

# 1451.4™

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

**IEEE Instrumentation and Measurement Society**

Sponsored by the  
Technical Committee on Sensor Technology TC-9



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

15 December 2004

Print: SH95225

PDF: 599-243



**Documents**

Document Sharing Hub

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

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.

# 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.

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.

## 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>.

## 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.

## Participants

At the time this standard was completed, the Working Group had the following membership:

**Torben Licht**, *Chair, Editor*

**David Potter**, *Vice Chair*

**Paul Hufnagel**, *Secretary*

Jørgen Bække  
Mike Dillon  
Garritt Foote  
Fernando Gen-Kuong  
Ed Herceg

Charles H. Jones  
Kang Lee  
Carlos Lopez-Reyna  
John Mark  
Jochen Schneider

Bill Schuh  
Eugene Song  
Charles Summey  
Darold Wobschall  
Jay Zemel

The following persons contributed to the earlier versions of this standard:

Robert Broadbent  
Thurston Brooks  
Mark Buckner  
Russell Dominique

Leon Fialkowski  
Brett Gidge  
David Goetzinger  
Mike Moore

Joseph Que  
Mark Schiefer  
David Smiczek  
Stephen Webb

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Jørgen Bække  
Chris Bagge  
Jacob Ben Ary  
L. Wayne Catlin  
Keith Chow  
Amir El-Sheikh  
Fernando GenKuong  
Patrick Gonia  
Erich Gunther  
Ernst Hanique  
Howard Howland  
Paul Hufnagel

Tony Jeffree  
Robert Johnson  
Charles Jones  
George Kalacherry  
James Kemerling  
Kang Lee  
Yeou-Song Lee  
Torben Licht  
Gregory Luri  
John J. Mark  
Gary Michel  
Rajesh Moorkath

Charles Ngethe  
David Potter  
Gary S. Robinson  
Mark Schiefer  
Michael J. Seaman  
Matthew Smith  
Geoffrey Thompson  
Steven Tilden  
Mark Tilllinghast  
Stephen C. Webb  
Darold Wobschall  
Stan Woods

When the IEEE-SA Standards Board approved this standard on 25 March 2004, it had the following membership:

**Don Wright, Chair**  
**Steve M. Mills, Vice Chair**  
**Judith Gorman, Secretary**

Chuck Adams  
H. Stephen Berger  
Mark D. Bowman  
Joseph A. Bruder  
Bob Davis  
Roberto de Boisson  
Julian Forster\*  
Arnold M. Greenspan

Mark S. Halpin  
Raymond Hapeman  
Richard J. Holleman  
Richard H. Hulett  
Lowell G. Johnson  
Joseph L. Koepfinger\*  
Hermann Koch  
Thomas J. McGean  
Daleep C. Mohla

Paul Nikolich  
T. W. Olsen  
Ronald C. Petersen  
Gary S. Robinson  
Frank Stone  
Malcolm V. Thaden  
Doug Topping  
Joe D. Watson

\*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*  
Richard DeBlasio, *DOE Representative*  
Alan Cookson, *NIST Representative*

