x is a number 0,1,… representing the LSbyte to MSbyte. For example, {A0} {A1} would specify a 16-bit bit address with the LSbyte first followed by the MSbyte.
{Dx}	Data to write to a memory device where the x is a number 0,1,… representing the LSbyte to MSbyte of the data. For example, {D0} {D1} {D2} is 3 bytes of data to write. Note the master processing these commands would place the actual data into the command flow.
{R}	Read memory bytes to end of memory. All values read are valid data however now verification is performed.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

Table E.2—Verification commands

Notation	Command description
{dx}	Data to read. This data can be for verification of data written to a memory device or result data such as a temperature conversion. Note it is in same format as the {Dx} command where x is a number indicating the byte number, with {d0} being the LSbyte.
{T}	Success is reading toggling bits such as 0xAA or 0x55.
{00}	Success is reading all 0's such as 0x00.
{FF}	Success is reading all 1's such as 0xFF.
{CRC16,start,seed}	Start CRC16 calculation by first setting the CRC16 to the provided "seed" represented in hex notation. All following command bytes will be included in the calculation until the "check" command is found.
{CRC16,check,value}	Check the CRC16 calculated value to make sure it equals the provided hex "value." If it is not then this is a failure. The CRC16 calculation can be stopped after the check.
{CRC8,start,seed}	Start CRC8 calculation by first setting the CRC8 to the provided "seed" represented in hex notation. All following command bytes will be included in the calculation until the "check" command is found.
{CRC16,check,value}	Check the CRC8 calculated value to make sure it equals the provided hex "value." If it is not, then this is a failure. The CRC8 calculation can be stopped after the check.

© IEEE 2004 – All rights reserved

E.1.2 Device types

The general types of devices covered by this standard are *Memory*, *Switch*, and *Temperature*.

The Memory device has some kind of data storage memory area. It may be write-once, but must support multiple reads. It is often arranged in pages and is usually written a page at a time. A Memory device may have multiple banks of memory with different attributes.

The Switch device can control a latch. The latch may connect the output to ground (lowside) or to the communication channel (highside). Some Switch devices can also sense voltage. A Switch device can have multiple channels.

The Temperature device returns a temperature value in Celsius. The result is a signed value representing temperature units.

Each device type contains one or more standard operations. For example, each Temperature device has a "read" operation. Table E.3 shows the standard operations and attributes of each device type.

Table E.3—Device operations and attributes by type

Device type	Operations	Attributes
Memory	Read Write	ReadWrite/ReadOnly/WriteOnce Starting physical address Number of pages Page length in bytes
Switch	Read Latch Enable Latch Disable Latch Read Level (optional)	HighSide/LowSide
Temperature	Read	Min Temperature Max Temperature Step (unit of Celsius returned from Read)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

The "Setup" operation is also included in any of the device type descriptions. A "Setup" is a command sequence that readies the device for operation. For any of the "Read" operations there are also two attributes "AndMask" and "Polarity." The "AndMask" is a hex value that is bitwise anded with the result data described in the command sequence with {d0}. The "Polarity" indicates that the operation is "TRUE" if it matches the resulting value from the "AndMask." For example, when reading the latch state (Read Latch) of a dual channel switch with ROM (Family Code `0x12`) channel A, the AndMask="0x01" and Polarity="0x00." So the value read from the command sequence is bitwise anded with 0x01 and if the result is 0, the latch is ON.

E.1.3 Configuration format

The XML syntax was selected for the example configuration file format. Since XML is so "eXtensible" it was easy to incorporate the device types, operations, attributes and the actual command sequences into a human readable format. The overall "tag" for grouping these descriptions was <DeviceDescriptions>. Within this group are individual device descriptions with a specified Family Code attribute, for example: <Device FamilyCode="0x23">. Each device group can contain <MemoryBank>, <SwitchChannel>, or <TemperatureChannel> groups that correspond to the device types already described. Note that some

© IEEE 2004 – All rights reserved

devices may have more than one channel and group. For example, the dual channel switch with ROM (Family Code `0x12`) has both memory and switch groups since it incorporates both of these features. See the example XML file in E.1.4 describing six different devices. Two of these devices are memory, two are switches, and two are temperature devices.

E.1.4 Examples

Example XML configuration file for 6 devices

```
<?xml version="1.0" encoding="UTF-8"?>

<DeviceDescriptions>
   <Device FamilyCode="0x23">
      <Description>
         4kbit EEPROM
      </Description>

      <MemoryBank attributes="ReadWrite">
         <Description>
            Main Memory
         </Description>

         <StartAddress> 0x0000 </StartAddress>
         <Pages> 16 </Pages>
         <PageLength> 32 </PageLength>

         <Write>
            <WriteScratchPad>
                {M} {CRC16,start,0}
                0F {A0} {A1}
                {D0}  {D1}  {D2}  {D3}  {D4}  {D5}  {D6}  {D7}
                {D8}  {D9}  {D10} {D11} {D12} {D13} {D14} {D15}
                {D16} {D17} {D18} {D19} {D20} {D21} {D22} {D23}
                {D24} {D25} {D26} {D27} {D28} {D29} {D30} {D31}
                FF FF {CRC16,check,0xB001}
            </WriteScratchPad>
            <CopyScratchPad>
                {M} 55 {A0} {A1} {P} 1F {L,10} {N} {T}
            </CopyScratchPad>
         </Write>

         <Read>
            <ReadMemory>
                {M} F0 {A0} {A1} {R}
            </ReadMemory>
         </Read>

      </MemoryBank>
   </Device>

   <Device FamilyCode="0x14">
      <Description>
         32-byte EEPROM with locking register
      </Description>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
<MemoryBank attributes="ReadWrite">
   <Description>
      Main Memory
   </Description>

   <StartAddress> 0x0000 </StartAddress>
   <Pages> 1 </Pages>
   <PageLength> 32 </PageLength>

   <Write>
      <WriteScratchPad>
          {M} 0F {A0}
          {D0}  {D1}  {D2}  {D3}  {D4}  {D5}  {D6}  {D7}
          {D8}  {D9}  {D10} {D11} {D12} {D13} {D14} {D15}
          {D16} {D17} {D18} {D19} {D20} {D21} {D22} {D23}
          {D24} {D25} {D26} {D27} {D28} {D29} {D30} {D31}
      </WriteScratchPad>
      <ReadScratchPad>
          {M} AA {A0}
          {d0}  {d1}  {d2}  {d3}  {d4}  {d5}  {d6}  {d7}
          {d8}  {d9}  {d10} {d11} {d12} {d13} {d14} {d15}
          {d16} {d17} {d18} {d19} {d20} {d21} {d22} {d23}
          {d24} {d25} {d26} {d27} {d28} {d29} {d30} {d31}
      </ReadScratchPad>
      <CopyScratchPad>
          {M} 55 {P} A5 {L,20} {N}
      </CopyScratchPad>
   </Write>

   <Read>
      <ReadMemory>
          {M} F0 {A0} {R}
      </ReadMemory>
   </Read>

</MemoryBank>

<MemoryBank attributes="WriteOnce">
   <Description>
      Application Register
   </Description>

   <StartAddress> 0x0000 </StartAddress>
   <Pages> 1 </Pages>
   <PageLength> 8 </PageLength>

   <Write>
      <WriteAppReg>
          {M} 99 {A0}
          {D0}  {D1}  {D2}  {D3}  {D4}  {D5}  {D6}  {D7}
      </WriteAppReg>
      <ReadAppReg>
          {M} C3 {A0}
          {d0}  {d1}  {d2}  {d3}  {d4}  {d5}  {d6}  {d7}
      </ReadAppReg>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
            <CopyAndLock>
                {M} 5A {P} A5 {L,20} {N}
            </CopyAndLock>
        </Write>

        <Read>
            <ReadAppReg>
                {M} C3 {A0} {R}
            </ReadAppReg>
        </Read>
    </MemoryBank>
</Device>

<Device FamilyCode="0x12">
    <Description>
        DS2406, dual channel switch with 1kbit EPROM
    </Description>

    <MemoryBank attributes="WriteOnce">
        <Description>
            Main Memory
        </Description>

        <StartAddress> 0x0000 </StartAddress>
        <Pages> 4 </Pages>
        <PageLength> 32 </PageLength>

        <Write>
            <WriteScratchPad>
                {M} {CRC16,start,0}
                0F {A0} {A1} {D0}
                FF FF {CRC16,check,0xB001}
            </WriteScratchPad>
            <Program>
                {U}
            </Program>
            <ReadVerify>
                {d0}
            </ReadVerify>
        </Write>

        <Read>
            <ReadMemory>
                {M} F0 {A0} {A1} {R}
            </ReadMemory>
        </Read>

    </MemoryBank>

    <SwitchChannel attributes="LowSide">
        <Description>
            PIO-A
        </Description>

        <ReadLatch AndMask="0x01" Polarity="0x00">
            {M} {CRC16,start,0}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
            F5 55 FF {d0}
            FF FF {CRC16,check,0xB001}
         </ReadLatch>

         <ReadLevel AndMask="0x04" Polarity="0x04">
            {M} {CRC16,start,0}
            F5 55 FF {d0}
            FF FF {CRC16,check,0xB001}
         </ReadLevel>

         <EnableLatch>
            {M} {CRC16,start,0}
            F5 05 FF 00
            FF FF {CRC16,check,0xB001}
         </EnableLatch>

         <DisableLatch>
            {M} {CRC16,start,0}
            F5 05 FF FF
            FF FF {CRC16,check,0xB001}
         </DisableLatch>

      </SwitchChannel>

      <SwitchChannel attributes="LowSide">
         <Description>
            PIO-B
         </Description>

         <ReadLatch AndMask="0x02" Polarity="0x00">