Noelle D. Humenick  
*IEEE Standards Project Editor*

# Contents

|     |                                                           |    |
|-----|-----------------------------------------------------------|----|
| 1.  | Overview.....                                             | 1  |
| 1.1 | Scope.....                                                | 2  |
| 1.2 | Purpose.....                                              | 2  |
| 1.3 | Conformance, shall, should, may, and can .....            | 2  |
| 2.  | References.....                                           | 2  |
| 3.  | Definitions and abbreviations .....                       | 3  |
| 3.1 | Terms .....                                               | 3  |
| 3.2 | Abbreviations.....                                        | 6  |
| 4.  | IEEE 1451.4 Transducer.....                               | 7  |
| 4.1 | Foundation .....                                          | 7  |
| 4.2 | IEEE 1451.4 Transducer configuration .....                | 8  |
| 4.3 | Compliance with this standard, IEEE Std 1451.4-2004 ..... | 9  |
| 5.  | Transducer Electronic Data Sheet.....                     | 10 |
| 5.1 | Basic TEDS.....                                           | 10 |
| 5.2 | IEEE, User, and Manufacturer TEDS.....                    | 11 |
| 5.3 | Data format and templates .....                           | 11 |
| 5.4 | Nodes, addresses, Family Codes, URN, and CRC .....        | 13 |
| 5.5 | Data transmission.....                                    | 14 |
| 5.6 | Structure of the TEDS data system.....                    | 14 |
| 6.  | Templates.....                                            | 15 |
| 6.1 | Overview .....                                            | 15 |
| 6.2 | Discovery of the transducer(s) present .....              | 16 |
| 6.3 | Identification of transducers and their nodes .....       | 16 |
| 6.4 | Assembling the Transducer TEDS .....                      | 19 |
| 6.5 | Parsing the Transducer TEDS.....                          | 20 |
| 7.  | Template Description Language (TDL) .....                 | 22 |
| 7.1 | Overview .....                                            | 22 |
| 7.2 | Identification commands.....                              | 23 |
| 7.3 | Control commands .....                                    | 27 |
| 7.4 | Property commands (%) .....                               | 30 |
| 8.  | Mixed Mode Transducer Interface (MMI) specification.....  | 56 |
| 8.1 | Introduction.....                                         | 56 |
| 8.2 | Analog Mode .....                                         | 59 |
| 8.3 | Digital Mode .....                                        | 60 |
| 8.4 | Line definitions .....                                    | 60 |
| 8.5 | MMI digital Data Transmission Protocol .....              | 61 |

|     |                                                                                                  |     |
|-----|--------------------------------------------------------------------------------------------------|-----|
| 9.  | Transducer Block specification .....                                                             | 67  |
| 9.1 | Overview .....                                                                                   | 68  |
| 9.2 | TBOM specification .....                                                                         | 72  |
| 9.3 | Common Object Interface (COI) specification.....                                                 | 89  |
| 9.4 | TEDS Service .....                                                                               | 102 |
| 9.5 | IEEE 1451.4 Transducer Block general interface .....                                             | 104 |
|     | Annex A (normative) IEEE standard templates .....                                                | 126 |
|     | Annex B (normative) Property definitions .....                                                   | 147 |
|     | Annex C (informative) TDL formal grammar .....                                                   | 286 |
|     | Annex D (informative) Template file checksum example.....                                        | 321 |
|     | Annex E (informative) Family Codes.....                                                          | 324 |
|     | Annex F (informative) IEEE 1451.4 XML device description schema.....                             | 339 |
|     | Annex G (informative) Communication with nodes in sensors on remote locations .....              | 343 |
|     | Annex H (normative) Procedures for adding new IEEE templates and TDL items and to get URNs ..... | 377 |
|     | Annex I (informative) IEEE P1451.4, version 0.9, and beta information .....                      | 378 |
|     | Annex J (normative) IEEE 1451.4 Manufacturer IDs and model numbers.....                          | 380 |
|     | Annex K (normative) IEEE 1451.4 TBOM schema .....                                                | 383 |
|     | Annex L (normative) IEEE 1451.4 Transducer Block IEEE 1451.1 adapter definition .....            | 409 |
|     | Annex M (informative) Bibliography .....                                                         | 430 |

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

## 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.<sup>1</sup> Annex M is the bibliography.

<sup>1</sup>Information on references can be found in Clause 2.

## 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.<sup>2</sup>

IEEE Std 754™-1985 (Reaff 1990), IEEE Standard for Binary Floating-Point Arithmetic.<sup>3,4</sup>

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

<sup>2</sup>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/>).

<sup>3</sup>The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

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

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 10646-1:2002, Information technology—Universal Multiple-Octet Coded Character Set (UCS)—Part 1: Architecture and Basic Multilingual Plane.<sup>5</sup>

Le Système international d’unités (SI), The International System of Units (SI), 7th Edition (1998), with Supplement 2000.<sup>6</sup>

### 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).

<sup>5</sup>ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembe, 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/>).

<sup>6</sup>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/>).

**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.

**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 than 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 than 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.

### 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                                                  |

|      |                                  |
|------|----------------------------------|
| 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.