            {M} {CRC16,start,0}
            F5 55 FF {d0}
            FF FF {CRC16,check,0xB001}
         </ReadLatch>

         <ReadLevel AndMask="0x08" Polarity="0x08">
            {M} {CRC16,start,0}
            F5 55 FF {d0}
            FF FF {CRC16,check,0xB001}
         </ReadLevel>

         <EnableLatch>
            {M} {CRC16,start,0}
            F5 09 FF 00
            FF FF {CRC16,check,0xB001}
         </EnableLatch>

         <DisableLatch>
            {M} {CRC16,start,0}
            F5 09 FF FF
            FF FF {CRC16,check,0xB001}
         </DisableLatch>

      </SwitchChannel>
   </Device>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
<Device FamilyCode="0x1F">
   <Description>
      Coupler
   </Description>

   <SwitchChannel attributes="HighSide">
      <Description>
         Main
      </Description>

      <ReadLatch AndMask="0x01" Polarity="0x00">
         {M} 5A 18 {d0}
      </ReadLatch>

      <ReadLevel AndMask="0x02" Polarity="0x02">
         {M} 5A 18 {d0}
      </ReadLevel>


      <ReadActivity AndMask="0x10" Polarity="0x10">
         {M} 5A 18 {d0}
      </ReadActivity>

      <EnableLatch>
         {M} A5 FF
      </EnableLatch>

      <DisableLatch>
         {M} 66 FF
      </DisableLatch>
   </SwitchChannel>

   <SwitchChannel attributes="HighSide">
      <Description>
         Auxilary
      </Description>

      <ReadLatch AndMask="0x04" Polarity="0x00">
         {M} 5A 18 {d0}
      </ReadLatch>

      <ReadLevel AndMask="0x08" Polarity="0x08">
         {M} 5A 18 {d0}
      </ReadLevel>

      <EnableLatch>
         {M} 33 FF FF FF
      </EnableLatch>

      <DisableLatch>
         {M} 66 FF
      </DisableLatch>
   </SwitchChannel>
</Device>

<Device FamilyCode="0x10">
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
      <Description>
         fixed resolution temperature sensor
      </Description>

      <TemperatureChannel min="-55" max="125" step="0.5">
         <Read>
            <Recall>
               {M} B8
            </Recall>
            <Conversion>
               {M} {P} 44 {L,750} {N} {FF}
            </Conversion>
            <Result>
               {M} BE {CRC8,start,0} {d0} {d1}
               FF FF FF FF FF FF FF {CRC8,check,0x00}
            </Result>
         </Read>
      </TemperatureChannel>
   </Device>



   <Device FamilyCode="0x28">
      <Description>
         high-resolution temperature sensor
      </Description>

      <TemperatureChannel min="-55" max="125" step="0.0625">
         <Setup>
            <WriteScatchPad>
               {M} 00 00 7F
            </WriteScatchPad>

            <CopyScatchPad>
               {M} {P} 48 {L,10} {N}
            </CopyScatchPad>
         </Setup>

         <Read>
            <Recall>
               {M} B8
            </Recall>
            <Conversion>
               {M} {P} 44 {L,750} {N} {FF}
            </Conversion>
            <Result>
               {M} BE {CRC8,start,0} {d0} {d1}
               FF FF FF FF FF FF FF {CRC8,check,0x00}
            </Result>
         </Read>
      </TemperatureChannel>
   </Device></DeviceDescriptions>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

E.2 Memory configuration page

The following general memory description describes an idealized memory device with a configuration page that provides all of the necessary information to utilize the remaining memory space.

All memory devices support the Read Memory command (F0 hex), and with the exception of the Family Code `0x14`, it requires 2 address bytes. Consequently the Read Memory command will be used to retrieve the configuration page information at a fixed address of FF7F hex. The memory location will have a length byte, 26 bytes of data followed by an inverted CRC16 for validation.

E.2.1 Operations

Table E.4—List of operations

Operation	EEPROM	EPROM	Description
Read Memory	X	X	Read memory with device generated CRC
Read Page with CRC	x	x	Read a page of memory with device generated CRC
Write Scratchpad	X		Write the scratchpad in preparation of writing to memory
Read Scratchpad	X		Read the scratchpad to verify the write was correct
Copy Scratchpad	X		Copy the scratchpad to the final memory location
Write Memory		X	Write a byte to memory
Read special page with CRC		x	Read a page of special memory with device generated CRC
Write special byte		x	Write a byte to the special memory
Write protect page	x	x	Write protect a page
Set page for write-once	x		Set an EEPROM page to be write-once like (pseudo EPROM)
Calculate Free Pages		x	Calculate the number of free pages in an EPROM device by looking at the map of used pages

X = supported by all devices of this type

x = supported by some devices of this type

<blank> = not supported by devices of this type

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

E.2.2 Configuration page format

Table E.5—Configuration page format

Byte offset	Name	Content	
0	Length	Length of data in the configuration page (fixed at 26)	
1	General_Flags	Bit 0	Memory type (1 EEPROM, 0 EPROM)
		Bit 1	Scratchpad erased on read-memory (1 YES, 0 NO)
		Bit 2	Device has read page with CRC16 (1 YES, 0 NO)
		Bit 3	Device has write-once mode like pseudo EPROM (1 YES, 0 NO) (EEPROM only)
		Bit 4	Device has map of used pages (1 YES, 0 NO)
		Bit 5	Not used 0
		Bit 6	Not used 0
		Bit 7	Not used 0
2	WriteProt_Flags	Bit 0	Individual page write-protect (1 YES, 0 NO)
		Bit 1	Global device write-protect (1 YES, 0 NO)
		Bit 2	Write protect register is organized with one page per bit (1 YES, 0 NO). If no then is one page per byte.
		Bit 3	Not used 0
		Bit 4	Not used 0
		Bit 5	Not used 0
		Bit 6	Not used 0
		Bit 7	Not used 0
3	CRC_Flags	Bit 0	Write scratchpad has CRC16 (1 YES, 0 NO)
		Bit 1	Read scratchpad has CRC16 (1 YES, 0 NO)
		Bit 2	Read special memory command has CRC16 (1 YES, 0 NO)
		Bit 3	Not used 0
		Bit 4	Not used 0
		Bit 5	Not used 0
		Bit 6	Not used 0
		Bit 7	Not used 0
4	Scratchpad_Length	Length of scratchpad in bytes (EEPROM only)	
5	Page_Length	Length of normal memory page in bytes	
6	Pages	Number of pages (2 bytes)	
8	Special_Pages	Number of special function pages	

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Table E.5—Configuration page format

Byte offset	Name	Content
9	Special_Page_Length	Length of special memory page in bytes
10	ReadScratch_CMD	Read scratchpad command
11	Write_CMD	Write command (scratchpad for EEPROM)
12	CopyScratch_CMD	Copy scratchpad command
13	ReadPageCRC_CMD	Read page of memory with CRC16 command
14	ReadSpecial_CMD	Read special memory page command
15	Write_Special_CMD	Write special memory command
16	WriteProt_Addr	Address of write-protect registers in special memory (2 bytes)
18	WriteProtDev_Addr	Address of write-protect entire device register in special memory (2 bytes)
20	WriteOnce_Addr	Address to write-once mode (pseudo EPROM) flag in special memory (2 bytes)
22	UsedPgs_Addr	Address in special memory for map of used pages (2 bytes)
24	UsedPgs_Offset	Bit offset of the map of used pages
25	WriteProt_Value	Value written to special memory register to write-protect a page
26	WriteOnce_Value	Value written to special memory register to make a page write-once like pseudo EPROM
27	CRC16	Bitwise inverted CRC16 of bytes 0 to 24, LSByte first (2 bytes)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

E.2.3 Operations detail

E.2.3.1 Read Memory

- Reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write ReadMemory command (F0 hex)
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Read data

E.2.3.2 Read Page with CRC

- If General_Flags.Bit2 = 1
- Reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write ReadPageCRC_CMD
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Read Page_Length bytes (unless address is not at page beginning)
- Read bitwise inverted CRC16

© IEEE 2004 – All rights reserved

E.2.3.3 Write Scratchpad

- If General_Flags.Bit0 = 1
- Reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write Write_CMD
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Write data bytes
- If CRCFlags.Bit0 = 1 AND At end of page
- Read bitwise inverted CRC16

E.2.3.4 Read Scratchpad

- If General_Flags.Bit0 = 1
- Reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write ReadScratch_CMD
- Read first address byte TA1, LSByte
- Read second address byte TA2, MSByte
- Read offset and status flags ES
- Read data bytes
- If CRCFlags.Bit1 = 1 AND At end of page
- Read bitwise inverted CRC16

E.2.3.5 Copy Scratchpad

- If General_Flags.Bit0 = 1
- Reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write CopyScratch_CMD
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Write offset and status flags ES
- Strong pullup applied to the device for a minimum of 10 ms
- Read confirmation byte (should be AA or 55)

E.2.3.6 Write Memory

- If General_Flags.Bit0 = 0
- Reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write Write_CMD
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Write data byte to write
- Read bitwise inverted CRC16 of command, address, and data (first pass) or address and data (second pass)

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

— Apply 480us 12V programming pulse on the device.
— Read confirmation data byte (should OR of old data and current data bytes)
— If next address to write is sequential, then can send the next data byte …

E.2.3.7 Read Special Page with CRC

— If General_Flags.Bit2 = 1
— Reset and presence
— ROM level command sequence (read/search/match/overdrive match/overdrive skip)
— Write ReadSpecial_CMD
— Read first address byte TA1, LSByte
— Read second address byte TA2, MSByte
— Read Special_Page_Length bytes (unless address is not at page beginning)
— Read bitwise inverted CRC16

E.2.3.8 Write Special byte

— If General_Flags.Bit0 = 0