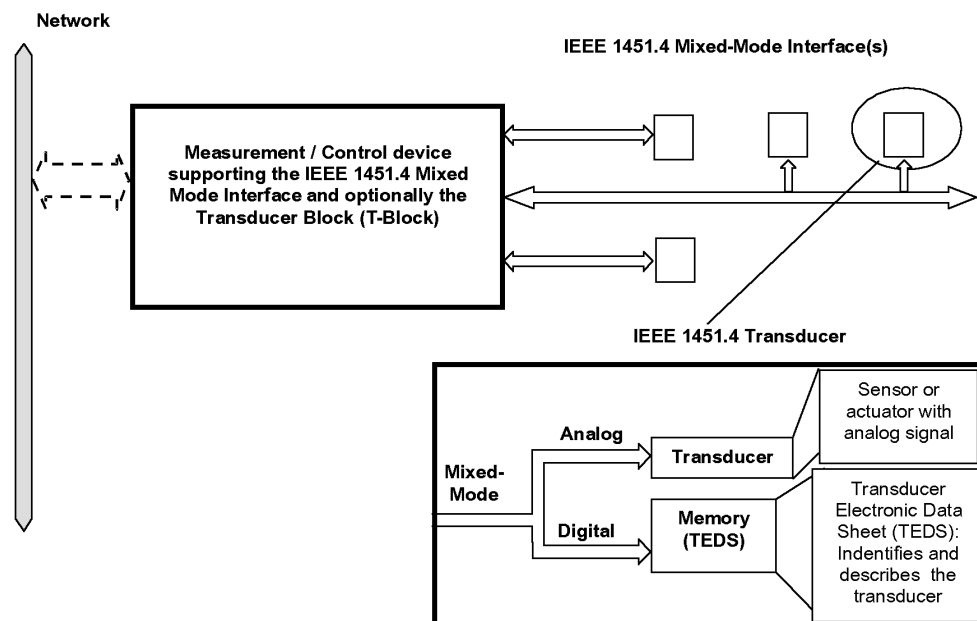


**Figure 1—Context for the IEEE 1451.4 Transducer and Interface**

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

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

#### 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:

- 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.<sup>7</sup>

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

<sup>7</sup>Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

**Table 2—Basic TEDS content**

| <b>Manufacturer ID</b>             | <b>Model number</b>  | <b>Version letter</b>                | <b>Version number</b> | <b>Serial number</b>    |
|------------------------------------|----------------------|--------------------------------------|-----------------------|-------------------------|
| 14 bits<br>(17–16381) <sup>a</sup> | 15 bits<br>(0–32767) | 5 bits Char<br>(A–Z, data type Chr5) | 6 bits<br>(0–63)      | 24 bits<br>(0–16777215) |

<sup>a</sup>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]<sup>8</sup>, 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.

<sup>8</sup>The numbers in brackets correspond to those of the bibliography in Annex M.

Table 3—Memory map, IEEE 1451.4 Basic TEDS

| IEEE 1451.4 Basic TEDS                |                       |   |                        |                      |                                        |   |                      |   |     |
|---------------------------------------|-----------------------|---|------------------------|----------------------|----------------------------------------|---|----------------------|---|-----|
| msb                                   | 7                     | 6 | 5                      | 4                    | 3                                      | 2 | 1                    | 0 | lsb |
| 07                                    | Manufacturer ID (7:0) |   |                        |                      |                                        |   |                      |   | 00  |
| 15                                    | Model number (1:0)    |   | Manufacturer ID (13:8) |                      |                                        |   |                      |   | 08  |
| 23                                    | Model number (9:2)    |   |                        |                      |                                        |   |                      |   | 16  |
| 31                                    | Version letter (2:0)  |   |                        | Model number (14:10) |                                        |   |                      |   | 24  |
| 39                                    | Version number (5:0)  |   |                        |                      |                                        |   | Version letter (4:3) |   | 32  |
| 47                                    | Serial number (7:0)   |   |                        |                      |                                        |   |                      |   | 40  |
| 51                                    | Serial number (15:8)  |   |                        |                      |                                        |   |                      |   | 48  |
| 63                                    | Serial number (23:16) |   |                        |                      |                                        |   |                      |   | 52  |
| msb =<br>most sig-<br>nificant<br>bit |                       |   |                        |                      | lsb =<br>least<br>signifi-<br>cant bit |   |                      |   |     |