— Reset and presence
— ROM level command sequence (read/search/match/overdrive match/overdrive skip)
— Write Write_Special_CMD
— Write first address byte TA1, LSByte
— Write second address byte TA2, MSByte
— Write data byte to write
— Read bitwise inverted CRC16 of command, address, and data (first pass) or address and data (second pass)
— Apply 480us 12V programming pulse on the device
— Read confirmation data byte (should OR of old data and current data bytes)
— If next address to write is sequential then can send the next data byte …

E.2.3.9 Write Protect Page

— If WriteProt_Flags.Bit0 = 1
— If WriteProt_Flags.Bit2 = 1
— Address = WriteProt_Addr + Page / 8
— Data = WriteProt_Value Rotate left Remainder (Page / 8)
— Else
— Address = WriteProt_Addr + Page
— Data = WriteProt_Value
— Write Special Byte with Address and Data

E.2.3.10 Set Page for Write-Once

— If General_Flags.Bit3 = 1
— Address = WriteOnce_Addr
— Data = WriteOnce_Value
— Write Special Byte with Address and Data

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

E.2.3.11 Mark Page Used

- If General_Flags.Bit4 = 1
- Address = UsedPgs_Addr + (Page + UsedPgs_Offset)/ 8
- Data = BitInverse(1 Rotate left Remainder ((Page + UsedPgs_Offset)/ 8)
- Write special byte at Address and Data

E.2.3.12 Calculate Free Pages

- If General_Flags.Bit4 = 1
- Address = UsedPgs_Addr
- Read special page with CRC starting at Address until (Special_Pages / 8) number of bytes read
- Count the number of 1's in the bytes read, this is the number of free pages

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Annex F

(informative)

IEEE 1451.4 XML device description schema

The IEEE 1451.4 XML device description schema provides a template for manufacturers and users of IEEE Std 1451.4-2004 to add support for new devices to their systems. The schema defines devices that support memory, switching, and temperature reading. The schema is based on the document described in "1-Wire Master Device Configuration" [B1] with the addition of CRC map location and memory map ordering. An example of a device description file can be found in "1-Wire Master Device Configuration" [B1].

F.1 IEEE 1451.4 XML device description schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- The IEEE 1451.4 XML device description schema provides a template
for manufacturers and users of the IEEE14514 to add support for new
devices to their systems. The schema defines devices that support memory,
switching and temperature reading.  -->

<xs:schema                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
      <xs:element name="Conversion" type="xs:string"/>
      <xs:element name="CopyAndLock" type="xs:string"/>
      <xs:element name="CopyScatchPad" type="xs:string"/>
      <xs:element name="CopyScratchPad" type="xs:string"/>
      <xs:element name="Description" type="xs:string"/>
      <xs:complexType name="IEEE1451_Dot4DeviceType">
            <xs:sequence>
                  <xs:element ref="Description"/>
                  <xs:element  name="MemoryBank"  type="MemoryBankType"
minOccurs="0" maxOccurs="unbounded"/>
                  <xs:element                        name="SwitchChannel"
type="SwitchChannelType" minOccurs="0" maxOccurs="unbounded"/>
                  <xs:element                 name="TemperatureChannel"
type="TemperatureChannelType" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="FamilyCode" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                              <xs:enumeration value="0x10"/>
                              <xs:enumeration value="0x12"/>
                              <xs:enumeration value="0x14"/>
                              <xs:enumeration value="0x1F"/>
                              <xs:enumeration value="0x23"/>
                              <xs:enumeration value="0x28"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
      </xs:complexType>
      <xs:element name="DeviceDescriptions">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element                          name="Device"
type="IEEE1451_Dot4DeviceType" maxOccurs="unbounded"/>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="DisableLatch" type="xs:string"/>
      <xs:element name="EnableLatch" type="xs:string"/>
      <xs:complexType name="MemoryBankType">
            <xs:sequence>
                  <xs:element ref="Description"/>
                  <xs:element ref="StartAddress"/>
                  <xs:element ref="Pages"/>
                  <xs:element ref="PageLength"/>
                  <xs:element name="Write" type="WriteType"/>
                  <xs:element name="Read" type="ReadType"/>
                  <xs:element name="CRCInformation" minOccurs="0">
                        <xs:complexType>
                              <xs:sequence maxOccurs="unbounded">
                                    <xs:element
name="CRCStartBitPageLocation" type="xs:unsignedLong"/>
                                    <xs:element      name="CRCBitLength"
type="xs:unsignedLong"/>
                              </xs:sequence>
                        </xs:complexType>
                  </xs:element>
            </xs:sequence>
            <xs:attribute name="attributes" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                              <xs:enumeration value="ReadWrite"/>
                              <xs:enumeration value="WriteOnce"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
      </xs:complexType>
      <xs:element name="PageLength" type="xs:unsignedLong"/>
      <xs:element name="Pages" type="xs:unsignedLong"/>
      <xs:element name="Program" type="xs:string"/>
      <xs:complexType name="ReadType">
            <xs:sequence>
                  <xs:element ref="ReadMemory" minOccurs="0"/>
                  <xs:element ref="ReadAppReg" minOccurs="0"/>
                  <xs:element ref="Recall" minOccurs="0"/>
                  <xs:element ref="Conversion" minOccurs="0"/>
                  <xs:element ref="Result" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="ReadActivityType">
            <xs:simpleContent>
                  <xs:extension base="xs:string">
                        <xs:attribute   name="AndMask"   type="xs:string"
use="required"/>
                        <xs:attribute  name="Polarity"  type="xs:string"
use="required"/>
                  </xs:extension>
            </xs:simpleContent>
      </xs:complexType>
      <xs:element name="ReadAppReg" type="xs:string"/>
      <xs:complexType name="ReadLatchType">
            <xs:simpleContent>
                  <xs:extension base="xs:string">
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
                        <xs:attribute name="AndMask" use="required">
                              <xs:simpleType>
                                    <xs:restriction base="xs:NMTOKEN">
                                          <xs:enumeration value="0x01"/>
                                          <xs:enumeration value="0x02"/>
                                          <xs:enumeration value="0x04"/>
                                    </xs:restriction>
                              </xs:simpleType>
                        </xs:attribute>
                        <xs:attribute  name="Polarity"  type="xs:decimal"
use="required"/>
                  </xs:extension>
            </xs:simpleContent>
      </xs:complexType>
      <xs:complexType name="ReadLevelType">
            <xs:simpleContent>
                  <xs:extension base="xs:string">
                        <xs:attribute name="AndMask" use="required">
                              <xs:simpleType>
                                    <xs:restriction base="xs:NMTOKEN">
                                          <xs:enumeration value="0x02"/>
                                          <xs:enumeration value="0x04"/>
                                          <xs:enumeration value="0x08"/>
                                    </xs:restriction>
                              </xs:simpleType>
                        </xs:attribute>
                        <xs:attribute name="Polarity" use="required">
                              <xs:simpleType>
                                    <xs:restriction base="xs:NMTOKEN">
                                          <xs:enumeration value="0x02"/>
                                          <xs:enumeration value="0x04"/>
                                          <xs:enumeration value="0x08"/>
                                    </xs:restriction>
                              </xs:simpleType>
                        </xs:attribute>
                  </xs:extension>
            </xs:simpleContent>
      </xs:complexType>
      <xs:element name="ReadMemory" type="xs:string"/>
      <xs:element name="ReadScratchPad" type="xs:string"/>
      <xs:element name="ReadVerify" type="xs:string"/>
      <xs:element name="Recall" type="xs:string"/>
      <xs:element name="Result" type="xs:string"/>
      <xs:complexType name="SetupType">
            <xs:sequence>
                  <xs:element ref="WriteScatchPad"/>
                  <xs:element ref="CopyScatchPad"/>
            </xs:sequence>
      </xs:complexType>
      <xs:element name="StartAddress" type="xs:string"/>
      <xs:complexType name="SwitchChannelType">
            <xs:sequence>
                  <xs:element ref="Description"/>
                  <xs:element name="ReadLatch" type="ReadLatchType"/>
                  <xs:element name="ReadLevel" type="ReadLevelType"/>
                  <xs:element                    name="ReadActivity"
type="ReadActivityType" minOccurs="0"/>
                  <xs:element ref="EnableLatch"/>
                  <xs:element ref="DisableLatch"/>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