## 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.

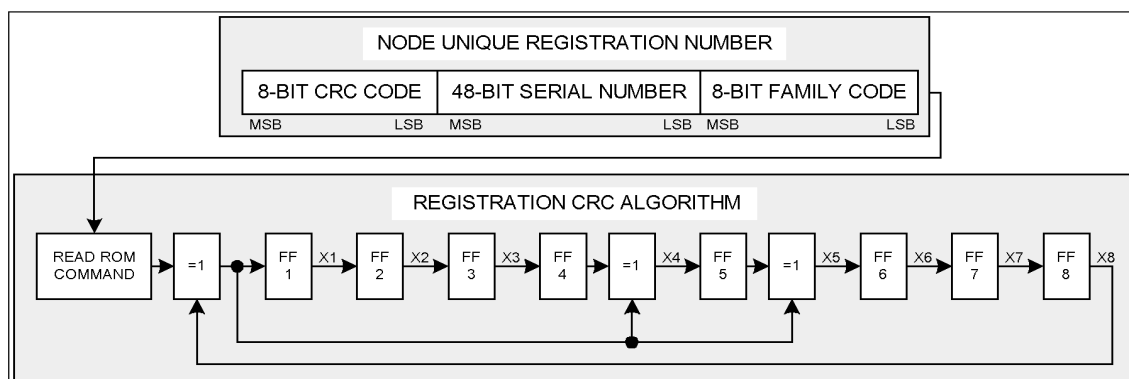


Figure 2—URN and CRC generation

## 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.

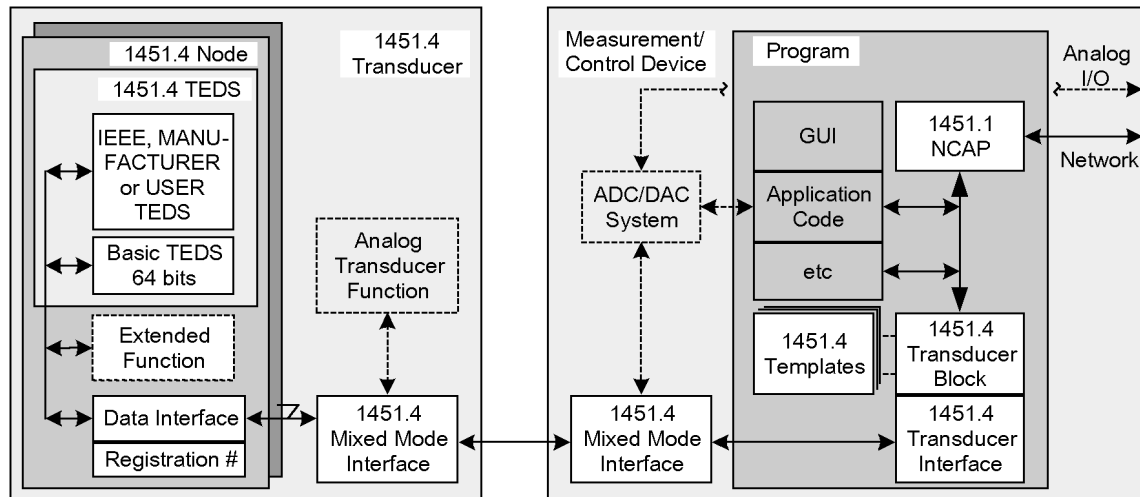


Figure 3—IEEE 1451.4 TEDS data system block diagram

## 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.

## 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.

- 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).