```
            </xs:sequence>
            <xs:attribute name="attributes" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                              <xs:enumeration value="HighSide"/>
                              <xs:enumeration value="LowSide"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
      </xs:complexType>
      <xs:complexType name="TemperatureChannelType">
            <xs:sequence>
                  <xs:element       name="Setup"      type="SetupType"
minOccurs="0"/>
                  <xs:element name="Read" type="ReadType"/>
            </xs:sequence>
            <xs:attribute name="min" type="xs:byte" use="required"/>
            <xs:attribute name="max" type="xs:byte" use="required"/>
            <xs:attribute name="step" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                              <xs:enumeration value="0.0625"/>
                              <xs:enumeration value="0.5"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
      </xs:complexType>
      <xs:complexType name="WriteType">
            <xs:sequence>
                  <xs:element ref="WriteScratchPad" minOccurs="0"/>
                  <xs:element ref="ReadScratchPad" minOccurs="0"/>
                  <xs:element ref="CopyScratchPad" minOccurs="0"/>
                  <xs:element ref="WriteAppReg" minOccurs="0"/>
                  <xs:element ref="ReadAppReg" minOccurs="0"/>
                  <xs:element ref="CopyAndLock" minOccurs="0"/>
                  <xs:element ref="Program" minOccurs="0"/>
                  <xs:element ref="ReadVerify" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:element name="WriteAppReg" type="xs:string"/>
      <xs:element name="WriteScatchPad" type="xs:string"/>
      <xs:element name="WriteScratchPad" type="xs:string"/>
</xs:schema>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved

Annex G

(informative)

Communication with nodes in sensors on remote locations

G.1 Background

2-conductor bus devices are used in remote places where the distance between the device and the host may exceed the bus specifications.

In such cases it is useful to be able to have some kind of communication equipment in between the sensor and the data processing computer that allows data from nodes to be efficiently and transparently transferred, for instance. over LAN or WAN networks, in a uniform and consistent way.

The goal with this standard is to suggest how this problem can be easily solved in a relatively simplified manner by inserting a transport layer in the driver software, which operates with a uniform frame buffer format. This extra layer allows different parts of the protocol software to be placed at different physical locations, and in this way, extends the versatility of the concept.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

G.2 Scenario

Devices built into remote sensors are, via a 2-conductor bus and a number of individual instruments, connected to a host application. The instruments are connected together via different communication lines and are using different communication protocols. The instruments will in this respect act as communication repeaters, which transfer data transparently between the Host application and devices on the 2-conductor bus.

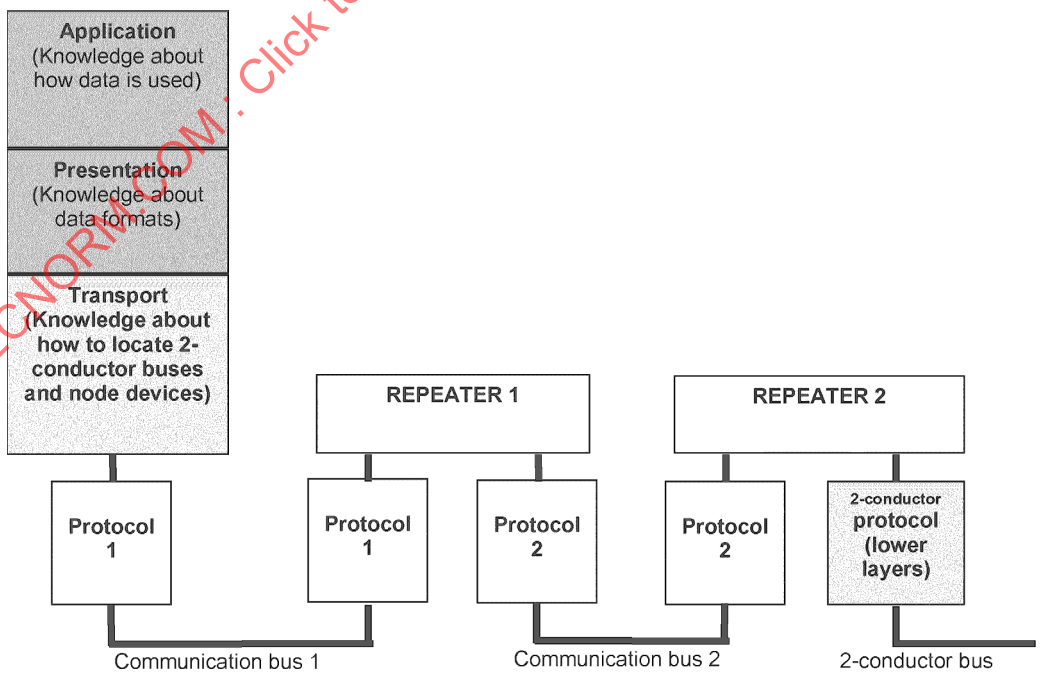


Figure G.1—Repeater in a multiple-protocol scenario

© IEEE 2004 – All rights reserved

G.2.1 Primary goals

The software in the instruments (the repeaters) should be stable for the lifetime of the instruments (> 10 years) even if new (and yet unknown) devices are connected the 2-conductor bus.

The communication speed should be optimized. This implies that the number of communication transactions on the other communication busses (bus 1 and bus 2 in Figure G.1) should be minimized, as these busses may have a much lower bandwidth than the 2-conductor bus itself, and/or may also be used for other communication tasks not related to the 2-conductor bus communication.

G.2.2 Derived goals

All knowledge about specific device types should be isolated to the host program. The repeaters should not contain any device specific knowledge.

Communication sessions should be based on whole buffers (instead of individual bytes and bits) in order to minimize the communication overhead on the intervening busses (bus 1 and 2 in Figure G.1).

A few basic and device transparent 2-conductor bus transactions for the repeaters should be defined, together with the corresponding buffer formats. These few device transparent transactions should be sufficient for communication with all types of 2-conductor bus devices.

The minimum buffer size, which a repeater shall be able to handle, shall be well-defined. (It is assumed that the repeaters may have a very limited buffering capability).

If communication with a device requires a larger buffer, it should be possible to split a transaction over several intervening buffers transferred between the host and the repeater that have the 2-conductor bus connection. The intervening communication protocols will typically pack the 2-conductor bus buffer frame in “envelopes” using their own format (for instance, add some header and tail bytes). This is transparent to the 2-conductor bus communication and is not a part of this standard.

G.2.3 Accepted limitations

It is not required that the 2-conductor bus interface in the repeater handle EPROM programming voltages, higher speed “overdrive” communication, or strong pullup power delivery but its use is to be defined by this specification.

It is assumed that all communication initiatives through the repeaters are initiated from the host application.

G.2.4 Transparent 2-conductor bus buffer transactions

The transparent buffer transaction on the 2-conductor bus takes advantage of the fact that transmit and receive can be done at the same time on a bit-to-bit basis. [A one (1) shall be transmitted by the repeater when receiving bit frames from a 2-conductor device].

After a transaction, the buffer in the repeater will contain any information read from the device. The buffer in the repeater with the resulting 2-conductor bus transaction can then be transmitted back to the host, if needed. All buffer communication initiatives are taken by the host.

IECNORM.COM : Click to view the full PDF of ISO/IEC/IEEE 21451-4:2010

© IEEE 2004 – All rights reserved