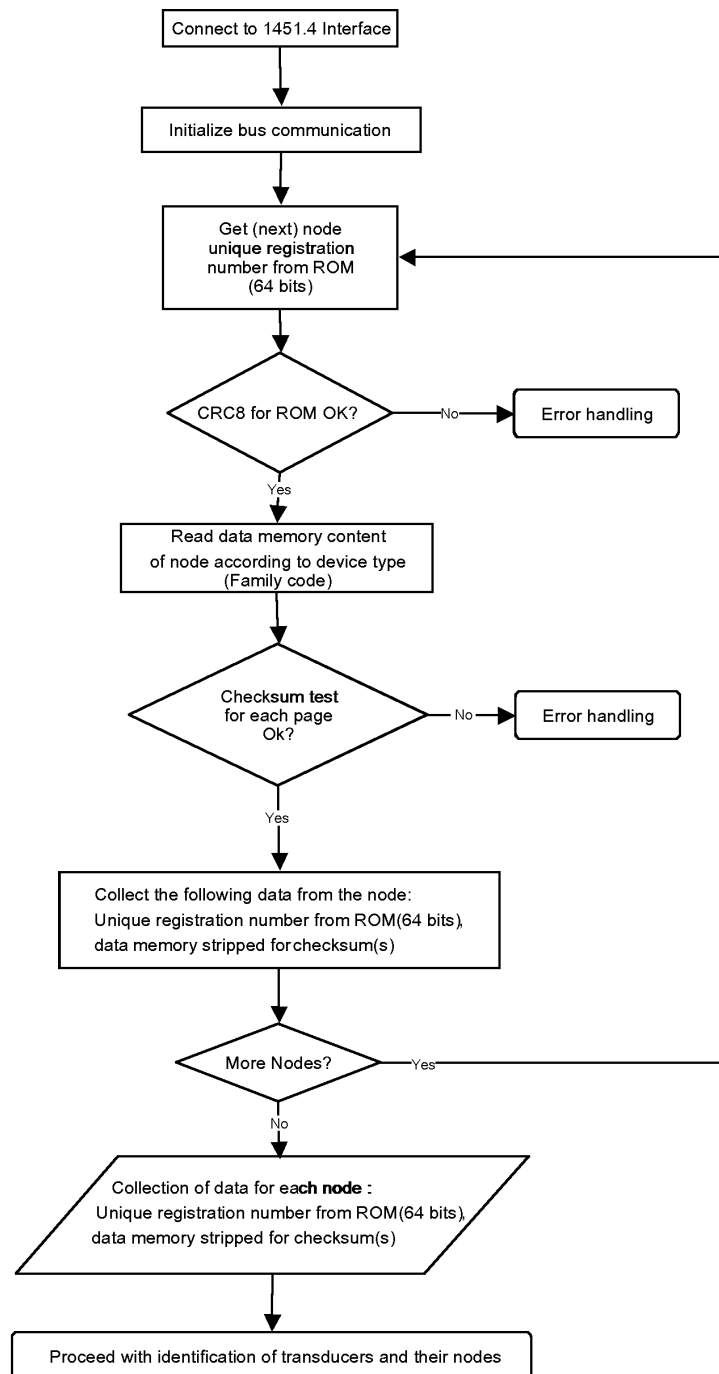


**Figure 4—Flowchart showing the discovery of the transducer(s) present**

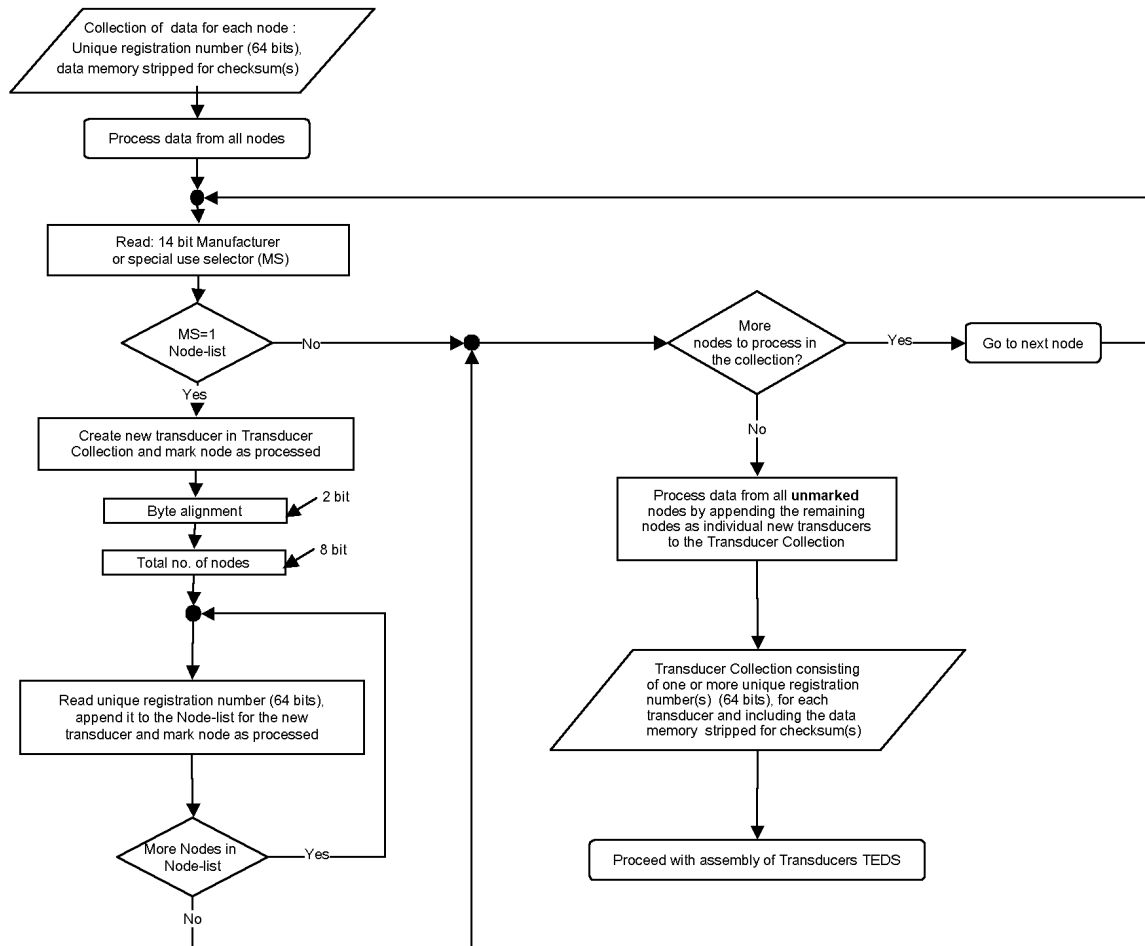


**Figure 5—Flowchart showing the identification of the transducer(s) and their nodes**

### 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

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 |

## 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.

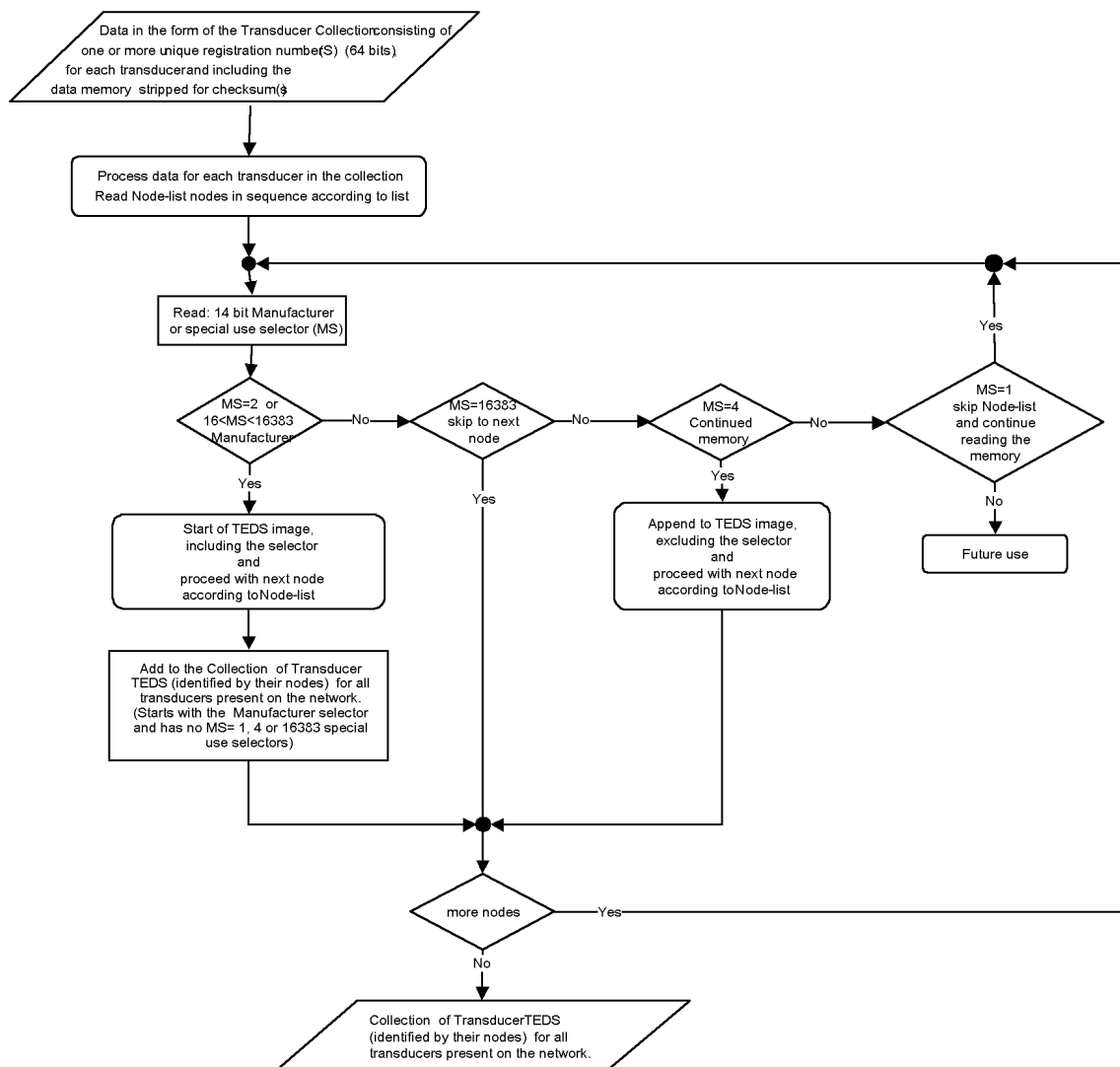


**Figure 6—Flowchart showing the assembly of the Transducer TEDS**

- 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.

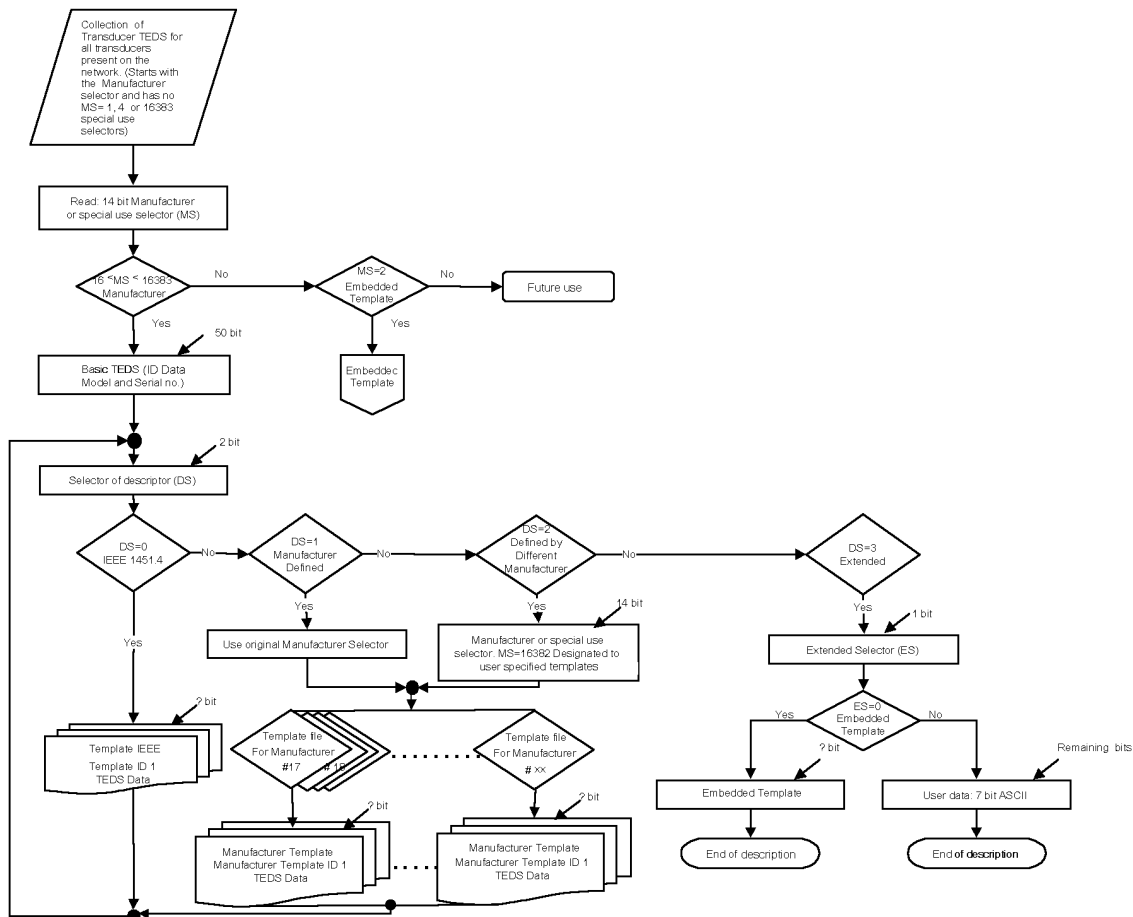


**Figure 7—Flowchart showing parsing of the Transducer TEDS**

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

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.

### 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.

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.

- 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:

$$\text{UDID} = \{I<\text{tid}>\{\text{---}<\text{sc}>\}_0^\infty \mid M<\text{mid}>:<\text{tid}>\{\text{---}<\text{sc}>\}_0^\infty E \mid U\}_1^\infty$$

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 occur between 0 to  $\infty$  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.

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>*

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
```

```

        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.

*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

$$\text{<word\_size>} - (\text{Current\_Position} \bmod \text{<word\_size>})$$

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,

$$8 - (238 \bmod 8) = 2$$

## 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

%<property\_tag>,<"description">,<access\_level>,<data\_type>,<format>,<physical\_unit> = <value>

<physical\_unit> = <value>

or

%<property\_tag><[subproperty]>,<"description">,<access\_level>,<data\_type>,<format>,<physical\_unit>

<format>,<physical\_unit>

or

%<property\_tag><[subproperty]>,<"description">,<access\_level>,<data\_type>,<format>,<physical\_unit> = <value>

<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.

#### 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.                       |

#### 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.

**Table 7—Data types**

| <b>&lt;dtype&gt;</b>       | <b>Description</b>                                                                                             |
|----------------------------|----------------------------------------------------------------------------------------------------------------|
| 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 10646-1:2002 and <i>The Unicode Standard, Version 2.0</i> [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.                     |
| <b>&lt;float_type&gt;</b>  | <b>Description</b>                                                                                             |
| 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.                                                |
| <b>&lt;enum_type&gt;</b>   | <b>Description</b>                                                                                             |
| (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.

#### 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
```

**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 10646-1:2002 (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.

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:

$$\text{property name} = \text{<start\_value>} + [\text{<tolerance>} \times \text{<teds\_value>}]$$

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  $[0, (2^{\text{<number\_of\_bits>} - 2})]$  to the real interval  $[\text{<start\_value>}, \text{maximum value}]$ . Where:

$$\text{maximum value} = \text{<start\_value>} + [\text{<tolerance>} \times (2^{\text{<number\_of\_bits>} - 2})]$$

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:

$$\text{TempCoef} = -0.5 + [0.002 \times 8] = -0.484$$

$$\text{maximum value} = -0.5 + [0.002 \times (2^9 - 2)] = 0.52$$

*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  $2 \times 10^7$  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.

$$\text{property name} = \text{<start\_value>} * [1 + 2 * \text{<tolerance>}] ^ \text{<teds\_value>}$$

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  $[0, (2^{\text{<number\_of\_bits>} - 2})]$  to the real interval  $[\text{<start\_value>}, \text{maximum value}]$ , where the maximum value is realized when the <teds\_value> is maximal.

$$\text{Maximum value} = \text{<start\_value>} * \{[1 + (2 * \text{<tolerance>})] ^ [(2^{\text{<number\_of\_bits>} - 2}) - 2]\}$$

*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:

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 101000000000000000000000

which would be used to calculate:

$$\text{MaxPhysVal} = -1 \times 2 \times (129 - 127) \times 1.625 = -6.5$$

#### 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.

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**

| <b>Numeric formatting symbols</b> | <b>Description</b>                                                                                                                                                                                             |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .                                 | (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.                                                                                                                                 |
| <b>String format</b>              | <b>Description</b>                                                                                                                                                                                             |
| S                                 | Display as a standard string.                                                                                                                                                                                  |

**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              |

#### 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

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 | $\mu$  |
| $10^{18}$ | 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      | 999,7 $\mu$                | 999,66 $\mu$                  |
| 9,99808E-4                | 1,00011E-3 | 25340                       | 9,99957E-4      | 1000,0 $\mu$               | 999,96 $\mu$                  |

A similar template command with lower resolution: