IEEE Std 1377-1997
also known as
ANSI C12.19-1997
and
Industry Canada MC-XXX

IEEE Standard for Utility Industry End Device Data Tables

Sponsor

IEEE Standards Coordinating Committee 31
on
Automatic Meter Reading and Energy Management

Approved 26 June 1997

IEEE Standards Board

Abstract: Functionally related utility application data elements, grouped into a single data structure for transport are described. Data may be utilized peer-to-peer or upstream to readers or billing systems by being carried by one lower layered protocol to another stack of lower layered protocol. The data structure does not change from end device to the user of the data.
Keywords: decade of tables, end device, function-limiting control (FLC) table, time-of-use (TOU) table

---

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

ISBN 1-55937-932-4

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. 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.

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.

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 all 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.

Comments on standards and requests for interpretations should be addressed to:

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

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.

Introduction

(This introduction is not part of IEEE Std 1377-1997, IEEE Standard for Utility Industry End Device Data Tables.)

The beginning of the tables was sparked by the desire of a number of people who needed a single communications protocol for their newly emerging electronics metering devices in the 1970s. One of those people was Bill Earthman of Duke Power Company, who brought a young engineer from the distribution field organization to help update the Duke Power Company mechanical metering to the new and unwieldly growing state of electronic metering. A single communications protocol was the new staff engineer's first assignment in 1980.

In 1983, a meeting was sponsored for protocols discussion. After a hotly debated day's session between memory map versus command driven techniques, engineers held further discussions and compromised on an "indirect addressing" method of manipulating and describing data that small meter processors could manage. The table concept was born.

In 1984, EEI created an adhoc committee that vigorously created a high-level language for metering, known as the "MCL" Meter Command Language. This language required too much meter microprocessor power and was rejected. Although rejected, this work provided valuable input to the continued effort with tables.

The few interested manufacturers and utilities started meeting under the auspices of EIA standard's CEBus meetings. With the help of Dick Stilwell, Kevin McDonald, Larry Schmidt, Deryl Rhoades, Michael Anderson, Ted York, Bill Beverly, John Lauletta, Brian Seal, Skeeter Wall, and the Duke Power Company engineer, the tables work became recognized as the Utility Context in the CEBus standard. The Utility Context became more than EIA wanted to be responsible for; the Tables work was orphaned and found a new home with the AMRA/IEEE SCC31 standards body navigated by Bill Rush, who urged the work forward.

The Southeastern Electric Exchange adopted the the Tables as a guideline under the Metering Committee headed by Ed Malemezian. Jean Joly of Hyro Quebec, Canada visited a number of U.S. Electric Utilities seeking a solution to the problem of proliferation of metering protocols. He found the tables work at Duke Power Company. The Canadian and U.S. collaboration started, and under the guidance of Vuong Nguyen, prodigious work was created within the Canadian "Task Force on Data Communication Protocol for Electronic Metering Devices." That task force is now under the auspices of Industry Canada. With the work of Kah-Fae Chan, Avygdor Moise, and Michel Veillette, the tables became rich with new capabilities.

ANSI C12 Subcommittee 17 became active with Electric Utilities metering communications protocol work. Under the guidance of Lynnda Ell, the table work accelerated within the ANSI C12 standards body. The C12.19 work and other works have since been carefully sheparded by Wes Ray.

All three standards bodies, ANSI C12, Industry Canada, and IEEE SCC31, decided to collectively meet to finish the tables work and establish one North American standard for metering data description. There were 12 meetings of (usually) four days. These meetings became known as "Tablefests". Bill Gibson chaired these meetings beginning with a biblical verse reminding everyone the purpose of the meeting. Terry Penn and George Stephens edited the evergrowing working document. Input from Europe was cheerfully brought through John Newbury. Thousands of hours of work by the best talent to be had produced the rich and unique fuctionality of the "Tables."

The following members participated in the working group:

Richard D. Tucker, *Chair*

Ted York, *Secretary*

Charlie Adkins
GlennAllen
Tariq Amjed
Michael Anderson
Jim Andrus
Tom Asp
Paul Aubin
Ron Bane
Claire Becker-Castle
Ludo Bertsch
Bill Beverly
Pierre Bezzina
Manthi Bilinsky
William Blair
Donald Block
Tom Bowling
William Bray
Jim Brennan
Andrew Brock
Tom Broe
William Buckley
Eddie Buckner
Eric Buffkin
Dennis Burman
Martin Burns
Randy Butler
Jeff Buxton
Ken Caird
Tom Campbell
Larry Carmichael
Murray Carney
Tony Centorino
Kah-Fae Chan
Kathy Chang
Gerard Chevalier
Ron Claridge
Frances Cleveland
Bernice Cochran
Mike Crowley
Robert Cruickshank
Karen Daderko
Mike Dalton
Chuck Davis
Joe Delaney
Emil Drobber
Bill Earthman
Ellen Edge
Don Eggleston
Mark Eldeckin
Lynnda Ell
Paul Estes
Ron Farquharson
Donald Fisher
Natan Galperin
Stuart Garland
Bob Garry
Ron Genova
Dan Gerhart
David Gestler
Clark Getty
Bill Gibson
Jeff Gill
Jerry Gillean
Frank Glaser
Greg Gomez
Peter Gorman
Dave Gorton
Bruce Gray
John Grubbs
Richard Guenther
Michael Hajny
Paul Hargaden
Joseph Harley
Leon Harris
Clifford Hubbard
Ivo Hug
Joe Hughes
Paul Hunt
Pavel Jamnik
John Jansen
Bill Jenrette
Jean Joly
Clair Jones
Lee Karsten
John Korz
Lawrence Kotewa
Bill Kram
Donald Kuilmann
Jeff Kuiper
Michael Kunst
Becky LaBrosse
Trung Lang
John Lauletta
John Lennox
Mark Lockareff
Joe Lola
Ed Malemezian
Brian Markwalter
John Marsh
Kit Maughn
Kevin McDonald
Jay McLellan
Al Messano
Laurel Miller
Jon Milliken
James Mining
Avygdor Moise
Andrew Morgan
Jim Nakama
Robert Neu
August Nevolo
John Newbury
Vuong Nguyen
Jeff O'Kuiper
Dan Nordell
Diane Palmer
Lauren Pananen
Chuck Partridge
Duane Paulson
Jack Pazdon
Terry Penn
Dan Pinney
Andy Porten
Daniel Pouliot
Dale Praught
David Propp
Harold Rabbie
Denny Radford
Wesley Ray
Richard Reed
Derl Rhoades
Jeff Richardson
Steve Robinson
Benoit Robitallie
Art Robleto
Aubin Roy
Rick Rundus
Bill Rush
Phil Russel
Frank Safertel
Steve Sanislo
Chris Schafer
Ben Schall
George Schimmel
Larry Schmidt
Richard Schmidt
Paul Schroeder
Jerome Schull
Lee Schwanke
Felix Sciulli
David Scott
Howard Scott
Brian Seal
Dick Seubert
Ken Shadek
Tushar Shah
Brian Simpson
Joe Simpson
Eric Smith
Kendle Smith
Ivo Steklac
George Stephens
Richard Stetler
Dick Stilwell
Donald Strobel
Dan Sugarman
Mike Surratt
Victor Tamosaitis
Paul Taylor
Sumin Tchen
Fred Toepfer
Paula Tomkinson
Tim Vahlsrom
Jim VanderMey
Michael Vecchi
Michel Viellette
Kenneth Wachs
Stuart Wagonfeld
H.A. (Sketter) Wall
Steve Ward
Avery Washington
Trevor White
Fred Whitmore
Kurt Weichert
Kent Williams
David Wiseman
Christopher Yasko
Gary Ziegler

The following persons were on the balloting committee:

L. A. Carmichael
Donald Fisher
Stephen Hadden
Richard Holbert
Richard Holtz
August J. Nevolo
Daniel E. Nordell
Rick Rundus
William F. Rush
George Schimmel
Jerome W. Schull
Howard Scott
Tushar K. Shah
Joseph Simpson
Michael Surratt
Richard D. Tucker
Raymond S. Turgel
Ted York

When the IEEE Standards Board approved this standard on 26 June 1997, it had the following membership:

Donald C. Loughry, *Chair* **Richard J. Holleman,** *Vice Chair*
Andrew G. Salem, *Secretary*

Clyde R. Camp
Stephen L. Diamond
Harold E. Epstein
Donald C. Fleckenstein
Jay Forster*
Thomas F. Garrity
Donald N. Heirman
Jim Isaak
Ben C. Johnson
Lowell Johnson
Robert Kennelly
E. G. "Al" Kiener
Joseph L. Koepfinger*
Stephen R. Lambert
Lawrence V. McCall
L. Bruce McClung
Marco W. Migliaro
Louis-François Pau
Gerald H. Peterson
John W. Pope
Jose R. Ramos
Ronald H. Reimer
Ingo Rüsch
John S. Ryan
Chee Kiow Tan
Howard L. Wolfman

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
Alan H. Cookson

Lisa S. Young
IEEE Standards Project Editor

Contents

IEEE Standard for Utility Industry End Device Data Tables

1. Scope

This standard defines a table structure for utility application data to be passed between an end device and a computer. It does not define device design criteria, nor specify the language or protocol used to transport that data. The purpose of the tables is to define structures for transporting data to and from end devices.

2. References

ANSI C12.10-1987: Electromechanical Watthour Meters

ECMA-94 (June 1986), 8-Bit Single Byte Coded Graphic Character Sets—Latin Alphabet No. 1 to No. 4.[1]

IEEE Std 100-1996, The IEEE Standard Dictionary of Electrical and Electronics Terms. [2]

IEEE Std 754-1985 (Reaff 1990), IEEE Standard for Binary Floating-Point Arithmetic.

ISO/IEC 646: 1991, Information technology—ISO 7-bit coded character set for information interchange.[3]

ISO 7498-1: 1994, Information technology—Open Systems Interconnection—Basic reference model: The Basic Model.

ISO 8859-1: 1987, Information processing—8-bit single byte coded graphic character sets—Part 1: Latin Alphabet No. 1.

---

[1]ECMA publications are available from the European Computer Manufacturers Association, 114 rue du Rhone, CH-1204, Geneva, Switzerland/Suisse.

[2]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

[3]ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

3. Definitions

For the purposes of this standard, the following definitions apply:

3.1 act: Abbreviation for ACTUAL, indicating the programmed functional capabilities of an end device.

3.2 address: Those inputs whose states select a particular cell or group of cells.

3.3 average demand: The consumption (e.g., energy, volume) recorded during the integration period divided by the integration time period.

3.4 block average demand: An average value occurringoccuring over a demand period specified by the end device (e.g., watthours/hours). The value may be saved by the end device for maximum or minimum registration.

3.5 clock: A device that generates periodic signals used for synchronization. A device that measures and indicates time. A register whose content changes at regular intervals in such a way as to measure time.

3.6 cumulative demand: An indicating demand meter in which the accumulated total of maximum demands during the preceding periods is indicated during the period after the meter has been reset and before it is reset again. NOTE—The maximum demand for any one period is equal or proportional to the difference between the accumulated readings before and after reset.

3.7 customer : The purchaser and/or user of a product or service supplied by a service provider or utility.

3.8 data encryption: The changing of the form of a data stream such that only the intended recipient can read or alter the information and detect unauthorized messages.

3.9 decade: A functional grouping of tables by application into groups of ten. The tables are numbered “X0” through “X9,” with “X” representing the decade number.

3.10 demand: The rate of consumption, e.g., power, volume/hour.

3.11 dim: Abbreviation for DIMENSION, indicating the maximum functional capability designed into an end device. *See:* **function limiting control table.**

3.12 end device: The closest device to the sensor or control point within a metering application communication system that is compliant with the utility industry end device data tables.

3.13 function-limiting control (FLC) table: The first table in each decade specifies the limits designed into an end device with respect to variables used within the decade.

3.14 load profile: The recording, storage, and analysis of consumption data over a period of time for a particular installation.

3.15 maximum demand: The highest demand measured over a selected period of time, e.g., one month.

3.16 meter: A device that measures and records the consumption or usage of the product/service.

3.17 minimum demand: The lowest demand measured over a selected period of time, such as one month.

3.18 octet: A sequence of eight bits. For application within this standard, see 5.2.

3.19 peak demand: *See:* **maximum demand.**

3.20 present average demand: An average value occurring during a current demand interval or subinterval (e.g., watts or voltamperes).

3.21 season: A calendar-specified period used for activation of rate schedules.

3.22 sliding window (rolling-interval) demand: The block demand calculated over an integration period that includes sub-intervals of previous demand calculations.

3.23 table: Functionally related utility application data elements, grouped together into a single data structure for transport.

3.24 tariff: A published list of rate schedules and terms and conditions.

3.25 time-of-use (TOU) metering: Metering equipment that separately records metered or measured quantities according to a time schedule.

3.26 utility: A provider of electricity, gas, water, telecommunications, or related services to a community.

4. General

4.1 Standard tables

Standard tables are those whose structures are specified by this standard. These should be used for both end device programming and reading. The tables provide control tables as well as data tables for a wide variety of functions to be implemented in addition to those to be defined. This standard provides for a total of 2048 standard tables, although not all are defined.

The standard tables are grouped together in decades, with each decade covering a general area of functionality. The first table of each decade, beginning with the tens decade, is referred to as a function limiting control (FLC) table. The purpose of the FLC table is to specify the designed DIMENSION (DIM) limits for variables used within the decade for the end device.

If present, the table immediately following an FLC table is called the FLC+1 table. The FLC+1 table defines the ACTUAL (ACT) limits used for variables in the end device current configuration.

If the FLC+1 table is not present, the ACTUAL (ACT) limits are identical to the designed DIMENSION (DIM) limits specified in the FLC table.

If the FLC and FLC+1 tables are not used in a decade, then the limits are defined by a default set of values. The default sets for a particular decade are defined within this standard in Annex C**.** Possible combinations of tables are defined in Figure 1.

	FLC table	FLC+1 table	FLC+N table	Where ACT values are found
A	Not used	Not used	Not used	None—Decade not used
B	Not used	Not used	Used	Default set, per standard
C	Not used	Used	Not used	Not practical
D	Not used	Used	Used	Decade FLC+1 table
E	Used	Not used	Not used	Not practical
F	Used	Not used	Used	Decade FLC table
G	Used	Used	Not used	Not practical
H	Used	Used	Used	Decade FLC+1 table

Figure 1—Possible combinations of FLC, FLC+1, and decade tables

4.2 Manufacturer tables

Manufacturer tables are those structures specified by individual end device vendors. They should be used to allow introduction of new innovations or to provide customer requested data structures. They also provide a path for new functions to evolve into a standard table. This standard provides access for a total of 2048 manufacturer tables.

5. Syntax

5.1 Descriptive syntax

Describing data definitions is usually accomplished within the confines of a given language and the grammar rules of that language. Since the data definitions embodied within this standard are meant to be independent of a specific language and to be capable of being implemented within the confines of any language, a method for describing the data definitions has been adopted.

Symbol	Meaning
< >	A string contained inside angle brackets is called a non-terminal. That is, while it may be viewed as a single unit it can and shall be redefined as consisting of one or more simpler elements.
:: =	This symbol is read as "is defined as." The non-terminal that occurs on the left-hand side (LHS) of this symbol consists of the elements (non-terminals, terminals, or a combination of the two) found on the right-hand side (RHS). A line containing an LHS, :: =, and an RHS is known as a production rule.
|	The vertical bar is an **OR** symbol. The **OR** symbol always occurs on the right hand side of a production where the left hand side can be defined in more than one way. The **OR** bar separates valid alternative right hand sides.
[ ]	A symbol enclosed in square brackets is optional. The production is valid whether or not it is included.
*	The superscript asterisk is known as the Kleene star. A symbol followed by the Kleene star may occur zero or more times without violating the grammar.
+	The superscript plus sign is known as the Kleene cross. A symbol followed by the Kleene cross shall occur one or more times.
{ }	Text enclosed within curly braces is a comment provided for clarity and has no impact on the syntax or data definition.

The data structure elements are indicated by **BOLD** type. Reserved words, characters, identifiers, and other symbols required for the language are also indicated by **BOLD** type.

5.2 Flow of information

The flow of information is a stream of objects of multiple “n” octet length, when “n” is greater than one**.** Objects are transmitted in order of their appearance in the table structure.

For purposes of transmission, the elementary data type is an octet or byte, which is eight bits in length. The bits are numbered zero to seven. Zero is the least significant bit. Figure 2 illustrates this bit numbering and ordering. The order of transmission of the bits in an octet is defined elsewhere in the appropriate OSI layer, typically the Data Link Layer (layer 2). An octet may also be referenced as a byte.

MSBit							LSBit
7	6	5	4	3	2	1	0

Figure 2—Octet bit ordering

For some data types that are more than one octet in length, it is possible to transmit them with either most or least significant byte first. The order of transmission is dictated by **DATA_ORDER**, found in the **GEN_CONFIG_TBL** (Table 00). Each data type shall indicate a specific order of transmission or refer to order of transmission in Basic Data Type Definitions (see 5.4.1). Figure 3 illustrates a multibyte structure.

MSByte							LSByte
Octet	Octet	Octet	Octet	Octet	Octet	Octet	Octet
7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0

Figure 3—Multibyte ordering

5.3 Identifiers

An identifier is a name identifying a table, a packed record, a bit field, or a member of a packed record or of a bit field.

An identifier is made of letters, and/or numbers, and/or the character ‘_’. Some suffixes have been reserved for use with specific identifiers. An identifier ending with **_TBL** identifies a table, **_RCD** identifies a packed record and **_BFLD** identifies a bit field.

The identifiers used to identify the members of a packed record or of a bit field are subdivided according to the type of data they define. These identifiers are <int-identifier>, <bool-identifier>, <set-identifier> and <array-identifier>.

<number> :: = <digit>$^{+}$

<digit> :: = 0 to 9

<lchar> :: = <digit>l<chararacter>

<character> :: = A to Z only, upper case.

<ID> :: = <lchar>$^{+}$[_<lchar>$^{+}$]*

<tbl-identifier> :: = <ID>**_TBL**

<rcd-identifier> :: = <ID>**_RCD**

<bfld-identifier> :: = <ID>**_BFLD**

<member-identifier> :: = <ID>

<int-identifier> :: = <member-identifier>

<bool-identifier> :: = <member-identifier>

<set-identifier> :: = <member-identifier>

<array-identifier> :: = <member-identifier>

<cnst-identifier> :: = <ID>**_CNST**

5.4 Basic data types

All **TYPES** shall be defined prior to use. The following basic types can be used to build more complex types:

<int> :: = <**INT8**>|
<INT16>|
<INT24>|
<INT32>|
<INT40>|
<INT48>|
<INT64>

<unsigned-int> :: = <**UINT8**>|
<UINT16>|
<UINT32>

<floating-point> :: = **<FLOAT32>**|
<FLOAT64>

<fill> :: = **<FILL8>**|
<FILL16>|
<FILL32>

<spec-type> :: = **<CHAR>**|
<NI_FMAT1>|
<NI_FMAT2>|
<LTIME_DATE>|
<STIME_DATE>|
<TIME>|
<DATE>|
<RDATE>

<base-type> :: = **<BCD>**|
<NIL>|
<int>|
<unsigned-int>|
<floating-point>|
<fill>|
<spec-type>

5.4.1 Basic data type definitions

The types are defined in Figure 4 as follows:

Type	Definition	Order of transmission
INT8	8 b signed integer, binary signed two's complement	Not applicable
INT16	16 b signed integer	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
INT24	24 b signed integer	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
INT32	32 b signed integer	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
INT40	40 b signed integer	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
INT48	48 b signed	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
INT64	64 b signed integer	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
UINT8	8 b unsigned integer	Not applicable
UINT16	16 b unsigned integer	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
UINT32	32 b unsigned integer	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
FLOAT32	Single precision floating point real number, per IEEE Std 754-1985	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
FLOAT64	Double precision floating point real number, per IEEE Std 754-1985	**DATA_ORDER** in **GEN_CONFIG_TBL** (Table 00)
FILL8	8 b of zeroes, used as space holder or filler	Not applicable
FILL16	16 b of zeroes, used as space holder or filler	Not applicable
FILL32	32 b of zeroes, used as space holder or filler	Not applicable

Figure 4—Basic type definitions and order of transmission

5.4.2 Special data types

In order to add flexibility to the structure of the tables, the following data types can be used. The specific definition is determined within the tables themselves.

5.4.2.1 Signed integers

To allow signed integer values to be easily transmitted without requiring data format translations, an indicator in **GEN_CONFIG_TBL** (Table 00) is provided. This indicator, **INT_FORMAT**, specifies the data

format for all **INTx** types within the tables. The presently delineated signed integer data formats are twos complement, ones complement and sign/magnitude.

Examples of signed integer formats are as follows. In these examples, the number –1 is used.

Twos complement for **INT8**:

1111 1111

Ones complement for **INT8**:

1111 1110

Sign/magnitude for **INT8**:

1000 0001

5.4.2.2 Characters

To allow different character sets to be used within the tables, a data type of **CHAR** has been defined. **CHAR** defines both the size in octets of the character and the character set being used. Clause 6 provides a definition for **CHAR**. The value of **CHAR** is defined in the **GEN_CONFIG_TBL** (Table 00) through the use of the variable **CHAR_FORMAT.**

5.4.2.3 Non-integer numbers

To allow non-integer numeric values to be transmitted by different methods within the tables, additional data types have been defined. **NI_FMAT1** and **NI_FMAT2** are used to transmit numeric values of varying precision. The value of the variables **NI_FORMAT1** and **NI_FORMAT2**, which are found in the **GEN_CONFIG_TBL** (Table 00), define how **NI_FMAT1** and **NI_FMAT2** are to be interpreted when reading from or writing to tables in an end device. Clause 6 provides definitions for **NI_FMAT1** and **NI_FMAT2** and for the different values of **NI_FORMAT1** and **NI_FORMAT2**.

5.4.2.4 Date and time formats

To allow date and time to be transmitted by different methods within the tables, additional data types have been defined. These are **LTIME_DATE**, **STIME_DATE**, **TIME**, **DATE**, and **RDATE**. The value of the variable **TM_FORMAT**, which is found in the **GEN_CONFIG_TBL** (Table 00), defines how the various date and time formats are to be interpreted when reading from or writing to tables in an end device. Clause 6 provides definitions for the date and time formats.

5.4.3 Miscellaneous data types

The following additional types are defined:

BCD Definition:		Each octet is two **BCD** values. The more significant value of each octet is in bits (7..4).
Binary bit values of	0000 to 1001	Used for the digits 0 to 9
	1010	Used for minus (–) sign
	1011	Used for blank ( ) character
	1100	Not used
	1101	Used for decimal point (typically, a period or comma)
	1110	Not used
	1111	Not used
Order of transmission:		Not applicable

NIL Definition: Indicator for field of length zero octets.

5.5 Values and constants

5.5.1 Constants

Constants are provided for making certain tasks of table specification easier, but not for transport. An example of this is the accessing of a bit position within a set. The bit field is specified within a given table by a <constant> that is equated to a particular number.

<constant_type> :: = **CONSTANTS** <constant> **END;**

<constant> :: = <cnst-identifier> = <number>**; |**
<constant>$^{+}$

The constants presently defined for the standard are found in 6.5.

5.5.2 Value

A value shall be a number, an <int-identifier> or an arithmetic operation between two values. Value is always an integer result.

<value> :: = <number> |
<cnst-identifier> |
<tbl-identifier>**.**<int-identifier> |
<bfld-identifier>**.**<int-identifier> |
-<value> |
<value> **+** <value> |
<value> **-** <value> |
<value> / <value> |
<value> * <value>

5.6 Conditionals

5.6.1 If Statements

The **IF** condition shall be a Boolean member of a bit field, or the value of a **SET**, or a relation between two values.

<condition> :: = <bfld-identifier>**.**<bool-identifier> |
<tbl-identifier>**.**<set-identifier>.(<value>)|
<value> **>** <value> |
<value> **>=** <value> |
<value> **<** <value> |
<value> **<=** <value> |
<value> **<>** <value> |
<value> **=** <value> |
<condition> **OR** <condition> |
<condition> **AND** <condition> |
<condition> **XOR** <condition> |
NOT <condition>

5.6.2 Case statements

A case expression identifies the member of a table that controls the case statement.

<case-expression> :: = <tbl-identifier>.<int-identifier> |
<bfld-identifier>.<int-identifier>

5.7 Bit field

Some data types do not always lend themselves to end on an octet boundary. Where these occur, they shall be logically grouped together into a bit field definition that shall end on an octet boundary. For purposes of description, the bit field is treated as the basic object for appearance in the table structure.

Figure 5 illustrates a subtype structure.

Most significant byte					Least significant byte
Octet	Octet	Octet	Octet	Octet	Octet
((n*8)–1)..((n*8)–9)	...	...	...	15..8	7..0

*n = # of octets used

Figure 5—Subtypes and bit field bit ordering

A bit field may contain one or more **IF** or **CASE** statements to modify its structure under some conditions. The dimension of a bit field and its octet ordering for the purpose of transport is defined in terms of a basic data type (see 5.4) as expressed in <bfld-type>.

<bfld-type> :: = **TYPE** <bfld-identifier> **=** <bfld>

<bfld> :: = **BIT FIELD OF** <unsigned-int> <bfld-member> **END;**

<bfld-member> :: = <member-identifier> : <sub-type>; |
<int-identifier> : **INT** (<value>..<value>); |
<int-identifier> : **UINT** (<value>..<value>); |
<bool-identifier> : **BOOL** (<value>); |
<bfld-case> |
<bfld-if> |
<bfld-member>$^{+}$

<sub-type> :: = **BOOL** (<value>)|
INT (<value>..<value>)|
UINT (<value>..<value>)|
FILL (<value>..<value>)

<bfld-if> :: = **IF** <condition> **THEN** <bfld-member> **END; |**
IF <condition> **THEN** <bfld-member> **ELSE** <bfld-member> **END;**

<bfld-case> :: = **CASE** <case-expression> **OF** <bfld-case-member> **END;**

<bfld-case-member> :: = <value> : <bfld-member> |
<value> - <value> : <bfld-member> |
<bfld-case-member>$^{+}$

5.7.1 Subtype definitions

BOOL(<value>)	Boolean operand of zero (0) for FALSE or one (1) for TRUE, in bit *x* of a bit field.
INT(<value>..<value>)	Signed binary integer of "n" bits, represented by a range of successive bits.
UINT(<value>..<value>)	Unsigned binary integer of "n" bits, represented by a range of successive bits.
FILL(<value>..<value>)	Fill of "n" bits, represented by a range of successive bits. Used to fill out unused bits in an octet, or as a separator in a bit field. The value of the fill bits is always zero (0).

5.8 Set

A <set> is a collection of booleans. It is defined with its dimension in octets.

<set> **::** = **SET(**<dim_octet>**)**

<dim_octet> **::** =<value>

When evaluated for sizing of a set dimension the evaluation of <value> shall produce a non-negative integer. An example follows that illustrates the method for specifying which member of a set is to be accessed.

Example:

```
FLAGS_USED: SET(16);
IF FLAGS_USED.1 THEN ...
IF GEN_CONFIG_TBL.UDT_0_CONST THEN ...
```

5.9 Array

Multiple repetitions of the same data type, used to describe a single variable may be grouped together in an array. Array elements are indexed starting with zero. Arrays are always transmitted from element zero to the last element, as shown in Figure 6. The order of transmission of an element is a function of the data type being used.

Element 0	Element 1	...	Element n–2	Element n–1

Figure 6—Single dimension array ordering

Arrays can be multidimensional. The right-most or last dimension is incremented prior to the first dimension. For a two-dimensional array, the first dimension can be thought of as a row and the second as a column. The array shall be transmitted in row order, as shown in Figures 7 and 8.

0, 0	0, 1
1, 0	1, 1

Figure 7— Two-dimension array

0, 0	0,1	1, 0	1, 1

Figure 8—Two-dimension array ordering

An array can be of a single dimension or of multiple dimensions separated by commas. Variables shall be defined before they are used in a dimension statement.

<array> :: = **ARRAY**[<dim>] **OF** <simple-type>

<simple-type> :: = <base-type> |
<rcd-identifier> |
<bfld-identifier> |
<tbl-identifier>.<rcd-identifier> |
<tbl-identifier>.<bfld-identifier>

<dim> :: = <value> |
<dim>[,<dim>]*

If the value of <dim> is zero then the array collapses to a zero element array and drops out of the table. When evaluated for sizing of an array the evaluation of <value> shall produce a non-negative integer.

When **CHAR** is used in an array, there is no justification implied. Full arrays are always transported. It is suggested that a blank ( ) be used as fill or pad.

5.10 Packed record

Packed records are used to group simple types, arrays, packed records, and bit fields together. A packed record can contain one or more **IF** statements or **CASE** statements to modify its structure upon some conditions. The dimension of a packed record will vary with applied conditions.

Bit fields and packed records shall be defined before they are used in a packed record.

<rcd-type> :: = **TYPE** <rcd-identifier> **=** <rcd>

<rcd> :: = **PACKED RECORD** <rcd-member>
END;

<rcd-member> :: = <member-identifier> **:** <simple-type>; |
<int-identifier> **:** <int>; |
<int-identifier> **:** <unsigned-int>; |
<set-identifier> **:** <set>; |
<array-identifier> **:** <array>; |
<rcd-case> |
<rcd-if> |
<rcd-member>+

<rcd-if> :: = **IF** (<condition>) **THEN** <rcd-member> **END; |**
IF (<condition>) **THEN** <rcd-member> **ELSE** <rcd-member> **END;**

<rcd-case> :: = **CASE** <case-expression> **OF** <rcd-case-member> **END;**

<rcd-case-member> :: = <value> **:** <rcd-member> |
<value>**..**<value> **:** <rcd-member> |
<rcd-case-member>+

5.11 Table

Tables are used to group simple types, arrays, packed records and bit fields together into a final structure.

Bit fields and packed records shall be defined before they are used in a table.

<types> :: = <rcd-type> |
<bfld-type> |
<types>+
<table> :: = <types> **TABLE** <tbl-identifier> **=** <rcd-identifier>;

6. Special data types

In order to add flexibility to the definition of some of the data types used within the tables, special data types are provided. Their definition, once selected within the **GEN_CONFIG_TBL** (Table 00), remains consistent throughout all of the remaining used tables.

6.1 Character set selection

This selection is used to determine the structure of characters used in the tables. **CHAR_FORMAT** is the controlling selector and **CHAR** is the data type. **CHAR_FORMAT** is specified in **GEN_CONFIG_TBL.**

Value of **CHAR_FORMAT**	Definition of **CHAR**
0	Unassigned
1	ISO 7-bit coded character set for information interchange, per ISO/IEC 646: 1991.[4]
2	ISO 8859-1: 1982 or ECMA-94 (June 1986).
3..7	Unassigned

6.2 Non-integer formats

This selection is used to determine the structure of non-integer numbers used in the tables. **NI_FORMAT1** and **NI_FORMAT2** are the controlling selectors, **NI_FMAT1** and **NI_FMAT2** provide type selection control for various data items. **NI_FORMAT1** and **NI_FORMAT2** are specified in **GEN_CONFIG_TBL**.

Value of **NI_FORMAT1** and **NI_FMAT2**	Definition of **NI_FMAT1** and **NI_FMAT2**
0	**FLOAT64**
1	**FLOAT32**
2	**ARRAY[12] OF CHAR**
3	**ARRAY[6] OF CHAR**
4	**INT32** [Implied decimal point between fourth and fifth digits from least significant digit. (e.g., 0.0001 is represented as 1)]
5	**ARRAY[6] OF BCD**

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

Value of **NI_FORMAT1** and **NI_FMAT2**	Definition of **NI_FMAT1** and **NI_FMAT2**
6	**ARRAY[4] OF BCD**
7	**INT24**
8	**INT32**
9	**INT40**
10	**INT48**
11	**INT64**
12..15	Unassigned

6.2.1 CHAR numbers

When **CHAR** is used and **NI_FORMAT1** or **NI_FORMAT2** has a value of 2 or 3, the number shall be represented in the following way:

<char_number> :: = [_*][+|-]<digit>+[.<digit>*][<exponent>][_*]

<exponent> :: = E|e|^[+|-]<digit>$^{+}$

_ :: = space character

A **CHAR** number representation may have any number of leading spaces; followed by an optional plus or minus sign; followed by a mandatory one or more digits; followed by an optional period and zero or more digits; followed by an optional exponent. An exponent is composed of the letter “E,” “e,” or “^”; followed by an optional plus or minus sign; followed by a mandatory one or more digits. Any number of spaces can lead or follow the <char_number> but no spaces are allowed to be embedded within the <char_number>. This format ensures that clear identification of any <char_number> is possible in a stream (sequence) of <char_number>$^{+}$, using the space character.

Examples:

Valid <char_number>
1.0E–7
123.6478e+03
1.2345
1.^3

Invalid <char_number>
.5
1.0 E–3
e+03

6.3 Date and time formats

This selection is used to determine the structure of dates and times used in the tables. **TM_FORMAT** is the controlling selector and **LTIME_DATE**, **STIME_DATE**, **TIME**, **RDATE**, and **DATE** are the resulting data types.

In general, the following is true about date and time formats:

TM_FORMAT	Data type used
0	No clock in the end device
1	**BCD** with discrete fields for month, day, year, hour, minute, and seconds.
2	**UINT8** with discrete fields for month, day, year, hour, minute, and seconds.
3	Binary value relative to a starting time as referenced by either **CLOCK_TBL** (Table 52) or **CLOCK_STATE_TBL** (Table 55): **LTIME_DATE:** 01/01/1970 @ 00:00:00, rounded to nearest second. **STIME_DATE:** 01/01/1970 @ 00:00:00, rounded to nearest minute. **TIME:** Time in seconds, since 00:00:00 local time.
4..7	Unassigned

The date and time structures are described below as packed records, but are used in the tables as data types.

6.3.1 LTIME_DATE, STIME_DATE, TIME types

```
TYPE LTIME_DATE = PACKED RECORD
      CASE GEN_CONFIG_TBL.TM_FORMAT OF
            0 :   NIL          : NIL;
            1 :   YEAR         : BCD;
                  MONTH        : BCD;
                  DAY          : BCD;
                  HOUR         : BCD;
                  MINUTE       : BCD;
                  SECOND       : BCD;
            2 :   YEAR         : UINT8;
                  MONTH        : UINT8;
                  DAY          : UINT8;
                  HOUR         : UINT8;
                  MINUTE       : UINT8;
                  SECOND       : UINT8;
            3 :   U_TIME       : UINT32;
                  SECOND       : UINT8
      END;
END;

TYPE STIME_DATE = PACKED RECORD
      CASE GEN_CONFIG_TBL.TM_FORMAT OF
            0 :   NIL          : NIL;
            1 :   YEAR         : BCD;
                  MONTH        : BCD;
                  DAY          : BCD;
                  HOUR         : BCD;
                  MINUTE       : BCD;
            2 :   YEAR         : UINT8;
                  MONTH        : UINT8;
                  DAY          : UINT8;
```

```
                  HOUR          : UINT8;
                  MINUTE        : UINT8;
            3 :   U_TIME        : UINT32;
      END;
END;

TYPE TIME = PACKED RECORD
      CASE GEN_CONFIG_TBL.TM_FORMAT OF
            0 :   NIL           : NIL;
            1 :   HOUR          : BCD;
                  MINUTE        : BCD;
                  SECOND        : BCD;
            2 :   HOUR          : UINT8;
                  MINUTE        : UINT8;
                  SECOND        : UINT8;
            3 :   D_TIME        : UINT32;
      END;
END;
```

Identifier	*Value*	*Definition*
YEAR	00..89	Years 2000..2089
	90..99	Years 1990..1999
	> 100	Unassigned
MONTH	0	Unassigned
	1..12	Month of year
	> 12	Unassigned
DAY	0	Unassigned
	1..31	Day of month
	> 31	Unassigned
HOUR	00..23	Hour of day, 24 h basis.
	> 23	Unassigned
MINUTE	00..59	Minute of hour
	> 59	Unassigned
SECOND	00..59	Seconds of minute
	> 59	Unassigned
U_TIME		Universal time in minutes, since 01/01/1970 @ 00:00:00 as referenced by either **CLOCK_TBL** (Table 52) or **CLOCK_STATE_TBL** (Table 55).
D_TIME		Time in seconds, since 00:00:00 local time.

6.3.2 RDATE type

This type defines a recurrent date. This recurrence can be yearly, monthly, weekly, or based on a constant cycle of up to 64 days.

```
TYPE RDATE = BIT FIELD OF UINT16
      MONTH                         : UINT(0..3);
      CASE MONTH OF
            1..13 :     OFFSET      : UINT(4..7);
```



```
                        WEEKDAY     : UINT(8..10);
                        DAY         : UINT(11..15);
              14 :      FILLER1     : FILL(4..7);
                        WEEKDAY     : UINT(8..10);
                        FILLER2     : FILL(11..15);
              15 :      PERIOD      : UINT(4..9);
                        DELTA       : UINT(10..15);
      END;
END;.
```

Identifier	*Value*	*Definition*
MONTH	0	Unassigned
	1..12	Month of year
	13	Action is repeated monthly
	14	Action is repeated weekly
	15	Action is repeated each **PERIOD** referenced by an **ANCHOR_DATE** plus **DELTA**. The **ANCHOR_DATE** shall be supplied separately.
OFFSET	0	No offset
	1	Advance to **WEEKDAY** before date entered
	2	Postpone to the first **WEEKDAY** on or after date entered
	3	Postpone to the second **WEEKDAY** on or after date entered
	4	Postpone to the third **WEEKDAY** on or after date entered
	5	Postpone to the fourth **WEEKDAY** on or after date entered
	6	Postpone to the last **WEEKDAY** of the **MONTH** on or after date entered
	7	Observe on date entered as well as day following date entered
	8	Postpone to Monday if Sunday
	9	Advance to Friday if Sunday
	10	Postpone to Monday if Saturday
	11	Advance to Friday if Saturday
	12	Postpone to Monday if Sunday or Saturday
	13	Advance to Friday if Sunday or Saturday
	14	Postpone to Monday if Sunday, advance to Friday if Saturday
	15	Do not observe date entered. Observe on day following date entered.
WEEKDAY	0..6	Sunday to Saturday
	7	Unassigned
PERIOD	0..63	0 to 63 days
DELTA	0..63	0 to 63 days

6.3.3 DATE Type

This type defines a non-recurrent date using a field having the same size as the recurrent date type "**RDATE**."

```
TYPE DATE = BIT FIELD OF UINT16
      YEAR        : UINT(0..6);
      MONTH       : UINT(7..10);
      DAY         : UINT(11..15);
END;
```

Identifier	*Value*	*Definition*
YEAR	00..89	Years 2000..2089
	90..99	Years 1990..1999
	> 100	Unassigned
MONTH	0	Unassigned
	1..12	Month of year
	> 12	Unassigned
DAY	0	Unassigned
	1-31	Day of month

6.4 Common table or procedure identifier formats

These types are provided to ease the specification of table access parameters and provide consistent specification of these table access parameters throughout the standard.

6.4.1 TABLE_IDA_BFLD Bit field

Type **TABLE_IDA_BFLD** provides the table or procedure number; a flag indicating standard or manufacturer table or procedure; a flag indicating pending status; and three additional fields for definition and use within the defining data structure.

```
TYPE TABLE_IDA_BFLD = BIT FIELD OF UINT16
      TBL_PROC_NBR      : UINT(0..10);
      STD_VS_MFG_FLAG   : BOOL(11);
      PENDING_FLAG      : BOOL(12);
      FLAG1             : BOOL(13);
      FLAG2             : BOOL(14);
      FLAG3             : BOOL(15);
END;
```

6.4.2 TABLE_IDB_BFLD Bit field

Type **TABLE_IDB_BFLD** provides the table or procedure number; a flag indicating standard or manufacturer table or procedeure; and a four bit unsigned integer for definition and use within the defining data structure.

```
TYPE TABLE_IDB_BFLD = BIT FIELD OF UINT16
      TBL_PROC_NBR      : UINT(0..10);
      STD_VS_MFG_FLAG   : BOOL(11);
      SELECTOR          : UINT(12..15);
END;
```

6.4.3 TABLE_IDC_BFLD Bit field

Type **TABLE_IDC_BFLD** provides the table or procedure number; a flag indicating standard or manufacturer; a flag indicating table or procedure; and three additional fields for definition and use within the defining data structure.

```
TYPE TABLE_IDC_BFLD = BIT FIELD OF UINT16
      TBL_PROC_NBR      : UINT(0..10);
      STD_VS_MFG_FLAG   : BOOL(11);
```



```
      PROC_FLAG          : BOOL(12);
      FLAG1              : BOOL(13);
      FLAG2              : BOOL(14);
      FLAG3              : BOOL(15);
END;
```

6.5 Constants

The following constants are defined for use in these tables. Note that these constants are not transported from device to device, but merely serve as an aid in the logical specification or selection of tables. Values for the following constants are expressed in decimal form.

```
TYPE CONSTANTS
      FALSE_CNST                                = 00;
      TRUE_CNST                                 = 01;

      GEN_CONFIG_TBL_CNST                       = 00;
      GENERAL_MFG_ID_TBL_CNST                   = 01;
      DEVICE_NAMEPLATE_TBL_CNST                 = 02;
      ED_MODE_STATUS_TBL_CNST                   = 03;
      PENDING_STATUS_TBL_CNST                   = 04;
      DEVICE_IDENT_TBL_CNST                     = 05;
      UTIL_INFO_TBL_CNST                        = 06;
      PROC_INITIATE_TBL_CNST                    = 07;
      PROC_RESPONSE_TBL_CNST                    = 08;

      DIM_SOURCES_LIM_TBL_CNST                  = 10;
      ACT_SOURCES_LIM_TBL_CNST                  = 11;
      UOM_ENTRY_TBL_CNST                        = 12;
      DEMAND_CONTROL_TBL_CNST                   = 13;
      DATA_CONTROL_TBL_CNST                     = 14;
      CONSTANTS_TBL_CNST                        = 15;
      SOURCES_TBL_CNST                          = 16;

      DIM_REGS_TBL_CNST                         = 20;
      ACT_REGS_TBL_CNST                         = 21;
      DATA_SELECTION_TBL_CNST                   = 22;
      CURRENT_REG_DATA_TBL_CNST                 = 23;
      PREVIOUS_SEASON_DATA_TBL_CNST             = 24;
      PREVIOUS_DEMAND_RESET_DATA_TBL_CNST       = 25;
      SELF_READ_DATA_TBL_CNST                   = 26;
      PRESENT_REGISTER_SELECT_TBL_CNST          = 27;
      PRESENT_REGISTER_DATA_TBL_CNST            = 28;

      DIM_DISP_TBL_CNST                         = 30;
      ACT_DISP_TBL_CNST                         = 31;
      DISP_SOURCE_TBL_CNST                      = 32;
      PRI_DISP_LIST_TBL_CNST                    = 33;
      SEC_DISP_LIST_TBL_CNST                    = 34;

      DIM_SECURITY_LIMITING_TBL_CNST            = 40;
      ACT_SECURITY_LIMITING_TBL_CNST            = 41;
      SECURITY_TBL_CNST                         = 42;
```

```
        DEFAULT_ACCESS_CONTROL_TBL_CNST          = 43;
        ACCESS_CONTROL_TBL_CNST                  = 44;
        KEY_TBL_CNST                             = 45;

        DIM_TIME_TOU_TBL_CNST                    = 50;
        ACT_TIME_TOU_TBL_CNST                    = 51;
        CLOCK_TBL_CNST                           = 52;
        TIME_OFFSET_TBL_CNST                     = 53;
        CALENDAR_TBL_CNST                        = 54;
        CLOCK_STATE_TBL_CNST                     = 55;
        TIME_REMAIN_TBL_CNST                     = 56;

        DIM_LP_TBL_CNST                          = 60;
        ACT_LP_TBL_CNST                          = 61;
        LP_CTRL_TBL_CNST                         = 62;
        LP_STATUS_TBL_CNST                       = 63;
        LP_DATA_SET1_TBL_CNST                    = 64;
        LP_DATA_SET2_TBL_CNST                    = 65;
        LP_DATA_SET3_TBL_CNST                    = 66;
        LP_DATA_SET4_TBL_CNST                    = 67;

        DIM_LOG_TBL_CNST                         = 70;
        ACT_LOG_TBL_CNST                         = 71;
        EVENTS_ID_TBL_CNST                       = 72;
        HISTORY_LOG_CTRL_TBL_CNST                = 73;
        HISTORY_LOG_DATA_TBL_CNST                = 74;
        EVENT_LOG_CTRL_TBL_CNST                  = 75;
        EVENT_LOG_DATA_TBL_CNST                  = 76;

        DIM_UDT_FUNC_LIM_TBL_CNST                = 80;
        ACT_UDT_FUNC_LIM_CNST                    = 81;
        UDT_LIST_TBL_CNST                        = 82;
        UDT_SET_TBL_CNST                         = 83;
        UDT_0_TBL_CNST                           = 84;
        UDT_1_TBL_CNST                           = 85;
        UDT_2_TBL_CNST                           = 86;
        UDT_3_TBL_CNST                           = 87;
        UDT_4_TBL_CNST                           = 88;
        UDT_5_TBL_CNST                           = 89;
END;
```

7. Compliance

A device is considered to be in compliance with this standard if all of the following conditions are met:

a) The device shall accept and act upon all services defined in Clause 8.

b) **GEN_CONFIG_TBL** (Table 00):

   1) Two-way communicating devices shall transmit **GEN_CONFIG_TBL** (Table 00) on request;

   2) Suppliers of one-way communicating devices shall, as a minimum, supply **GEN_CONFIG_TBL** (Table 00) in some form.

c) Any table transported between devices shall be either a standard table as defined by this standard or a manufacturer table defined by the device manufacturer. All standard and manufacturer tables supported for the current device implementation shall be specified in **GEN_CONFIG_TBL** (Table 00).

d) Any procedure written to the device shall be either a standard procedure as defined by this standard or a manufacturer procedure defined by the device manufacturer. All standard and manufacturer procedures supported for the current device implementation shall be specified in **GEN_CONFIG_TBL** (Table 00).

e) Some standard tables contain data structures that can be collapsed by a control flag. The control flag shall be determinable based on this standard. Collapsed fields shall not be transported. An array of zero elements is a collapsed field.

f) A device is not required to use every data structure within a table; however, it shall transport all non-collapsed data structures within a table. Unused fields are considered in compliance if byte count and format are correct and if manufacturers identify unused fields. It is recommended that unused fields be filled with zeros or spaces.

g) This standard specifies variable formats for some data structures. The variable formats shall be consistent with the format as specified in **GEN_CONFIG_TBL** (Table 00). Refer to Clauses 5 and 6 for details.

h) Dimensional function limiting tables (FLC) and actual function limiting tables (FLC+1) specify the designed and usable limits for variables used within a given decade. (For a complete explanation of function limiting tables, see 4.1.) The permissible combinations are listed in Figure 1.

8. Table transportation issues

The effective transport of table structures is dependent only on the presence of basic read and write services. While the marketplace may dictate the incorporation of more sophisticated services, it is left to the implementors of specific protocol stacks to select the specific read and write services to be included.

8.1 Minimum services and parameters

8.1.1 Read service

This service causes the transfer of data from the initiating device to the target device and is required for two-way communications.

Read request — The read request shall be capable of both a complete and a partial table read. This is accomplished by one required parameter and optional parameters. The required parameter is a table identifier. Optional parameters define partial table retrieval.

Table identifier — This parameter has a range of 0–8191. This range is further subdivided as follows:

a) The range from 0–2047 provides access to standard tables 0–2047
b) The range from 2048–4095 provides access to manufacturer tables 0– 2047.
c) The range 4096–6143 provides access to standard pending tables 0–2047.
d) The range 6144–8191 provides access to manufacturer pending tables 0– 2047.

Partial table retrieval — The retrieval of a portion of a table, a partial table, is possible through the use of one of two methods. These methods are offset-count and index-count. The absence of either of these methods implies the entire table shall be read.

Offset count method The offset-count method is based upon supplying an offset from the beginning of the selected table and, optionally, a count of the number of octets to read from the offset point. The range of the offset from the beginning of the table is 0 to a minimum of 65 535 octets. The range of the count of the number of octets to read from the offset point is 0 to a minimum of 65 535 octets. A zero for the count value or the absence of the count value shall cause will imply to read the entire remaining table to be read starting atfrom the offset point.

Indexed-count method The index-count method is based upon accommodating a minimum of 5 indices relative to the first data structure within the table and, optionally a count of the number of octets to read from the index point. Each index shall range from 0 to a maximum of 65 535 octets. The range of the count of the number of octets to read from the index point is 0 to a minimum of 65 535 octets. A zero for the count value or the absence of the count value shall cause to read the entire remaining table to be read starting at the index point.

Read response data The read response shall contain as a minimum the table data selected by the aforementioned parameters supplied with the read request.

When the read request table identifier specifies a pending table, a six-octet pending event description precedes the requested information. The pending event description parameters are described in 8.2.

8.1.2 Write service

This service causes the transfer of unrequested data to a target device from the initiating device and is required for both one- and two-way communications.

Write request The write request shall be capable of both a complete and a partial table write. This is accomplished by one required parameter and partial table identifiers and the write data field. The required parameter is a table identifier. Optional parameters define partial table write.

Table identifier

a) This parameter has a range of 0–8191. This range is further subdivided as follows:
b) The range from 0–2047 provides access to standard tables 0–2047.
c) The range from 2048– 4095 provides access to manufacturer tables 0–2047.
d) The range 4096–6143 provides access to standard pending tables 0–2047. Several additional parameters are required for writing to a pending table. These parameters are appended to the front of the table data to be written and are described in the Table Data Write section.
e) The range 6144 –8191 provides access to manufacturer pending tables 0–2047. Several additional parameters are required for writing to a pending table. These parameters are appended to the front of the table data to be written as a pending event description and are described in 8.2.

Partial table identifier The writing of a portion of a table, a partial table, is possible through the use of one of two methods. These methods are offset-count and index-count. The absence of either of these methods implies the entire table shall be written.

Offset-count method The offset-count method is based upon supplying an offset from the beginning of the selected table and, optionally, a count of the number of octets to write to the offset point. The range of the offset from the beginning of the table is 0 to a minimum of 65 535 octets. The range of the count of the number of octets to write to the offset point is 0 to a minimum of 65 535 octets. A zero for the count value, or the absence of the count value, shall cause to write the entire remaining table to be written from the offset point.

Indexed-count method The indexed method is based upon accommodating a minimum of 5 indices relative to the first data structure within the table and, optionally, a count of the number of octets to write to the index point. Each index shall range from 0 to a maximum of 65 535 octets. The range of the count of the number of octets to write shall range from 0 to a maximum of 65 535 octets. A zero for the count value or the absence of the count value shall imply writing the entire remaining table from the index point.

Data The data field contains the information to write.

When the write request table identifier specifies a pending table, a six-octet pending event description precedes the information in the data field.

Write response As a minimum, no response is required to the write request. This does not preclude use of write responses supported by implementor languages or their supporting protocols.

8.2 Pending event description

Data read from or written into a pending table is identical to data read from or written into non-pending tables in structure, with the exception of the addition of a preceding pending event descriptor.

```
TYPE  STATUS_BFLD  = BIT FIELD OF UINT8
      EVENT_CODE               : UINT(0..3);
      SELF_READ_FLAG:          : BOOL(4);
      DEMAND_RESET_FLAG:       : BOOL(5);
      RESERVED                 : FILL(6..7);
END;

TYPE  PENDING_EVENT_DESC_RCD = PACKED RECORD
      EVENTS_SELECTOR:         : STATUS_BFLD;
      EVENT_STORAGE:           : ARRAY[5] OF UINT8;
END;
```

Identifier	*Value*	*Definition*
STATUS_BFLD		Selectors indicating action to be performed upon activation of selected table.

EVENT_CODE		Variable that specifies the condition upon which the pending table shall be made active. The trigger condition is stored in the **EVENT_STORAGE** variable.
	0	The pending table is activated based on an absolute time contained in the **EVENT_STORAGE** variable.
	1	The pending table is activated based on a relative time contained in the **EVENT_STORAGE** variable.
	2	The pending table is activated based on a non-time associated trigger contained in the **EVENT_STORAGE** variable.
	3..15	Reserved.
SELF_READ_FLAG	FALSE	Do not perform a self read before pending table is activated.
	TRUE	Perform a self read, if capable, before pending table is activated.
DEMAND_RESET_FLAG	FALSE	Do not perform a demand reset before pending table is activated.
	TRUE	Perform a demand reset, if capable, before pending table is activated.
RESERVED		Reserved for future use.
PENDING_EVENT_DESC_RCD		
EVENTS_SELECTOR		Status bits indicating various conditions associated with a pending table.
EVENT_STORAGE		Variable for storage of the trigger condition that causes a pending table to become active.
STIME_DATE	0	For an **EVENT_CODE** of 0 this variable contains the year, month, day, hour, and minute in the format indicated.
ARRAY[5] OF UINT8	1	For an **EVENT_CODE** of 1 this variable contains an offset time where the array positions indicate the following: [0] weeks; [1] days; [2] hours; [3] minutes; and [4] seconds.
CHAR	2	For an **EVENT_CODE** of 2 this variable contains in the first four characters the ASCII manufacturer code per Annex A of this standard. The final character is an event number as defined by the manufacturer.

9. Tables

9.1 DECADE 00: Configuration tables

This decade contains tables associated with end device configuration, identification, procedures and responses, and information required for data manipulation due to hardware configurations.

9.1.1 TABLE 00 General configuration table

Table 00 Data Description

GEN_CONFIG_TBL (Table 00) contains general end device configuration information. It also establishes the total set of tables, procedures, and the selection of special types used in the end device. If a default set is used, it is indicated by this table.

```
TYPE FORMAT_CONTROL_1_BFLD = BIT FIELD OF UINT8
      DATA_ORDER               : UINT(0..0);
      CHAR_FORMAT              : UINT(1..3);
      MODEL_SELECT             : UINT(4..6);
      FILLER                   : FILL(7..7);
END;

TYPE FORMAT_CONTROL_2_BFLD = BIT FIELD OF UINT8
      TM_FORMAT                : UINT(0..2);
      DATA_ACCESS_METHOD       : UINT(3..4);
      ID_FORM                  : UINT(5..5);
      INT_FORMAT               : UINT(6..7);
END;

TYPE FORMAT_CONTROL_3_BFLD = BIT FIELD OF UINT8
      NI_FORMAT1               : UINT(0..3);
      NI_FORMAT2               : UINT(4..7);
END;

TYPE GEN_CONFIG_RCD = PACKED RECORD
      FORMAT_CONTROL_1         : FORMAT_CONTROL_1_BFLD;
      FORMAT_CONTROL_2         : FORMAT_CONTROL_2_BFLD;
      FORMAT_CONTROL_3         : FORMAT_CONTROL_3_BFLD;
      MANUFACTURER             : ARRAY[4] OF CHAR;
      NAMEPLATE_TYPE           : UINT8;
      DEFAULT_SET_USED         : UINT8;
      MAX_PROC_PARM_LENGTH     : UINT8;
      MAX_RESP_DATA_LEN        : UINT8;
      STD_VERSION_NO           : UINT8;
      STD_REVISION_NO          : UINT8;
      DIM_STD_TBLS_USED        : UINT8;
      DIM_MFG_TBLS_USED        : UINT8;
      DIM_STD_PROC_USED        : UINT8;
      DIM_MFG_PROC_USED        : UINT8;
      DIM_MFG_STATUS_USED      : UINT8;
      NBR_PENDING              : UINT8;
      STD_TBLS_USED            : SET(GEN_CONFIG_TBL.DIM_STD_TBLS_USED);
      MFG_TBLS_USED            : SET(GEN_CONFIG_TBL.DIM_MFG_TBLS_USED);
      STD_PROC_USED            : SET(GEN_CONFIG_TBL.DIM_STD_PROC_USED);
      MFG_PROC_USED            : SET(GEN_CONFIG_TBL.DIM_MFG_PROC_USED);
      STD_TBLS_WRITE           : SET(GEN_CONFIG_TBL.DIM_STD_TBLS_USED);
      MFG_TBLS_WRITE           : SET(GEN_CONFIG_TBL.DIM_MFG_TBLS_USED);
END;

TABLE GEN_CONFIG_TBL = GEN_CONFIG_RCD;
```

Identifier	*Value*	*Definition*
FORMAT_CONTROL_1_BFLD		Variable to define order of multibyte numeric data transfer.
DATA_ORDER	0	Least significant byte first.
	1	Most significant byte first.
CHAR_FORMAT		See Character Set Selection in Special Data Types, 6.1. Unsigned binary code where:
	0	Unassigned.
	1	ISO 7-bit coded character set for information interchange, per ISO/IEC 646: 1991.
	2	ISO 8859-1: 1987 or ECMA-94 (June 1986).
	3..7	Unassigned.
MODEL_SELECT		Unsigned binary code where:
	0	A code indicating the I/O model selects sources as an 8 b index into the **DATA_CONTROL_TBL** (Table 14).
	1..7	Reserved
FORMAT_CONTROL_2_BFLD		
TM_FORMAT		Data type used for dates and time:
	0	No clock in the end device
	1	**BCD** with discrete fields for month, day, year, hour, minute and seconds.
	2	**UINT8** with discrete fields for month, day, year, hour, minute and seconds.
	3	**UINT32** with time referenced as seconds relative to a starting time. The reference point is as follows: **LTIME_DATE:** 01/01/1970 @ 00:00:00 Greenwich Mean Time (GMT),rounded to nearest second. **STIME_DATE:** 01/01/1970 @ 00:00:00 GMT, rounded to nearest minute. **TIME:** Time in seconds since 00:00:00 local time (including time zone offset).
	4..7	Unassigned
DATA_ACCESS_METHOD		Variable to designate method of passing the table data after **GEN_CONFIG_TBL** (Table 00) has been interrogated. NOTE—All data access methods support complete table access.
	0	Complete table access only. No partial table access supported.
	1	Offset-count access method is supported.
	2	Index-count method is supported.
	3	Both methods 1 and 2 of data access method are supported.
ID_FORM		Indicates the format of the identifier field in **DEVICE_IDENT_TBL** (Table 05).

	0	**CHAR** string.
	1	**BCD** string.
INT_FORMAT		Indicates the format of signed integer types.
		Signed integer types are represented in twos complement format.
		Signed integer types are represented in ones complement format.
		Signed integer types are represented in sign/ magnitude format.
		Reserved.
FORMAT_CONTROL_3_BFLD		
NI_FORMAT1		Indicates the type of non-integer value used throughout the tables where specified as **NI_FMAT1**. Refer to 6.2 for details.
NI_FORMAT2		Indicates the type of non-integer value throughout the tables where shown as **NI_FMAT2**. Refer to 6.2 for details.
GEN_CONFIG_RCD		
FORMAT_CONTROL_1		See descriptions above.
FORMAT_CONTROL_2		See descriptions above.
FORMAT_CONTROL_3		See descriptions above.
MANUFACTURER		end device manufacturer identification. Refer to Annex A for list of manufacturer identifiers.
NAMEPLATE_TYPE		Entry used to select the nameplate record structure to be used in **DEVICE_NAMEPLATE_TBL** (Table 02).
	0	Gas
	1	Water
	2	Electric
	3..255	Unassigned
DEFAULT_SET_USED		Indicates which, if any, default sets are used. See Annex C for the default set definitions.
	0	No default values in use. See 4.1, Figure 1, conditions C, D, E, and F for more detail.
	1	Default set #1, Simple Meter Register, in use.
	2	Default set #2, Simple Demand Meter, in use.
	3	Default set #3, Simple TOU Meter, in use.
	4	Default set #4, Simple Profile Recorder, in use.
	5..255	Reserved for default table value sets.
MAX_PROC_PARM_LENGTH		Manufacturer defined maximum length for argument parameters passed to procedures in **PROC_INITIATE_TBL** (Table 07).
MAX_RESP_DATA_LEN		Manufacturer defined maximum length representing length returned by procedures in **PROC_RESPONSE_TBL** (Table 08).

STD_VERSION_NO — Unsigned binary number designating the version of the particular industry set of standard tables.

0	Pre-release or informal release document.
1	Original standard table set, first release document.
2..255	Reserved for Standards Committee use only.

STD_REVISION_NO — Unsigned binary number that designates a minor change to a version of this standard. Within a given version of this standard, all revisions with a lower revision number shall be backward compatible. If this is not true then a new version number shall be required.

DIM_STD_TBLS_USED — Octets required to represent the set of standard tables used in **GEN_CONFIG_TBL.STD_TBLS_USED** (Table 00). This is a mechanism to allow the table identifier set to expand in size from simple devices to complex devices.

DIM_MFG_TBLS_USED — Octets required to represent the set of manufacturer tables used in **GEN_CONFIG_TBL.MFG_TBLS_USED** (Table 00). This is a mechanism to allow the table identifier set to expand in size from simple devices to complex devices.

DIM_STD_PROC_USED — Octets required to represent the set of standard procedures used in **GEN_CONFIG_TBL.STD_PROC_USED** (Table 00). This is a mechanism to allow the standard defined procedure identifier set to expand in size.

DIM_MFG_PROC_USED — Octets required to represent the set of manufacturer procedures used in **GEN_CONFIG_TBL.MFG_PROC_USED** (Table 00). This is a mechanism to allow the manufacturer defined procedure identifier set to expand in size.

DIM_MFG_STATUS_USED

0..255	The number of octets allocated for indicating manufacturer specific status flags.

NBR_PENDING — Number of pending status sets in **PENDING_STATUS_TBL** (Table 04).

STD_TBLS_USED — This set variable indicates which of the standard tables are implemented by the device. Tables are represented by bits 0 through (8*__DIM_STD_TBLS_USED__ –1), with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.

MFG_TBLS_USED	This set variable indicates which of the manufacturer tables are implemented by the device. Tables are represented by bits 0 through (8*__DIM_MFG_TBLS_USED__ –1), with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.
STD_PROC_USED	This set variable indicates which of the standard procedures are implemented by the device. Procedures are represented by bits 0 through (8*__DIM_STD_PROC_USED__ –1), with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.
MFG_PROC_USED	This set variable indicates which of the manufacturer's procedures are implemented by the device. Procedures are represented by bits 0 through (8*__DIM_MFG_PROC_USED__ -1), with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.
STD_TBLS_WRITE	This set variable indicates which of the standard tables are capable of being rewritten. Tables are represented by bits 0 through (8*__DIM_STD_TBLS_USED__ –1), with a 1 representing a writeable table and a zero representing a table that cannot be written.
MFG_TBLS_WRITE	This set variable indicates which of the standard tables are capable of being rewritten. Tables are represented by bits 0 through (8*__DIM_MFG_TBLS_USED__ –1), with a 1 representing a writeable table and a zero representing a table that cannot be written.

9.1.2 TABLE 01 General manufacturer identification table

Table 01 Data description

GENERAL_MFG_ID_TBL (Table 01) is a general manufacturer identification data table for water, gas, and electric utility end devices. The values of the data items shall be set up by the manufacturer and left unchanged since they are determined by the functionality of the device.

```
TYPE MANUFACTURER_IDENT_RCD = PACKED RECORD
      MANUFACTURER             : ARRAY[4] OF CHAR;
      ED_MODEL                 : ARRAY[8] OF CHAR;
      HW_VERSION_NUMBER        : UINT8;
      HW_REVISION_NUMBER       : UINT8;
      FW_VERSION_NUMBER        : UINT8;
      FW_REVISION_NUMBER       : UINT8;
      IF GEN_CONFIG_TBL.ID_FORM THEN
```

```
            MFG_SERIAL_NUMBER : ARRAY[8] OF BCD;
      ELSE
            MFG_SERIAL_NUMBER : ARRAY[16] OF CHAR;
      END;
END;

TABLE GENERAL_MFG_ID_TBL = MANUFACTURER_IDENT_RCD;
```

Identifier	*Value*	*Definition*
MANUFACTURER_IDENT_RCD		
MANUFACTURER		Name of manufacturer. Refer to Annex A.
ED_MODEL		Model identifier of the end device left justified. For example, 'ENG M3.'
HW_VERSION_NUMBER	0..255	Manufacturer's hardware version number. Implies functional changes.
HW_REVISION_NUMBER	0..255	Manufacturer's hardware revision number. Implies product corrections or improvements. Recommend use of "0" to designate prototyping device.
FW_VERSION_NUMBER	0..255	Manufacturer's firmware version number. Changes in value imply functional differences.
FW_REVISION_NUMBER	0..255	Manufacturer's firmware revision number. Changes in value imply product corrections or improvements. Recommend use of "0" to designate preproduction or prototyping software.
MFG_SERIAL_NUMBER		Manufacturer's serial number for the end device. 16 **BCD** digits **OR CHARs**.

9.1.3 TABLE 02 Device nameplate table

Table 02 Data description

DEVICE_NAMEPLATE_TBL (Table 02) is the nameplate data table. This table contains a record structure for each type of end device (e.g., water, gas, electric). The structure to be used is specified in **GEN_CONFIG_TBL.NAMEPLATE_TYPE** (Table 00). This table presently has nameplate structures defined for an electric meter, a gas meter, and a water meter.

{water device structures}

```
TYPE W_WATER_DEVICE_BFLD = BIT FIELD OF UINT16
      W_ED_TYPE          : UINT(0..2);
      W_FLUID_TYPE       : UINT(3..6);
      W_ED_DRIVE         : UINT(7..10);
      W_ED_PIPE_SIZE     : UINT(11..15);
END;
```

{gas device structures}

```
TYPE G_ED_TYPE_BFLD = BIT FIELD OF UINT8
      G_ED_TYPE          : UINT(0..2);
      G_MECH_FORM        : UINT(3..5);
```

```
      G_ENG_METRIC      : UINT(6..6);
      FILLER            : FILL(7..7);
END;

TYPE G_PRESSURE_RCD = PACKED RECORD
      G_MAX_PRESS       : NI_FMAT2;
      G_UOM_PRESS       : UINT8;
END;
TYPE G_FLOW_RCD = PACKED RECORD
      G_MAX_FLOW        : NI_FMAT2;
      G_UOM_FLOW        : UINT8;
END;

TYPE G_SIZE_BFLD = BIT FIELD OF UINT8
      G_GEAR_DRIVE            : UINT(0..2);
      G_INPUT_OUTPUT_PIPE     : UINT(3..7);
END;

TYPE G_COMPENSATION_BFLD = BIT FIELD OF UINT8
      G_COMP_TEMP             : UINT(0..4);
      G_COMP_PRESS            : UINT(5..7);
END;

TYPE G_GAS_DEVICE_RCD = PACKED RECORD
      G_ED_TYPE               : G_ED_TYPE_BFLD;
      G_MAX_PRESS             : G_PRESSURE_RCD;
      G_MAX_FLOW              : G_FLOW_RCD;
      G_GEAR_PIPE_SIZE        : G_SIZE_BFLD;
      G_COMPENSATION          : G_COMPENSATION_BFLD;
END;

{electric device structures}
TYPE E_ELEMENTS_BFLD = BIT FIELD OF UINT16
      E_FREQ                  : UINT(0..2);
      E_NO_OF_ELEMENTS        : UINT(3..5);
      E_BASE_TYPE             : UINT(6..9);
      E_ACCURACY_CLASS        : UINT(10..15);
END;

TYPE E_VOLTS_BFLD = BIT FIELD OF UINT8
      E_ELEMENTS_VOLTS        : UINT(0..3);
      E_ED_SUPPLY_VOLTS       : UINT(4..7);
END;

TYPE E_AMPS_RCD = PACKED RECORD
      E_CLASS_MAX_AMPS        : ARRAY[6] OF CHAR NI_FMAT2;
      E_TA                    : ARRAY [6] OF CHAR NI_FMAT2;
END;

TYPE E_ELECTRIC_DEVICE_RCD = PACKED RECORD
      E_kh                    : ARRAY [6] OF CHAR NI_FMAT2;
      E_kt                    : ARRAY [6] OF CHAR NI_FMAT1;
      E_INPUT_SCALAR          : UINT8;
      E_ED_CONFIG             : ARRAY[5] OF CHAR;
```

```
      E_ELEMENTS              : E_ELEMENTS_BFLD;
      E_VOLTS                 : E_VOLTS_BFLD;
      E_AMPS                  : E_AMPS_RCD;
END;

TYPE DEVICE_DEFINITION_RCD = PACKED RECORD
      CASE GEN_CONFIG_TBL.NAME_PLATE_TYPE OF
            0           : G_GAS_DEVICE          :G_GAS_DEVICE_RCD;
            1           : W_WATER_DEVICE        :W_WATER_DEVICE_BFLD;
            2           : E_ELECTRIC_DEVICE     :E_ELECTRIC_DEVICE_RCD;
            3..255      : RESERVED              :NIL;
      END;
END;

TABLE DEVICE_NAMEPLATE_TBL = DEVICE_DEFINITION_RCD;
```

The following structures are generally associated with water utility end devices. The values of the data items shall be set up by the manufacturer and left unchanged since they are determined by the functionality of the device.

Identifier	*Value*	*Definition*
W_WATER_DEVICE_BFLD		
W_ED_TYPE		Code representing water end device type.
	0	Meter
	1	Valve
	2	Pressure Regulator
	3	Pressure gauge
	4	Backflow
	5	Other
	6-7	Unassigned
W_FLUID_TYPE		Code representing type of fluid measured by water end device.
	0	Potable water
	1	Hot water
	2	Non-potable water
	3	Sewage primary water
	4	Sewage secondary water
	5	Sewage tertiary water
	6	Other
	7	Unassigned
W_ED_DRIVE		Code representing water end device drive.
	0	Piston
	1	Disc
	2	Multijet
	3	Turbine
	4	Compound
	5	Propeller
	6	Ultrasonic
	7	Magnetic-coupled
	8	Differential pressure
	9	Mass

	10	Variable area
	11	Open channel
	12	Oscillatory
	13	Other
	14..15	Unassigned

W_ED_PIPE_SIZE		Code representing end device pipe size.	
	0	1/2 in	13 mm
	1	5/8 in	15 mm
	2	3/4 in	20 mm
	3	1 in	25 mm
	4	1 1/2 in	40 mm
	5	2 in	50 mm
	6	3 in	80 mm
	7	4 in	100 mm
	8	6 in	160 mm
	9	8 in	200 mm
	10	10 in	250 mm
	11	12 in	300 mm
	12	14 in	350 mm
	13	16 in	400 mm
	14	18 in	450 mm
	15	20 in	500 mm
	16	22 in	Not standard
	17	24 in	600 mm
	18	26 in	Not standard
	19	28 in	Not standard
	20	30 in	Not standard
	21	32 in	800 mm
	22	34 in	Not standard
	23	36 in	900 mm
	24	40 in	1000 mm
	25	48 in	1200 mm
	26..31	Unassigned	

The following structures are generally associated with gas utility end devices. The values of the data items shall be set up by the manufacturer and left unchanged since they are determined by the functionality of the device.

Identifier	*Value*	*Definition*
G_ED_TYPE_BFLD		
G_ED_TYPE		Code representing gas end device type.
	0	Unclassified
	1	Gas meter
	2	Gas pressure regulator
	3	Sensor
	4	Load shed device
	5	Cathodic protection unit
	6..7	Unassigned

G_MECH_FORM		Unsigned binary number identifying the primary mechanical design principle of the device depending upon value of **G_ED_TYPE**.
		For **G_ED_TYPE** = 1.
	0	Unclassified
	1	Bellows meter
	2	Rotary
	3	Turbine meter
	4	Fluidic oscillator
	5	Anemometer
	6..7	Unassigned
		For **G_ED_TYPE** = 2.
	0	Unclassified
	1	Diaphragm
	2..7	Unassigned
		For **G_ED_TYPE** = 3.
	0	Unclassified
	1	Physical parameter to voltage
	2	Physical parameter to current
	3	Physical parameter to frequency
	4	Open/closed switch
	5..7	Unassigned
		For **G_ED_TYPE** = 5.
	0	Unclassified
	1	Pipe-to-soil
	2	Impedance
	3..7	Unassigned
G_ENG_METRIC	**0**	English
	1	Metric
G_PRESSURE_RCD		
G_MAX_PRESS		End device maximum design pressure.
G_UOM_PRESS		See field **ID_CODE** in **UOM_ENTRY_TBL** (Table 12).
G_FLOW_RCD		
G_MAX_FLOW		End device maximum design flow rate.
G_UOM_FLOW		See field **ID_CODE** in **UOM_ENTRY_TBL** (Table 12).
G_SIZE_BFLD		
G_GEAR_DRIVE		Code representing gas meter gear drive size.
	0	None
	1	1/2 ft

	2	1 ft
	3	2 ft
	4	5 ft
	5	10 ft
	6	100 ft
	7	1000 ft

G_INPUT_OUTPUT_PIPE — Code representing gas meter pipe size.

	English	Metric
0	None	None
1	1/2 in	
2	5/8 in	
3	3/4 in	
4	1 in	
5	1 1/2 in	
6	2 in	
7	4 in	
8	6 in	
9	8 in	
10	10 in	
11	12 in	
12	14 in	
13	16 in	
14	18 in	
15	20 in	
16	22 in	
17	24 in	
18	26 in	
19	28 in	
20	30 in	
21	32 in	
22	34 in	
23	36 in	
24..31	Unassigned	

G_COMPENSATION_BFLD
G_COMP_TEMP — Code representing the gas meter temperature compensation type.

0	Uncompensated
1	Mechanical
2	Sensor
3..7	Unassigned

G_COMP_PRESS — Code representing the gas meter pressure compensation type.

0	Uncompensated
1	Mechanical
2	Sensor
3..7	Unassigned

The following structures are generally associated with electric utility end devices. The values of the data items shall be set up by the manufacturer and left unchanged since they are determined by the functionality of the device.

Identifier	*Value*	*Definition*
E_ELEMENTS_BFLD		Contains binary codes physically describing the end device.
E_FREQ		Power frequency rating code. The codes are:
	0	DC
	1	25 Hz
	2	50
	3	50 or 60 Hz
	4	60 Hz
	5	400 Hz
	6	Unassigned
	7	Unclassified
E_NO_OF_ELEMENTS		Number of commodity measuring elements per measuring input to the end device. This code indicates the number of elements as follows:
	0	None
	1	1 element
	2	2 elements
	3	2.5 elements
	4	3 elements
	5	6 elements
	6	1.5 elements
	7	Unassigned
E_BASE_TYPE		Indicates the type of meter base as follows:
	0	None
	1	S-base (socket)
	2	A-base (ANSI bottom-connected)
	3	K-base
	4	IEC bottom-connected
	5	switchboard
	6	rackmount
	7	B-base
	8	P-base (Canadian Standard)
	9–15	Unassigned
E_ACCURACY_CLASS		Reserved for future ANSI solid-state meter standard accuracy class definitions.
E_VOLTS_BFLD		Contains binary codes describing meter RMS voltages.
E_ELEMENTS_VOLTS		Meter element voltage code. This binary code indicates the meter voltage class as follows:
	0	None
	1	69.3
	2	72
	3	120
	4	208
	5	240

	6	277
	7	480
	8	120–277
	9	120–480
	10–15	Unassigned

E_ED_SUPPLY_VOLTS — External supply voltage code. This value identifies the meter supply voltage as follows:

0	Internal
1	69.3 ac
2	72 ac
3	120 ac
4	208 ac
5	240 ac
6	277 ac
7	480 ac
8	120–277 ac
9	120–480 ac
10	48 dc
11	125 dc
12	250 dc
13–15	Unassigned

E_AMPS_RCD

E_CLASS_MAX_AMPS — End device class or IEC max amp rating.

E_TA — The RMS amperage test amps (TA) specified by the manufacturer for the main test and/or adjustment of the meter.

E_ELECTRIC_DEVICE_RCD

E_kh — Watt-hours per revolution. (May be represented as unit-hours per equivalent revolution).

E_kt — The commodity amount selected for the test pulse output.

E_INPUT_SCALAR — Divisor by which to scale input values. For example if input consists of pulses, an **INPUT_SCALAR** value of 2 would cause the pulse stream to be divided by 2.

E_ED_CONFIG — Form number per ANSI C12.10.

DEVICE_DEFINITION_RCD — This is full device nameplate data structure configured as a case statement for selection of water, gas or electric end devices. The selection variable is **GEN_CONFIG_TBL.NAMEPLATE_TYPE** (Table 00).

9.1.4 TABLE 03 ED_MODE Status table

Table 03 Data description

ED_MODE_STATUS_TBL (Table 03) provides the current operating mode and status of various conditions in the end device. It allows for both standard and manufacturer specific status definitions.

```
TYPE ED_MODE_BFLD = BIT FIELD OF UINT8
      METERING_FLAG           : BOOL(0);
      TEST_MODE_FLAG          : BOOL(1);
      METER_SHOP_MODE_FLAG    : BOOL(2);
      FILLER                  : FILL(3..15);
END;

TYPE ED_STD_STATUS1_BFLD = BIT FIELD OF UINT16
      UNPROGRAMMED_FLAG             : BOOL(0);
      CONFIGURATION_ERROR_FLAG      : BOOL(1);
      SELF_CHK_ERROR_FLAG           : BOOL(2);
      RAM_FAILURE_FLAG              : BOOL(3);
      ROM_FAILURE_FLAG              : BOOL(4);
      NONVOL_MEM_FAILURE_FLAG       : BOOL(5);
      CLOCK_ERROR_FLAG              : BOOL(6);
      MEASUREMENT_ERROR_FLAG        : BOOL(7);
      LOW_BATTERY_FLAG              : BOOL(8);
      LOW_LOSS_POTENTIAL_FLAG       : BOOL(9);
      DEMAND_OVERLOAD_FLAG          : BOOL(10);
      POWER_FAILURE_FLAG            : BOOL(11);
      FILLER                        : FILL(12..15);
END;

TYPE ED_STD_STATUS2_BFLD = BIT FIELD OF UINT8
      FILLER            : FILL(0..7);
END;

TYPE ED_MFG_STATUS_RCD = PACKED RECORD
      ED_MFG_STATUS     : SET(GEN_CONFIG_TBL.DIM_MFG_STATUS_USED);
END;

TYPE ED_MODE_STATUS_RCD = PACKED RECORD
      ED_MODE           : ED_MODE_BFLD;
      ED_STD_STATUS1    : ED_STD_STATUS1_BFLD;
      ED_STD_STATUS2    : ED_STD_STATUS2_BFLD;
      ED_MFG_STATUS     : ED_MFG_STATUS_RCD;
END;

TABLE ED_MODE_STATUS_TBL = ED_MODE_STATUS_RCD;
```

Identifier	*Value*	*Definition*
TYPE ED_MODE_BFLD		
METERING_FLAG	FALSE	End device is not in metering mode. It may or may not be accumulating measured or input quantities.
	TRUE	End device is in metering mode and is accumulating measured or input quantities.
TEST_MODE_FLAG	FALSE	End device is not in test mode.
	TRUE	End device is in test mode.

METER_SHOP_MODE_FLAG	FALSE	End device is not in meter shop mode.
	TRUE	End device is in meter shop mode.

TYPE ED_STD_STATUS1_BFLD

UNPROGRAMMED_FLAG	FALSE	End device is programmed.
	TRUE	End device is not programmed or it is in a factory default state.
CONFIGURATION_ERROR_FLAG	FALSE	End device did not detect a configuration error.
	TRUE	End device did detect a configuration error.
SELF_CHK_ERROR_FLAG	FALSE	End device did not detect a self check error.
	TRUE	End device did detect a self check error.
RAM_FAILURE_FLAG	FALSE	End device did not detect a RAM memory failure.
	TRUE	End device did detect a RAM memory failure.
ROM_FAILURE_FLAG	FALSE	End device did not detect a ROM memory failure.
	TRUE	End device did detect a ROM memory failure.
NONVOL_MEM_FAILURE_FLAG	FALSE	End device did not detect a nonvolatile memory failure.
	TRUE	End device did detect a nonvolatile memory failure.
CLOCK_ERROR_FLAG	FALSE	End device did not detect a clock error.
	TRUE	End device did detect a clock error.
MEASUREMENT_ERROR_FLAG	FALSE	End device did not detect a measurement element error.
	TRUE	End device did detect a measurement element error.
LOW_BATTERY_FLAG	FALSE	End device did not detect a low-battery error.
	TRUE	End device did detect a low-battery error.

LOW_LOSS_POTENTIAL_FLAG

	FALSE	End device did not detect a potential that is below a predetermined value.
	TRUE	End device did detect a device potential that is below a predetermined value.

DEMAND_OVERLOAD_FLAG	FALSE	End device did not detect a demand threshold overload.
	TRUE	End device did detect a demand threshold overload.
POWER_FAILURE_FLAG	FALSE	End device did not detect a power failure.
	TRUE	End device did detect a power failure.
ED_STD_STATUS2_BFLD		Standard status code bit field 2 is a place holder for future expansion.
ED_MFG_STATUS_RCD		Set containing the manufacturer specific status flags.

```
ED_MODE_STATUS_RCD
      ED_MODE                                   See ED_MODE_BFLD.
      ED_STD_STATUS1                            See ED_STD_STATUS1_BFLD.
      ED_STD_STATUS2                            See ED_STD_STATUS2_BFLD.
      ED_MFG_STATUS                             See ED_MFG_STATUS_RCD.

ED_MODE
```

9.1.5 TABLE 04 Pending status table

Table 04 Data description

PENDING_STATUS_TBL (Table 04) is established to provide an indication of the tables present in the end device with a pending status. Information contained within this table includes table number, manufacturer or standard table indicator, conditions (events) upon which activation of a pending table is to occur, self-read or demand reset upon activation indicators, and several additional status variables.

```
TYPE STATUS_BFLD = BIT FIELD OF UINT8
      EVENT_CODE               : UINT(0..3);
      SELF_READ_FLAG           : BOOL(4);
      DEMAND_RESET_FLAG        : BOOL(5);
      RESERVED                 : FILL(6..7);
END;

TYPE EVENT_RCD = PACKED RECORD
      EVENT_SELECTOR           : STATUS_BFLD;
      EVENT_STORAGE            : ARRAY[5] OF UINT8;
END;

TYPE ENTRY_ACTIVATION_RCD = PACKED RECORD
      EVENT                    : EVENT_RCD;
      TABLE_SELECTOR           : TABLE_IDA_BFLD;
END;

TYPE PENDING_ACTIVATION_RCD = PACKED RECORD
      STANDARD_PENDING           : SET(GEN_CONFIG_TBL.DIM_STD_TBLS_USED);
      MANUFACT_PENDING           : SET(GEN_CONFIG_TBL.DIM_MFG_TBLS_USED);
      LAST_ACTIVATION_DATE_TIME  : STIME_DATE;
      NBR_PENDING_ACTIVATION     : UINT8;
      PENDING_TABLES             : ARRAY[GEN_CONFIG_TBL.NBR_PENDING] OF
                                      ENTRY_ACTIVATION_RCD;
END;

TABLE PENDING_STATUS_TBL = PENDING_ACTIVATION_RCD;
```

Identifier	*Value*	*Definition*
STATUS_BFLD		Selectors indicating action to be performed upon activation of selected table.
EVENT_CODE		Variable that specifies the condition upon which the pending table shall be made active. The trigger condition is stored in the **EVENT_STORAGE** variable.
	0	The pending table is activated based on an absolute time contained in the **EVENT_STORAGE** variable.

	1	The pending table is activated based on a relative time contained in the **EVENT_STORAGE** variable.
	2	The pending table is activated based on a non-time associated trigger contained in the **EVENT_STORAGE** variable.
	3..15	Reserved.
SELF_READ_FLAG	FALSE	Do not perform a self read before pending table is activated.
	TRUE	Perform a self read, if capable, before pending table is activated.
DEMAND_RESET_FLAG	FALSE	Do not perform a demand reset before pending table is activated.
	TRUE	Perform a demand reset, if capable, before pending table is activated.
RESERVED		Reserved for future use.

ENTRY_ACTIVATION_RCD

EVENTS_SELECTOR	Status bits indicating various conditions associated with a pending table.
EVENT_STORAGE	Variable for storage of the trigger condition that causes a pending table to become active.
STIME_DATE	For an **EVENT_CODE** of 0, this variable contains the year, month, day, hour, and minute in the format indicated.
STIME_DATE	For an **EVENT_CODE** of 1, this variable contains the weeks, days, hours, minutes, and seconds in the format indicated.
CHAR	For an **EVENT_CODE OF 2** this variable contains in the first four characters the ASCII manufacturer code per Annex A of this standard. The final character is an event number as defined by the manufacturer.
TABLE_SELECTOR	Variable containing the table number, the standard or manufacturer flag and the pending flag. The pending flag indicates that the associated table has not been activated.

PENDING_ACTIVATION_RCD

STANDARD_PENDING	This set variable indicates which of the standard tables are capable of being written with a pending status.
MANUFACT_PENDING	This set variable indicates which of the manufacturer tables are capable of being written with a pending status.
LAST_ACTIVATION_DATE_TIME	Date and time of the last pending table activated.

NBR_PENDING_ACTIVATION	Number of activation events in the **PENDING_STATUS_TBL** structure that have yet to be activated.
PENDING_TABLES	List of pending tables and associated activation triggers.

9.1.6 TABLE 05 Device identification table

Table 05 Data description

DEVICE_IDENT_TBL (Table 05) This table provides the unique identifier for the device as specified by the user.

```
TYPE IDENT_RCD = PACKED RECORD
      IF GEN_CONFIG_TBL.ID_FORM THEN
            IDENTIFICATION    : ARRAY[10] OF BCD;
      ELSE
            IDENTIFICATION: ARRAY[20] OF CHAR;
      END;
END;

TABLE DEVICE_IDENT_TBL = IDENT_RCD;
```

Identifier	*Value*	*Definition*
IDENT_RCD		
IDENTIFICATION		Array of 20 characters, or 20 **BCD** digits used to uniquely identify the device.

9.1.7 TABLE 06 Utility information table

Table 06 Data description

UTIL_INFO_TBL (Table 06) is for utility use in providing additional end device identifying or tracking information.

```
TYPE UTIL_INFO_RCD = PACKED RECORD
      OWNER_NAME                    : ARRAY[20] OF CHAR;
      UTILITY_DIV                   : ARRAY[20] OF CHAR;
      IF GEN_CONFIG_TBL.ID_FORM THEN
            SERVICE_POINT_ID        : ARRAY[10] OF BCD;
            ELEC_ADDR               : ARRAY[10] OF BCD;
            DEVICE_ID               : ARRAY[10] OF BCD;
            UTIL_SER_NO             : ARRAY[10] OF BCD;
            CUSTOMER_ID             : ARRAY[10] OF BCD;
      ELSE
            SERVICE_POINT_ID        : ARRAY[20] OF CHAR;
            ELEC_ADDR               : ARRAY[20] OF CHAR;
            DEVICE_ID               : ARRAY[20] OF CHAR;
            UTIL_SER_NO             : ARRAY[20] OF CHAR;
            CUSTOMER_ID             : ARRAY[20] OF CHAR;
      END;
```

```
      COORDINATE_1                  : ARRAY[10] OF UINT8;
      COORDINATE_2                  : ARRAY[10] OF UINT8;
      COORDINATE_3                  : ARRAY[10] OF UINT8;
      TARIFF_ID                     : ARRAY[8] OF CHAR;
      EX1_SW_VENDOR                 : ARRAY[4] OF CHAR;
      EX1_SW_VERSION_NUMBER         : UINT8;
      EX1_SW_REVISION_NUMBER        : UINT8;
      EX2_SW_VENDOR                 : ARRAY[4] OF CHAR;
      EX2_SW_VERSION_NUMBER         : UINT8;
      EX2_SW_REVISION_NUMBER        : UINT8;
      PROGRAMMER_NAME               : ARRAY[10] OF CHAR;
      MISC_ID                       : ARRAY[30] OF CHAR;
END;

TABLE UTIL_INFO_TBL = UTIL_INFO_RCD;
```

Identifier	*Value*	*Definition*
UTIL_INFO_RCD		
OWNER_NAME		Up to 20 characters. All printable characters are valid.
UTILITY_DIV		Utility division number. Up to 16 characters. All printable characters are valid.
SERVICE_POINT_ID		20 characters or **BCD** digits. **ID** number attached to the service point.
ELEC_ADDR		End device logical electrical address for mapping and study purposes. This is used by the utility as information only.
DEVICE_ID		The **ID** attached to the hardware. 20 characters or **BCD** digits.
UTIL_SER_NO		Utility specified serial number. Up to 20 characters or **BCD** digits.
CUSTOMER_ID		Up to 20 characters or **BCD** digits. All printable characters are valid if characters.
COORDINATE_1		Generalized mapping coordinate *x*.
COORDINATE_2		Generalized mapping coordinate *y*.
COORDINATE_3		Generalized mapping coordinate *z*.
TARIFF_ID		Identification of the billing tariff.

EX1_SW_VENDOR		Name of manufacturer, in ASCII, that provided configuration/programming software. Refer to Annex A for list of manufacturer identifiers.
EX1_SW_VERSION_NUMBER	0..255	Configuration/programming software version number, unsigned binary number. Changes in value imply functional differences.
EX1_SW_REVISION_NUMBER	0..255	Configuration/programming software revision number, unsigned binary number. Changes in value imply product corrections or improvements. Recommend use of "0" to designate preproduction or prototyping software.
EX2_SW_VENDOR		Name of manufacturer, in ASCII, that provided configuration/programming software. Refer to Annex A for list of manufacturer identifiers.
EX2_SW_VERSION_NUMBER	0..255	Configuration/programming software version number. Changes in value imply functional differences.
EX2_SW_REVISION_NUMBER	0..255	Configuration/programming software revision number, unsigned binary number. Changes in value imply product corrections or improvements. Recommend use of "0" to designate preproduction or prototyping software.
PROGRAMMER_NAME		Name of the last programmer or programming device.
MISC_ID		Thirty characters, free form. All printable characters are allowed. This could be used to contain approval, verification., bar code, etc.

9.1.8 TABLE 07 Procedure initiate table

Table 07 Data description

PROC_INITIATE_TBL (Table 07) allows for the execution of commands using the table structure. To execute a command a procedure identifier (**PROC**), a sequence number (**SEQ_NBR**), and an optional parameter list (**PARM**) are written into this table. The response to the command is placed in **PROC_RESPONSE_TBL** (Table 08) and is available to be read. As a minimum response, all commands produce an echo of the procedure identifier (**PROC**), an echo of the sequence number (**SEQ_NBR**), and a result code.

```
TYPE  PROC_FORMAT_RCD = PACKED RECORD
      PROC         : TABLE_IDB_BFLD;
      SEQ_NBR      : UINT8;
      PARM         : PARM_RCD;
END;

TABLE PROC_INITIATE_TBL = PROC_FORMAT_RCD;
```

Identifier	*Value*	*Definition*
TABLE_IDB_BFLD		
TBL_PROC_NBR	0..2047	Procedure number to be executed.
STD_VS_MFG_FLAG	FALSE	Procedure selected by **TBL_PROC_NBR** is a standard defined procedure.
	TRUE	Procedure selected by **TBL_PROC_NBR** is a manufacturer defined procedure.
SELECTOR		Describes how the response to the procedure is handled.
	0	Post response in **PROC_RESPONSE_TBL** (Table 08) on completion.
	1	Post response in **PROC_RESPONSE_TBL** (Table 08) on exception.
	2	Do not post response in **PROC_RESPONSE_TBL** (Table 08).
	3	Post response in **PROC_RESPONSE_TBL** (Table 08) immediately and post another response in **PROC_RESPONSE_TBL** (Table 08) on completion.
	4..15	Reserved.
PROC_FORMAT_RCD		
PROC		Bit field indicating procedure number to be executed and whether the procedure is a standard defined procedure or a manufacturer defined procedure. It also describes the response method.
SEQ_NBR	0..255	The sequence number is supplied by initiator of the procedure. It shalll be returned with procedure response in **PROC_RESPONSE_TBL** (Table 08) as a means of coordination.
PARM		Argument for procedure as defined for individual procedures in 9.1.8.1 and subsections.

9.1.8.1 Standard procedures

The following procedures are defined as standard procedures by this standard. **STD_VS_MFG_FLAG** shall be set to FALSE.

9.1.8.1.1 Cold start

This procedure causes the end device to return to a manufacturer default known state. All data values, programs and conditions may be lost. Communication may be broken. If an event log exists, an attempt shall be made to retain it.

TBL_PROC_NBR	00	Procedure number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.8.1.2 Warm start

This procedure causes the end device to return to a power-up state. Communication may be broken.

TBL_PROC_NBR	01	Procedure number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.8.1.3 Save configuration

This procedure causes the end device to save its active configuration.

TBL_PROC_NBR	02	Procedure number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.8.1.4 Clear data

This procedures causes the end device to purge generated data fields but retain programming fields. Data fields to be purged are specified by the device manufacturer.

TBL_PROC_NBR	03	Procedure number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.8.1.5 Reset list pointers

When invoked, the end device attempts to reset list control variables to their initial state. To execute this procedure, the initiator shall in addition be required to have access permission to the procedure and write access permission to the table containing the selected list(s).

TBL_PROC_NBR	04	Procedure number
PARM_RCD		Defined below
RESP_DATA_RCD		Not used

Parameters are as follows:

```
TYPE PARM_RCD = PACKED RECORD
      LIST: UINT8;
END;
```

Identifier	*Value*	*Definition*
PARM_RCD		
LIST	0	Reserved
	1	**EVENT_LOG_TBL** (Table 76)
	2	**SELF_READ_DATA_TBL** (Table 26)
	3	**LP_DATA_SET1_TBL** (Table 64)
	4	**LP_DATA_SET2_TBL** (Table 65)
	5	**LP_DATA_SET3_TBL** (Table 66)
	6	**LP_DATA_SET4_TBL** (Table 67)
	7	**LP_DATA_SET1_TBL** (Table 64) and **LP_DATA_SET2_TBL** (Table 65) and **LP_DATA_SET3_TBL** (Table 66) and **LP_DATA_SET4_TBL** (Table 67)
	8	**HISTORY_LOG_TBL** (Table 74)
	9..254	Reserved
	255	All lists except **EVENT_LOG_TBL** (Table 76)

9.1.8.1.6 Update last read entry

When invoked, the end device attempts to reduce the list variable **NBR_UNREAD_ENTRIES** by the value specified. To execute this procedure, the initiator shall in addition be required to have access to the procedure and write access to the table containing the selected list(s).

TBL_PROC_NBR	05	Procedure Number
PARM_RCD		Defined below
RESP_DATA_RCD		Not used

Parameters are as follows:

```
TYPE PARM_RCD = PACKED RECORD
      LIST: UINT8;
      ENTRIES_READ : UINT16;
END;
```

Identifier	*Value*	*Definition*
PARM_RCD		
LIST	0	Reserved
	1	**EVENT_LOG_TBL** (Table 76)
	2	**SELF_READ_DATA_TBL** (Table 26)
	3	**LP_DATA_SET1_TBL** (Table 64)
	4	**LP_DATA_SET2_TBL** (Table 65)
	5	**LP_DATA_SET3_TBL** (Table 66)
	6	**LP_DATA_SET4_TBL** (Table 67)
	7	**LP_DATA_SET1_TBL** (Table 64) and **LP_DATA_SET2_TBL** (Table 65) and **LP_DATA_SET3_TBL** (Table 66) and **LP_DATA_SET4_TBL** (Table 67)
	8	**HISTORY_LOG_TBL** (Table 74)
	9..254	Reserved
	255	All lists except **EVENT_LOG_TBL** (Table 76)
ENTRIES_READ	0..65535	Number of entries confirmed.

9.1.8.1.7 Change end device mode

This procedure changes the operational modes in the end device to the argument described below.

TBL_PROC_NBR	06	Procedure Number
PARM_RCD		Defined below
RESP_DATA_RCD		Defined below

Parameters are as follows:

```
TYPE PARM_RCD = PACKED RECORD
      ED_MODE     : ED_MODE_STATUS_TBL.ED_MODE_BFLD;
END;
```

Identifier	*Value*	*Definition*
PARM_RCD		
ED_MODE		See **ED_MODE_STATUS_TBL.ED_MODE** (Table 03). This is the desired mode.

Response is as follows:

```
TYPE RESP_DATA_RCD = PACKED RECORD
      ED_MODE     : ED_MODE_STATUS_TBL.ED_MODE_BFLD;
END;
```

Identifier	*Value*	*Definition*
RESP_DATA_RCD		
ED_MODE		See **ED_MODE_STATUS_TBL.ED_MODE** (Table 03). The mode the device is in after the execution of this procedure.

9.1.8.1.8 Clear standard status flags

When invoked, the end device attempts to clear all standard status flags.

TBL_PROC_NBR	07	Procedure Number
PARM_RCD		Not used
RESP_DATA_RCD		Defined below

Response is as follows:

```
TYPE RESP_DATA_RCD = PACKED RECORD
      ED_STD_STATUS_1   : ED_MODE_STATUS_TBL.ED_STD_STATUS1_BFLD;
      ED_STD_STATUS_2   : ED_MODE_STATUS_TBL.ED_STD_STATUS2_BFLD;
END;
```

Identifier	*Value*	*Definition*
RESP_DATA_RCD		
ED_STD_STATUS_1		See **ED_MODE_STATUS_TBL.ED_STD_STATUS_1** (Table 03).
ED_STD_STATUS_2		See **ED_MODE_STATUS_TBL.ED_STD_STATUS_2** (Table 03).

9.1.8.1.9 Clear manufacturer status flags

When invoked, the end device attempts to clear all manufacturer status flags.

TBL_PROC_NBR	08	Procedure Number
PARM_RCD		Defined below
RESP_DATA_RCD		Not used

Response is as follows:

```
TYPE PARM_RCD = PACKED RECORD
      ED_MFG_STATUS: ED_MODE_STATUS_TBL.ED_MFG_STATUS_RCD;
END;
```

Identifier	*Value*	*Definition*
PARM_RCD		
ED_MFG_STATUS		See **ED_MODE_STATUS_TBL.ED_MFG_STATUS** (Table 03).

9.1.8.1.10 Remote reset

When invoked, this procedure attempts to perform the specified combination of resets and change of season. The following resets are supported: self read, demand reset, and season change.

TBL_PROC_NBR	09	Procedure Number
PARM_RCD		Defined below
RESP_DATA_RCD		Defined below

Parameters are as follows:

```
TYPE ACTION_FLAG_BFLD = BIT FIELD OF UINT8
      DEMAND_RESET_FLAG       : BOOL(0);
      SELF_READ_FLAG          : BOOL(1);
      SEASON_CHANGE_FLAG      : BOOL(2);
      NEW_SEASON              : UINT(3..6);
      FILLER                  : FILL(7..7);
END;

TYPE PARM_RCD = PACKED RECORD
      ACTION_FLAG : ACTION_FLAG_BFLD;
END;
```

Identifier	*Value*	*Definition*
ACTION_FLAG_BFLD		
DEMAND_RESET_FLAG	FALSE	Do not perform a demand reset.
	TRUE	Perform a demand reset.
SELF_READ_FLAG	FALSE	Do not perform a self read.
	TRUE	Perform a self read.

SEASON_CHANGE_FLAG	FALSE	Do not perform a change to the new season specified
	TRUE	Perform a change to the new season specified.
NEW_SEASON	0..15	Season to change to if **SEASON_CHANGE** bit is set.

PARM_RCD
ACTION_FLAG — Bit mask denotes the reset actions to perform**.**

Response is as follows:

```
TYPE RESP_DATA_RCD = PACKED RECORD
      SUCCESS_FLAG : ACTION_FLAG_BFLD;
END;
```

Identifier	*Value*	*Definition*
ACTION_FLAG_BFLD		
DEMAND_RESET_FLAG	FALSE	Did not perform a demand reset.
	TRUE	Performed a demand reset.
SELF_READ_FLAG	FALSE	Did not perform a self read.
	TRUE	Performed a self read.
SEASON_CHANGE_FLAG	FALSE	Did not perform a change to the new season specified
	TRUE	Performed a change to the new season specified.
NEW_SEASON	0..15	The season number the end device changed **SEASON_CHANGE** bit was set.
RESP_DATA_RCD		
SUCCESS_FLAG		Indicates which “resets” were successful by setting corresponding bits.

9.1.8.1.11 Set date and/or time

This procedure attempts to set the date and/or time in the end device.

TBL_PROC_NBR	**10**	Procedure number.
PARM_RCD		Defined below
RESP_DATA_RCD		Defined below

Parameters are as follows:

```
TYPE SET_MASK_BFLD = BIT FIELD OF UINT8
      SET_TIME_FLAG           : BOOL(0);
      SET_DATE_FLAG           : BOOL(1);
      SET_TIME_DATE_QUAL      : BOOL(2);
      DST_FLAG                : BOOL(2);
```

```
      GMT_FLAG                : BOOL(3);
      TM_ZN_APPLIED_FLAG      : BOOL(4);
      DST_APPLIED_FLAG        : BOOL(5);
      FILLER                  : FILL(36..7);
END;

TYPE PARM_RCD = PACKED RECORD
      SET_MASK          : SET_MASK_BFLD;
      DATE_TIME         : LTIME_DATE;
      TIME_DATE_QUAL    : CLOCK_TBL.TIME_DATE_QUAL_BFLD;
END;
```

Identifier	*Value*	*Definition*
SET_MASK_BFLD		
SET_TIME_FLAG	FALSE	Do not set time.
	TRUE	Set time.
SET_DATE_FLAG	FALSE	Do not set date.
	TRUE	Set date.
DST_FLAG	FALSE	Time is not Daylight Savings Time (DST).
	TRUE	Time is DST.
GMT_FLAG	FALSE	Time not relative to Greenwich Mean Time (GMT).
	TRUE	Time relative to GMT.
TM_ZN_APPLIED_FLAG	FALSE	The time zone offset has not been applied to the time.
	TRUE	The time zone offset has been applied to the time.
DST_APPLIED_FLAG	FALSE	DST has not been applied to the time.
	TRUE	DST has been applied to the time.
SET_TIME_DATE_QUAL	FALSE	Do not set time date qualifier.
	TRUE	Set time date qualifier.
PARM_RCD		
SET_MASK		Bit mask denotes the actions to perform.
DATE_TIME		New date and time.
TIME_DATE_QUAL_BFLD		Status qualifying the end device time.
DAY_OF_WEEK		Current day of the week.
	0	Sunday
	1	Monday
	2	Tuesday
	3	Wednesday
	4	Thursday
	5	Friday
	6	Saturday
	7	Unassigned

DST_FLAG		DST status.
	FALSE	End device time is not in DST
	TRUE	End device time is in DST.
GMT_FLAG	FALSE	End device time does not correspond to GMT
	TRUE	End device time corresponds to GMT
TM_ZN_APPLIED_FLAG	FALSE	Time zone offset has not been applied to the end device time.
	TRUE	Time zone offset has been applied to the end device time.
DST_APPLIED_FLAG	FALSE	End device time does not include daylight savings adjustment.
	TRUE	End device time includes daylight savings adjustment.

Response is as follows:

```
TYPE RESP_DATA_RCD = PACKED RECORD
      DEV_DATE_TIME_BEFORE    : LTIME_DATE;
      DEV_DATE_TIME_AFTER     : LTIME_DATE;
END;
```

Identifier	*Value*	*Definition*
RESP_DATA_RCD		
DEV_DATE_TIME_BEFORE		Date and time in the end device just before set date time procedure was executed.
DEV_DATE_TIME_AFTER		Date and time in the end device just after set date time procedure was executed.

9.1.8.1.12 Execute diagnostics procedure

This procedure initiates an end device diagnostic procedure.

TBL_PROC_NBR	11	Procedure Number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.8.1.13 Activate all pending tables

This procedure causes the end device to immediately activate all pending tables regardless of activation trigger(s).

TBL_PROC_NBR	12	Procedure Number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.8.1.14 Activate specific pending table(s)

This procedure causes the end device to immediately activate any pending table(s) with the specified activation trigger.

TBL_PROC_NBR	13	Procedure Number
PARM_RCD		Defined below
RESP_DATA_RCD		Not used

Parameters are as follows:

```
TYPE PARM_RCD = PACKED RECORD
      EVENT              : PENDING_STATUS_TBL.EVENT_RCD;
END;
```

Identifier	*Value*	*Definition*
PARM_RCD		
EVENT		See **EVENT_RCD** (Table 04).

9.1.8.1.15 Clear all pending tables

This procedure causes the end device to immediately clear the pending state all pending tables.

TBL_PROC_NBR	14	Procedure Number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.8.1.16 Clear specific pending table(s)

This procedure causes the end device to immediately clear the pending status of all pending table(s) with the specified activation trigger.

TBL_PROC_NBR	15	Procedure Number
PARM_RCD		Defined below
RESP_DATA_RCD		Not used

Parameters are as follows:

```
TYPE PARM_RCD = PACKED RECORD
      EVENT       : PENDING_STATUS_TBL.EVENT_RCD;
END;
```

Identifier	Value	Definition
PARM_RCD		
EVENT		See **EVENT_RCD** (Table 04).

9.1.8.1.17 Start load profile

This procedure starts all defined load profile sets.

TBL_PROC_NBR	16	Procedure Number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.8.1.18 Stop load profile

This procedure stops all active load profile sets.

TBL_PROC_NBR	17	Procedure Number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.8.1.19 Log in

When invoked, this procedure establishes the active **USER_ID** that is used in the **HISTORY_LOG_TBL** (Table 74) and in the **EVENT_LOG_TBL** (Table 76) and supplies a password.

TBL_PROC_NBR	18	Procedure Number
PARM_RCD		Defined below
RESP_DATA_RCD		Not used

Parameters are as follows:

```
TYPE PARM_RCD = PACKED RECORD
      USER_ID     : UINT16;
      PASSWORD    : ARRAY[20] OF UINT8;
END;
```

Identifier	*Value*	*Definition*
PARM_RCD		
USER_ID	0..65535	**ID** associated with current user. Note: **USER_ID** of zero and **USER_ID** of one should be avoided since it represents an end device initiated event and manually initiated event respectively in the loggers.
PASSWORD		Password supplied. The first character of the password shall be placed in the zero element of the array.

9.1.8.1.20 Log out

Deactivates the **USER_ID** and password.

TBL_PROC_NBR	19	Procedure Number
PARM_RCD		Not used
RESP_DATA_RCD		Not used

9.1.9 TABLE 08 Procedure response table

Table 08 Data description

PROC_RESPONSE_TBL (Table 08) is a complementary table to **PROC_INITIATE_TBL** (Table 07). This table contains a response, which gets posted to Table 08 as a result of a write to **PROC_INITIATE_TBL** (Table 07).

```
TYPE  PROC_RESPONSE_RCD = PACKED RECORD
      PROC              : TABLE_IDB_BFLD;
      SEQ_NBR           : UINT8;
      RESULT_CODE       : UINT8;
      RESP_DATA         : RESP_DATA_RCD;
END;

TABLE PROC_RESPONSE_TBL = PROC_RESPONSE_RCD;
```

Identifier	*Value*	*Definition*
TABLE_IDB_BFLD		
TBL_PROC_NBR	0..2047	Procedure number executed by end device.
STD_VS_MFG_FLAG	FALSE	Procedure was a standard procedure.
	TRUE	Procedure was a manufacturer defined procedure.
SELECTOR		Not used in **PROC_RESPONSE_TBL** (Table 08). See **PROC_INITIATE_TBL** (Table 07).
PROC_RESPONSE_RCD		
PROC		Bit field indicating procedure number executed and whether the procedure was a standard defined or a manufacturer defined.
SEQ_NBR	0..255	Supplied by initiator of the procedure **PROC_INITIATE_TBL** (Table 07). Returned with response as a means of coordination.
RESULT_CODE		Code that identifies the status of the procedure execution. The codes are defined as follows:
	0	Procedure completed.
	1	Procedure accepted but not fully completed.
	2	Invalid parameter for known procedure, procedure was ignored.
	3	Procedure conflicts with current device setup, procedure was ignored.
	4	Timing constraint, procedure was ignored.
	5	No authorization for requested procedure, procedure was ignored.
	6	Unrecognized procedure, procedure was ignored.
	7..255	Reserved.
RESP_DATA_RCD		Procedure response record as defined for individual procedures in **PROC_INITIATE_TBL** (Table 07). **RESP_DATA_RCD** is posted if procedure was successfully completed (**RESULT_CODE** = 0) and **SELECTOR** in **PROC_INITIATE_TBL** (Table 07) is not set to a value of 2.

9.2 DECADE 10: Data source tables

This decade contains tables associated with data sources.

9.2.1 TABLE 10 Dimension sources limiting table

Table 10 Data description

DIM_SOURCES_LIM_TBL (Table 10) contains maximum dimensions and end device capabilities for establishing data sources.

```
TYPE SOURCE_FLAGS_BFLD = BIT FIELD OF UINT8
      PF_EXCLUDE_FLAG          : BOOL(0);
      RESET_EXCLUDE_FLAG       : BOOL(1);
      BLOCK_DEMAND_FLAG        : BOOL(2);
      SLIDING_DEMAND_FLAG      : BOOL(3);
      THERMAL_DEMAND_FLAG      : BOOL(4);
      SET1_PRESENT_FLAG        : BOOL(5);
      SET2_PRESENT_FLAG        : BOOL(6);
      FILLER                   : FILL(7..7);
END;

TYPE SOURCE_RCD = PACKED RECORD
      SOURCE_FLAGS             : SOURCE_FLAGS_BFLD;
      NBR_UOM_ENTRIES          : UINT8;
      NBR_DEMAND_CTRL_ENTRIES  : UINT8;
      DATA_CTRL_LENGTH         : UINT8;
      NBR_DATA_CTRL_ENTRIES    : UINT8;
      NBR_CONSTANTS_ENTRIES    : UINT8;
      CONSTANTS_SELECTOR       : UINT8;
      NBR_SOURCES              : UINT8;
END;

TABLE DIM_SOURCES_LIM_TBL = SOURCE_RCD;
```

Identifier	*Value*	*Definition*
SOURCE_FLAGS_BFLD		
		Power fail exclusion is defined as ignoring demand for maximum calculations for some period immediately after a power failure.
PF_EXCLUDE_FLAG	FALSE	End device is not capable of power fail exclusion.
	TRUE	End device is capable of power fail exclusion.
RESET_EXCLUDE_FLAG		Reset exclusion is defined as inhibiting demand reset for some period after a reset.
	FALSE	Reset exclusion is not supported by the end device.
	TRUE	Reset exclusion is supported by the end device.
BLOCK_DEMAND_FLAG	FALSE	Block demand is not supported by the end device.
	TRUE	Block demand is supported by the end device.

SLIDING_DEMAND_FLAG	FALSE	Sliding demand is not supported by the end device.
	TRUE	Sliding demand is supported by the end device.
THERMAL_DEMAND_FLAG	FALSE	Thermal demand is not supported by the end device.
	TRUE	Thermal demand is supported by the end device.
SET1_PRESENT_FLAG	FALSE	The end device does not support the first set of optional constants in the electric record of the **CONSTANTS_TBL** (Table 15).
	TRUE	The end device does support the first set of optional constants in the electric record of the **CONSTANTS_TBL** (Table 15).
SET2_PRESENT_FLAG	FALSE	The end device does not support the second set of optional constants in the electric record of the **CONSTANTS_TBL** (Table 15).
	TRUE	The end device does support the second set of optional constants in the electric record of the **CONSTANTS_TBL** (Table 15).

SOURCE_RCD

SOURCE_FLAGS		Bit fields that indicate what demand and TOU register functions the end device is capable of performing.
NBR_UOM_ENTRIES	0..255	Maximum number of entries in **UOM_ENTRY_TBL** (Table 122).
NBR_DEMAND_CTRL_ENTRIES	0..255	Maximum number of entries in the **DEMAND_CONTROL_TBL** (Table 13).
DATA_CTRL_LENGTH	0..255	Manufacturer supplied value that determines the width in octets of an entry in the first array of the **DATA_CONTROL_TBL** (Table 14).
NBR_DATA_CTRL_ENTRIES	0..255	Maximum number of entries in the **DATA_CONTROL_TBL** (Table 14).
NBR_CONSTANTS_ENTRIES	0..255	Maximum number of entries in the **CONSTANTS_TBL** (Table 15).
CONSTANTS_SELECTOR		This value is the selector for the record structure used in the **CONSTANTS_TBL** (Table 15).
	0	**GAS_CONSTANTS_AGA3_RCD** data structure selected.
	1	**GAS_CONSTANTS_AGA7_RCD** data structure selected.

	2	**ELECTRIC_CONSTANTS_RCD** data structure selected.
	3..255	Reserved for future use.
NBR_SOURCES	0..255	Maximum number of entries in the **SOURCES_TBL** (Table 16).

9.2.2 TABLE 11 Actual sources limiting table

Table 11 Data description

ACT_SOURCES_LIM_TBL (Table 11) contains the actual parameters of Table 10 the end device has been configured with.

TABLE ACT_SOURCES_LIM_TBL = DIM_SOURCES_LIM_TBL.SOURCE_RCD;

Identifier	*Value*	*Definition*
SOURCE_FLAGS_BFLD		
PF_EXCLUDE_FLAG		Power fail exclusion is defined as ignoring demand for maximum calculations for some period immediately after a power failure.
	FALSE	Power fail exclusion is not active.
	TRUE	Power fail exclusion is active.
RESET_EXCLUDE_FLAG		Reset exclusion is defined as inhibiting demand reset for some period after a reset.
	FALSE	Reset exclusion is not in use by the end device.
	TRUE	Reset exclusion is in use by the end device.
BLOCK_DEMAND_FLAG	FALSE	Block demand is not in use by the end device.
	TRUE	Block demand is in use by the end device.
SLIDING_DEMAND_FLAG	FALSE	Sliding demand is not in use by the end device.
	TRUE	Sliding demand is in use by the end device.
THERMAL_DEMAND_FLAG	FALSE	Thermal demand is not in use by the end device.
	TRUE	Thermal demand is in use by the end device.
SET1_PRESENT_FLAG	FALSE	The end device is not using the first set of optional constants in the electric record of the **CONSTANTS_TBL** (Table 15).
	TRUE	The end device is using the first set of optional constants in the electric record of the **CONSTANTS_TBL** (Table 15).
SET2_PRESENT_FLAG	FALSE	The end device is not using the second set of optional constants in the electric record of the **CONSTANTS_TBL** (Table 15).
	TRUE	The end device is using the second set of optional constants in the electric record of the **CONSTANTS_TBL** (Table 15).

SOURCE_RCD		
SOURCE_FLAGS		Bit fields that indicate what demand and TOU register functions are in use by the end device as configured.
NBR_UOM_ENTRIES	0..255	Actual number of entries in **UOM_ENTRY_TBL** Table 12).
NBR_DEMAND_CTRL_ENTRIES	0..255	Actual number of entries in the **DEMAND_CONTROL_TBL** (Table 13).
DATA_CTRL_LENGTH	0..255	Manufacturer defined number of octets in an element of the array of the **DATA_CONTROL_TBL** (Table 14).
NBR_DATA_CTRL_ENTRIES	0..255	Actual number of entries in the **DATA_CONTROL_TBL** (Table 14).
NBR_CONSTANTS_ENTRIES	0..255	Actual number of entries in the **CONSTANTS_TBL** (Table 15).
CONSTANTS_SELECTOR		This value is the selector for the record structure used in the **CONSTANTS_TBL** (Table 15).
	0	**GAS_CONSTANTS_AGA3_RCD** data structure selected.
	1	**GAS_CONSTANTS_AGA7_RCD** data structure selected.
	2	**ELECTRIC_CONSTANTS_RCD** data structure selected.
	3..255	Reserved for future use.
NBR_SOURCES	0..255	Actual number of entries in the **SOURCES_TBL** (Table 16).

9.2.3 TABLE 12 Unit of measure entry table

Table 12 Data description

UOM_ENTRY_TBL (Table 12) provides an efficient method for describing data attributes. It may be used to tag data sources.

```
TYPE UOM_ENTRY_BFLD = BIT FIELD OF UINT32
      ID_CODE                 : UINT(0..7);
      TIME_BASE               : UINT(8..10);
      MULTIPLIER              : UINT(11..13);
      Q1_ACCOUNTABILITY       : BOOL(14);
      Q2_ACCOUNTABILITY       : BOOL(15);
      Q3_ACCOUNTABILITY       : BOOL(16);
      Q4_ACCOUNTABILITY       : BOOL(17);
      NET_FLOW_ACCOUNTABILITY : BOOL(18);
```

```
      SEGMENTATION            : UINT(19..21);
      HARMONIC                : BOOL(22);
      RESERVED                : FILL(23..30);
      NFS                     : BOOL(31);
END;

TYPE UOM_ENTRY_RCD = PACKED RECORD
      UOM_ENTRY   : ARRAY[ACT_SOURCES_LIM_TBL.NBR_UOM_ENTRIES] OF
                        UOM_ENTRY_BFLD;
END;

TABLE UOM_ENTRY_TBL = UOM_ENTRY_RCD;
```

Identifier	*Value*	*Definition*
UOM_ENTRY_RCD		
ID_CODE		The **UOM ID_CODE** identifies the physical quantity of interest (equivalent to physical units in the field of physics as used for dimensional analysis). This standard defines entries for use by water, gas, or electrical utilities and provides for generic entries for description of internal data items. Defined units of measure have been grouped solely for readability. Note that because the unit of measure record includes a time base indicator, some of the units have an extra time factor that is cancelled out by the unit of measure record time base indicator. For example, **ID_CODE** 77, Therm per hr, is actually Therm, once the time base indicator is properly set.
	{Electric industry units}	
		Power:
	0	Active power—W
	1	Reactive power—VAR
	2	Apparent power—VA
	3	Phasor power—VA = sqrt(W^2+VAR^2)
	4	Quantity power—Q(60)
	5	Quantity power—Q(45)
	6	Reserved
	7	Reserved
		Voltage:
	8	RMS volts
	9	Average volts (average of |V|)
	10	RMS volts squared (V^2)
	11	Instantaneous volts
		Current:
	12	RMS amps
	14	RMS amps squared (I^2)
	15	Instantaneous current

	Percent Total Harmonic Distortion (T.H.D):
16	T. H. D. V (IEEE)
17	T. H. D. I (IEEE)
18	T. H. D. V (IC)
19	T. H. D. I (IC)
	Event codes:
50	Power outages
51	Number of demand resets
52	Number of times programmed
53	Number of minutes on battery carryover
	Phase angle:
20	$V–V_A$, voltage phase angle
21	$V_x–V_y$, where *x* and *y* are phases defined in phase selector.
22	$I–V_A$, current phase angle
23	$I_x–I_y$, where *x* and *y* are phases defined in phase selector.
24	Power factor computed using apparent power, **ID_CODE**=2.
25	Power factor computed using phasor power, **ID_CODE**=3.
26	Reserved
27	Reserved
28	Reserved
	Time:
29	Time of day
30	Date
31	Time of day and date
32	Interval timer
33	Frequency
34	Counter
35	Sense input (T/F)
36..39	Reserved
40	Voltage sag
41	Voltage swells
42	Power outage
43	Voltage excursion low
44	Voltage excursion high
45	Normal voltage level
46	Voltage unbalance
47	Voltage T. H. D. excess
48	Current T. H. D. excess
49..63	Reserved

{Gas industry units}

64	Cubic meter gas (volume un-corrected, meter index reading) per hour.
65	Cor cubic meter gas (volume corrected to base conditions) per hour
66	Cor cubic meter gas (volume corrected to pressure base, without supercompressibility per hour.
67	Cubic feet gas (volume corrected, meter index reading) per hour.
68	Cor cubic feet gas(volume corrected to base conditions) per hour.
69	Pcor cubic feet gas(volume corrected to pressure base, without supercompressibility per hour.
70	Dry bulb temp °C
71	Wet bulb temp °C
72	Dry bulb temp °F
73	Wet bulb temp °F
74	Dry bulb temp °K
75	Wet bulb temp °K
76	Joules per hour
77	Therm per hour
78	Static pascal
79	Differential pascal
80	Static pound per square inch
81	Differential pound per square inch
82	Gram cm^2
83	Meter HG column
84	Inch HG column
85	Inch H_2O column
86	Bar
87	Percent relative humidity
88	Parts per million odorant
89..127	Reserved

{Water industry units}

128	Cubic meter liquid per hr
129	Cubic feet liquid per hr
130	US gallons per hr
131	IMP gallons per hr
132	Acre feet per hr
133	Parts per million lead
134	Turbidity
135	Parts per million chlorine
136	PH factor
137	Corrosion
138	Ionization
139	Parts per million `SO_2`
140	Liters
141	Cubic feet liquid

	142	Pounds per square foot differential
	143	Inches of water
	144	Feet of water
	145	Atmospheres
	140..189	Reserved
{Generic industry units}		
	190	Local currency (e.g., dollars)
	191	Inch
	192	Foot
	193	Meter
	194..255	Reserved
TIME_BASE		This field describes the measurement method with respect to time.
	0	Bulk quantity of commodity (dial reading). Quantity of commodity; integral of commodity usage rate. Values have the units stated in **ID_CODE** × hour (energy units).
	1	Instantaneous (sampled). This is the fastest rate at which a measurement is acquired.
	2	Period based. This is a time period based upon the period of a fundamental frequency (power, RMS).
	3	Sub-block average demand. Sub-block average Demand values are the most recent averaging demand subinterval values. The averaging period may be of the order of minutes (demand).
	4	Block average demand. Block average demand values may be either the average over a number of Sub-block average demand subintervals or the average over a single interval. The averaging period is in the order of minutes, and typically greater or equal to the Sub-block average demand period (demand).
	5	Net bulk quantity of commodity (relative dial reading). Quantity of commodity; integral of commodity usage rate over a specified period of time T1 to T2. T1 and T2 are quite arbitrary and are defined by mechanisms and table driven schedules. Values have the units stated in the **ID_CODE** × hour [e.g., W × h = Wh (energy)].
	6	Thermal quantity (demand).
	7	Event quantity (number of occurrences of an identified item).
MULTIPLIER		The multiplier identifies the scaling value to apply to the reported value after delivery of the tagged item; e.g., if the **ID_CODE** = 0 and **MULTIPLIER** = 2 then the **UOM** represents kW (other flags are assumed to have been set accordingly).

0	10^{0}
1	10^{2}
2	10^{3}
3	10^{6}
4	10^{9}
5	10^{-2}
6	10^{-3}
7	10^{-6}

Q1_ACCOUNTABILITY — Indication that tagged quantity lies in quadrant 1 of the two-dimensional Cartesian coordinate system (e.g., *x*)

FALSE	Tagged item is not in quadrant 1.
TRUE	Tagged item is in quadrant 1.

Q2_ACCOUNTABILITY — Indication that tagged quantity lies in quadrant 2 of the two-dimensional Cartesian coordinate system (e.g., *x*)

FALSE	Tagged item is not in quadrant 2.
TRUE	Tagged item is in quadrant 2.

Q3_ACCOUNTABILITY — Indication that tagged quantity lies in quadrant 3 of the two-dimensional Cartesian coordinate system (e.g., *x*)

FALSE	Tagged item is not in quadrant 3.
TRUE	Tagged item is in quadrant 3.

Q4_ACCOUNTABILITY — Indication that tagged quantity lies in quadrant 4 of the two-dimensional Cartesian coordinate system (e.g., *x*)

FALSE	Tagged item is not in quadrant 4.
TRUE	Tagged item is in quadrant 4.

NET_FLOW_ACCOUNTABILITY — This bit is required to identify the manner in which the quadrants specified are being summed.

FALSE	Absolute power delivered through selected quadrants, (i.e., power is added positively regardless of direction of flow.)
TRUE	Net of delivered—received, where watts are delivered in quadrants Q1 and Q4; received in quadrants Q2 and Q3; and VARs are delivered in quadrants Q1 and Q2; received in quadrants Q3 and Q4.

SEGMENTATION — When the **ID_CODE** field represents the electric utility industry unit of measures this bit indicates phase measurement associations.

0	Measurement is not a phase related or no phase information is applicable (e.g., all phases on a polyphase end device).
1	Phase A to B (i.e., A–B)
2	Phase B to C (i.e., B–C)
3	Phase C to A (i.e., C–A)

	4	Neutral to ground, or no phase information. (e.g., neutral current in a 4Y wire system.)
	5	Phase A to Neutral (i.e., A–N)
	6	Phase B to Neutral (i.e., B–N)
	7	Phase C to Neutral (i.e., C–N)
		When the **ID_CODE** does not represent an electric utility unit of measure, then this field represents an as yet to be defined quantity, except for the value 0, which stands for all sources and flows.
HARMONIC		This identifies harmonic related quantities.
	0	The identified **ID_CODE** is the entire signal unfiltered.
	1	The identified **ID_CODE** is a harmonic component of an associated source.
RESERVED		Bits presently unassigned.
NFS		Bit used to indicate that this unit of measure entry does not follow the unit of measure definition in some manner
UOM_ENTRY_RCD		
UOM_ENTRY		Each of the entries in this array is associated to an entry in the **SOURCES_TBL** (Table 16). This array contains one entry for each **SOURCES_TBL** (Table 16) entry having the **UOM_ENTRY_FLAG** set. The order of the parameters in this array corresponds to the order of the sources in table **SOURCES_TBL** (Table 16).

9.2.4 TABLE 13 Demand control table

Table 13 Data description

DEMAND_CONTROL_TBL (Table 13) contains information pertaining to the application of rates and global rate controls.

```
TYPE INT_CONTROL_RCD = PACKED RECORD

      IF ACT_SOURCES_LIM_TBL.SLIDING_DEMAND_FLAG THEN
            SUB_INT           : UINT8;
            INT_MULTIPLIER    : UINT8;
      ELSE
            INT_LENGTH        : UINT16;
      END;
END;
```



```
TYPE  DEMAND_CONTROL_RCD = PACKED RECORD
      IF ACT_SOURCES_LIM_TBL.RESET_EXCLUDE_FLAG THEN
            RESET_EXCLUSION          : UINT8;
      END;
      IF ACT_SOURCES_LIM_TBL.PF_EXCLUDE_FLAG THEN
            P_FAIL_RECOGNTN_TM       : UINT8;
            P_FAIL_EXCLUSION         : UINT8;
            COLD_LOAD_PICKUP         : UINT8;
      END;
      INTERVAL_VALUE: ARRAY[ACT_SOURCES_LIM_TBL.NBR_DEMAND_CTRL_ENTRIES]
                        OF INT_CONTROL_RCD;
END;

TABLE DEMAND_CONTROL_TBL = DEMAND_CONTROL_RCD;
```

Identifier	*Value*	*Definition*
INT_CONTROL_RCD		
SUB_INT	0..255	The number of minutes in the subinterval.
INT_MULTIPLIER	0..255	The multiplier by which the **SUB_INT** is multiplied.
INT_LENGTH	0..65535	The length of the demand interval in minutes.
DEMAND_CONTROL_RCD		
RESET_EXCLUSION	0..255	Number of minutes after demand reset to exclude additional reset action.
P_FAIL_RECOGNTN_TM	0..255	Number of seconds after a power failure occurs until a valid power failure is recorded and a specified action is initiated.
P_FAIL_EXCLUSION	0..255	Number of minutes after a valid power failure occurs to inhibit demand calculations.
COLD_LOAD_PICKUP	0..255	Number of minutes after a valid power failure occurs to provide cold load pickup functions.
INTERVAL_VALUE		Each of the entries in this array is associated to an entry in the **SOURCES_TBL** (Table 16). This array contains one entry for each **SOURCES_TBL** (Table 16) entry having the **DEMAND_CTRL_FLAG** set. The order of the parameters in this array corresponds to the order of the sources in table **SOURCES_TBL** (Table 16).

9.2.5 TABLE 14 Data control table

Table 14 Data description

DATA_CONTROL_TBL (Table 14) contains the data source information that serves as data input sources for the following tables. This table serves as an interface between areas of the end device considered manufacturer specific and the areas of the end device to be standardized.

Each entry in the table is manufacturer defined. Examples of entries into this table are: entry contains address of data to be processed; or entry contains data to be processed.

```
TYPE DATA_RCD = PACKED RECORD
      SOURCE_ID : ARRAY[ACT_SOURCES_LIM_TBL.DATA_CTRL_LENGTH]
                    OF UINT8;
END;
TYPE DATA_CONTROL_RCD = PACKED RECORD
      SOURCES_ID       : ARRAY[ACT_SOURCES_LIM_TBL.NBR_DATA_CTRL_ENTRIES]
                           OF DATA_RCD;
END;

TABLE DATA_CONTROL_TBL = DATA_CONTROL_RCD;
```

Identifier	*Value*	*Definition*
DATA_RCD		
SOURCE_ID		Each entry in this array is defined by the manufacturer as to length and content. The length is identical for all entries.
DATA_CONTROL_RCD		
SOURCES_ID		Each of the entries in this array is associated to an entry in the **SOURCES_TBL** (Table 16). This array contains one entry for each **SOURCES_TBL** (Table 16) entry having the **DATA_CTRL_FLAG** set. The order of the parameters in this array corresponds to the order of the sources in table **SOURCES_TBL** (Table 16).

9.2.6 TABLE 15 Constants table

Table 15 data description

CONSTANTS_TBL (Table 16) contains the record structures supporting constants for application to the sources. This table has been designed for easy inclusion of new constants structures.

```
{gas industry constants}
TYPE GAS_PRESS_RCD = PACKED RECORD
      GAS_PRESS_ZERO          : NI_FMAT2;
      GAS_PRESS_FULLSCALE     : NI_FMAT2;
      BASE_PRESSURE           : NI_FMAT2;
END;

TYPE GAS_TEMP_RCD = PACKED RECORD
      GAS_TEMP_ZERO           : NI_FMAT2;
      GAS_TEMP_FULLSCALE      : NI_FMAT2;
      BASE_TEMP               : NI_FMAT2;
END;

TYPE GAS_DP_RCD = PACKED RECORD
      GAS_DP_ZERO       : NI_FMAT2;
      GAS_DP_FULLSCALE  : NI_FMAT2;
```

```
END;

TYPE PIPE_ORIF_DIA_RCD = PACKED RECORD
      PIPE_DIA    : NI_FMAT2;
      ORIF_DIA    : NI_FMAT2;
END;

TYPE GAS_AGA3_CORR_RCD = PACKED RECORD
      AUX_CORR_FCTR           : NI_FMAT2;
      GAS_AGA3_CORR_FCTR      : NI_FMAT2;
      PIPE_ORIF_DIA           : PIPE_ORIF_DIA_RCD;
      TAP_UP_DN               : UINT8;
      GAS_PRESS_PARM          : GAS_PRESS_RCD;
      GAS_TEMP_PARM           : GAS_TEMP_RCD;
END;

TYPE GAS_AGA7_CORR_RCD = PACKED RECORD
      GAS_PRESS_PARM    : GAS_PRESS_RCD;
      GAS_TEMP_PARM     : GAS_TEMP_RCD;
      AUX_CORR_FCTR     : NI_FMAT2;
      GAS_AGA7_CORR     : NI_FMAT2;
END;

TYPE GAS_ENERGY_RCD = PACKED RECORD
      GAS_ENERGY_ZERO   : NI_FMAT2;
      GAS_ENERGY_FULL   : NI_FMAT2;
END;

TYPE GAS_DP_SHUTOFF_RCD = PACKED RECORD
      GAS_SHUTOFF       : NI_FMAT2;
END;
```

{Constant selector 1}

```
TYPE GAS_CONSTANTS_AGA3_RCD = PACKED RECORD
      GAS_DP_PARM       : GAS_DP_RCD;
      GAS_DP_SHUTOFF    : GAS_DP_SHUTOFF_RCD;
      GAS_PRESS_PARM    : GAS_PRESS_RCD;
      GAS_AGA3_CORR     : GAS_AGA3_CORR_RCD;
      GAS_ENERGY        : GAS_ENERGY_RCD;
END;
```

{Constant selector 2}

```
TYPE GAS_CONSTANTS_AGA7_RCD = PACKED RECORD
      GAS_AGA7_CORR     : GAS_AGA7_CORR_RCD;
      GAS_ENERGY        : GAS_ENERGY_RCD;
END;
```

{Constant selector 3}

```
TYPE SET_CTRL_BFLD = BIT FIELD OF UINT8
      SET_APPLIED_FLAG  : BOOL(0);
      FILLER            : FILL(1..7);
END;

TYPE SET_APPLIED_RCD = PACKED RECORD
```

```
      SET_FLAGS   : SET_CTRL_BFLD;
      RATIO_F1    : NI_FMAT1;
      RATIO_P1    : NI_FMAT1;
END;

TYPE ELECTRIC_CONSTANTS_RCD = PACKED RECORD
      MULTIPLIER              : NI_FMAT1;
      OFFSET                  : NI_FMAT1;
      IF ACT_SOURCES_LIM_TBL.SET1_PRESENT_FLAG THEN
            SET1_CONSTANTS    : SET_APPLIED_RCD;
      END;
      IF ACT_SOURCES_LIM_TBL.SET2_PRESENT_FLAG THEN
            SET2_CONSTANTS    : SET_APPLIED_RCD;
      END;
END;
```

{Constants record structure selection}

```
TYPE  CONSTANTS_RCD = PACKED RECORD
      CASE ACT_SOURCES_LIM_TBL.CONSTANTS_SELECTOR OF
            0           : GAS_CONSTANTS_AGA3 :GAS_CONSTANTS_AGA3_RCD;
            1           : GAS_CONSTANTS_AGA7 :GAS_CONSTANTS_AGA7_RCD;
            2           : ELECTRIC_CONSTANTS :ELECTRIC_CONSTANTS_RCD;
            3..255      : RESERVED          :NIL;
END;

TYPE CONSTANT_SELECTION_RCD = PACKED RECORD
      SELECTION   : ARRAY[ACT_SOURCES_LIM_TBL.NBR_CONSTANTS_ENTRIES]
                        OF CONSTANTS_RCD;
END;

TABLE CONSTANTS_TBL = CONSTANT_SELECTION_RCD;
```

Identifier	*Value*	*Definition*
GAS_PRESS_RCD		
GAS_PRESS_ZERO		Flow pressure calibration zero offset.
GAS_PRESS_FULLSCALE		Flow pressure calibration full scale.
BASE_PRESSURE		Pressure for base condition; this value is absolute base pressure.
GAS_TEMP_RCD		
GAS_TEMP_ZERO		Flow temperature calibration zero offset.
GAS_TEMP_FULLSCALE		Flow temperature calibration full scale.
BASE_TEMP		Temperature for base condition.
GAS_DP_RCD		
GAS_DP_ZERO		Flow differential pressure calibration zero offset (used for AGA 3 only).
GAS_DP_FULLSCALE		Flow differential pressure calibration full scale (used for AGA 3 only).

PIPE_ORIF_DIA_RCD

PIPE_DIA		The pipe diameter of an orifice meter (used for AGA 3 only).
ORIF_DIA		The orifice diameter of an orifice meter (used for AGA 3 only).

GAS_AGA3_CORR_RCD

AUX_CORR_FCTR		Factor to provide other possible correction to the gas corrected volume.
GAS_AGA3_CORR_FCTR		Factor is used to correct the uncorrected gas volume using AGA3 (used for AGA 3 only).
PIPE_ORIF_DIA		Record structure from above.
TAP_UP_DN		Tap configuration (used for AGA 3 only).
	0	No tap or flange.
	1	Upstream flange.
	2	Downstream flange.
	3	Upstream pipe.
	4	Downstream pipe.
	5..255	Reserved.
GAS_PRESS_PARM		Refer to **GAS_PRESS_RCD**.
GAS_TEMP_PARM		Refer to **GAS_TEMP_RCD**.

TYPE GAS_AGA7_CORR_RCD

GAS_PRESS_PARM		Refer to **GAS_PRESS_RCD**.
GAS_TEMP_PARM		Refer to **GAS_TEMP_RCD**.
AUX_CORR_FCTR		Factor to provide other possible correction to the gas corrected volume.
GAS_AGA7_CORR_FCTR		Factor is used to correct the uncorrected gas volume using AGA3 (used for AGA 7 only).

GAS_ENERGY_RCD

GAS_ENERGY_ZERO		Energy calibration zero offset.
GAS_ENERGY_FULL		Energy calibrations full scale.

TYPE GAS_DP_SHUTOFF_RCD

GAS_DP_SHUTOFF		This parameter is used to disable the AGA3 flow calculations when differential pressure is below the value indicated by this parameter (used for AGA 3 only).

{Constants selector 0}

GAS_CONSTANTS_AGA3_RCD

GAS_DP_PARM		Refer to **GAS_DP_RCD**.

GAS_DP_SHUTOFF		Refer to **GAS_DP_SHUTOFF_RCD**.
GAS_PRESS_PARM		Refer to **GAS_PRESS_RCD**.
GAS_AGA3_CORR_FCTR		Refer to **GAS_AGA3_CORR_RCD**.
GAS_ENERGY		Refer to **GAS_ENERGY_RCD**.

{Constants selector 1}

GAS_CONSTANTS_AGA7_RCD

GAS_AGA7_CORR_FCTR		Refer to **GAS_AGA7_CORR_RCD**.
GAS_ENERGY		Refer to **GAS_ENERGY_RCD**.

{Constants selector 2}

SET_CTRL_BFLD

SET_APPLIED_FLAG	FALSE	The **RATIO_F1** and **RATIO** P1 represented by this flag have not been applied to the associated source.
	TRUE	The **RATIO_P1** and **RATIO** F1 represented by this flag have been applied to the associated source.

SET_APPLIED_RCD

SET_FLAGS	See **SET_CTRL_BFLD** above.
RATIO_F1	Ratio of intermediary device to allow interface of commodity flow to utility meters. For example, electric utilities use a current transformer to reduce the current at the meter; e.g., transforming 2000 A to 5 A is a ratio of 400.
RATIO_P1	Ratio of intermediary device to allow interface of commodity pressure to utility meters. For example, electric utilities use a voltage transformer to reduce the voltage at the meter; e.g., transforming 7200 V to 120 V is a ratio of 60.

ELECTRIC_CONSTANTS_RCD

MULTIPLIER	Storage for a value used in multiplication/ division adjustment.
OFFSET	Storage for a value used in addition/ subtraction adjustment.
SET1_CONSTANTS	Storage variable for the first set of constants and associated flags.
SET2_CONSTANTS	Storage variable for the second set of constants and associated flags.

CONSTANTS_RCD	Structure that selects the constant record structure to use.
CONSTANT_SELECTION_RCD	Each of the entries in this array is associated to an entry in the **SOURCES_TBL** (Table 16). This array contains one entry for each **SOURCES_TBL** (Table 16) entry having the **CONSTANTS_FLAG** set. The order of the parameters in this array corresponds to the order of the sources in table **SOURCES_TBL** (Table 16).

9.2.7 TABLE 16 Source definition table

Table 16 Data description

SOURCES_TBL (Table 16) contains the information regarding the sources selected by the other tables.

```
TYPE SOURCE_LINK_BFLD = BIT FIELD OF UINT8
      UOM_ENTRY_FLAG           : BOOL(0);
      DEMAND_CTRL_FLAG         : BOOL(1);
      DATA_CTRL_FLAG           : BOOL(2);
      CONSTANTS_FLAG           : BOOL(3);
      PULSE_ENGR_FLAG          : BOOL(4);
      CONSTANT_TO_BE_APPLIED   : BOOL(5);
      FILLER                   : FILL(6..7);
END;

TYPE  SOURCE_LINK_RCD = PACKED RECORD
      SOURCES_LINK      : ARRAY[ACT_SOURCES_LIM_TBL.NBR_SOURCES] OF
                               SOURCE_LINK_BFLD;
END;

TABLE SOURCES_TBL = SOURCE_LINK_RCD;
```

Identifier	*Value*	*Definition*
SOURCE_LINK_BFLD		
UOM_ENTRY_FLAG	FALSE	A **UOM_ENTRY_TBL** (Table 12) entry does not exist for this source.
	TRUE	The next **UOM_ENTRY_TBL** (Table 12) entry is associated with this source. It is recommended that the **UOM** entries reflect the source as it is referenced or transported by other tables.
DEMAND_CTRL_FLAG	FALSE	A **DEMAND_CONTROL_TBL** (Table 13) entry does not exist for this source.
	TRUE	A **DEMAND_CONTROL_TBL** (Table 13) entry is associated with this source.
DATA_CTRL_FLAG	FALSE	A **DATA_CONTROL_TBL** (Table 14) entry does not exist for this source.
	TRUE	A **DATA_CONTROL_TBL** (Table 14) entry is associated with this source.

CONSTANTS_FLAG	FALSE	A **CONSTANTS_TBL** (Table 15) entry does not exist for this source.
	TRUE	A **CONSTANTS_TBL** (Table 15) entry is associated with this source.
PULSE_ENGR_FLAG	FALSE	The source is in pulse units.
	TRUE	The source is in engineering units.
CONSTANT_TO_BE_APPLIED		
	FALSE	The entry in the **CONSTANTS_TBL** (Table 15) if present, has been applied to the source. This flag does not apply when **ACT_SOURCES_LIM_TBL.CONSTANTS_SELECTOR** is 3. Examine **CONSTANTS_TBL** (Table 15) **SET_APPLIED_FLAG** to determine if constants have been applied.
	TRUE	The entry in the **CONSTANTS_TBL** (Table 15) if present, has not been applied to the source. This flag does not apply when **ACT_SOURCES_LIM_TBL.CONSTANTS_SELECTOR** is 3. Examine **CONSTANTS_TBL** (Table 15) **SET_APPLIED_FLAG** to determine if constants have been applied.

9.3 DECADE 20: Register tables

This decade contains tables associated with registers for measured values. These registers can be accumulating (energy), demand, TOU, instantaneous, or other(s) as appropriate.

9.3.1 TABLE 20 Dimension register table

Table 20 Data description

DIM_REGS_TBL (Table 20) specifies the maximum dimensional values for measured values registers. The constants defined are used for setting the absolute maximum limits of arrays used in the transport of these values.

```
TYPE REG_FUNC1_BFLD = BIT FIELD OF UINT8
      SEASON_INFO_FIELD_FLAG         : BOOL(0);
      DATE_TIME_FIELD_FLAG           : BOOL(1);
      DEMAND_RESET_CTR_FLAG          : BOOL(2);
      DEMAND_RESET_LOCK_FLAG         : BOOL(3);
      CUM_DEMAND_FLAG                : BOOL(4);
      CONT_CUM_DEMAND_FLAG           : BOOL(5);
      TIME_REMAINING_FLAG            : BOOL(6);
      FILLER                         : FILL(7..7);
END;

TYPE REG_FUNC2_BFLD = BIT FIELD OF UINT8
      SELF_READ_INHIBIT_OVERFLOW_FLAG     : BOOL(0);
      SELF_READ_SEQ_NBR_FLAG              : BOOL(1);
      DAILY_SELF_READ_FLAG                : BOOL(2);
      WEEKLY_SELF_READ_FLAG               : BOOL(3);
      SELF_READ_DEMAND_RESET              : UINT(4..5);
      FILLER                              : FILL(6..7);
END;
```

```
TYPE REGS_RCD = PACKED RECORD
      REG_FUNC1_FLAGS          : REG_FUNC1_BFLD;
      REG_FUNC2_FLAGS          : REG_FUNC2_BFLD;
      NBR_SELF_READS           : UINT8;
      NBR_SUMMATIONS           : UINT8;
      NBR_DEMANDS              : UINT8;
      NBR_COIN_VALUES          : UINT8;
      NBR_OCCUR                : UINT8;
      NBR_TIERS                : UINT8;
      NBR_PRESENT_DEMANDS      : UINT8;
      NBR_PRESENT_VALUES       : UINT8;
END;

TABLE DIM_REGS_TBL = REGS_RCD;
```

Identifier	*Value*	*Definition*
REG_FUNC1_BFLD		
SEASON_INFO_FIELD_FLAG	FALSE	End device is not capable of reporting the representative season in tables in this decade.
	TRUE	End device is capable of reporting the representative season in tables in this decade.
DATE_TIME_FIELD_FLAG	FALSE	End device is not capable of providing the date and time in tables in this decade.
	TRUE	End device is capable of providing the date and time in tables in this decade.
DEMAND_RESET_CTR_FLAG	FALSE	End device is not capable of counting the number of activations of demand resets.
	TRUE	End device is capable of counting the number of activations of demand resets.
DEMAND_RESET_LOCK_FLAG	FALSE	Demand reset lockout is not supported by the end-device.
	TRUE	Demand reset lockout is supported by the end device.
CUM_DEMAND_FLAG	FALSE	Cumulative demand is not supported by the end-device.
	TRUE	Cumulative demand is supported by the end device.
CONT_CUM_DEMAND_FLAG	FALSE	Continuous cumulative demand is not supported by the end device.
	TRUE	Continuous cumulative demand is supported by the end device.
TIME_REMAINING_FLAG	FALSE	End device does not have the ability to report time remaining in demand interval.
	TRUE	End device does have the ability to report time remaining in demand interval.
REG_FUNC2_BFLD		
SELF_READ_INHIBIT_OVERFLOW_FLAG		
	FALSE	End device is not capable of inhibiting self reads once an overflow occurs.

		TRUE	End device is capable of inhibiting self reads once an overflow occurs.
	SELF_READ_SEQ_NBR_FLAG	FALSE	End device is not capable of providing a self read sequential number for each entry.
		TRUE	End device is capable of providing a self read sequential number.
	DAILY_SELF_READ_FLAG		Indicates whether daily self reads are supported. These readings are taken at 00:00:00.
		FALSE	Daily reading is not supported.
		TRUE	Daily reading is supported.
	WEEKLY_SELF_READ_FLAG		Indicates whether weekly self reads are supported. These readings are taken at 00:00:00 on Sunday.
		FALSE	Weekly reading is not supported.
		TRUE	Weekly reading is supported.
	SELF_READ_DEMAND_RESET		Specifies whether the device is capable of performing a self read whenever a demand reset (any method) is performed and whether the device is capable of performing a demand reset whenever a self read (any method) is performed.
		0	End device is not capable of performing either a self read on every demand reset or a demand reset on every self read.
		1	End device is capable of performing only a self read on every demand reset.
		2	End device is capable of performing only a demand reset on every self read.
		3	End device is capable of performing a self read on every demand reset and a demand reset on every self read.
REGS_RCD			
	REG_FUNC1_FLAGS		See definition of **REG_FUNC1_BFLD** above.
	REG_FUNC2_FLAGS		See definition of **REG_FUNC2_BFLD** above.
	NBR_SELF_READS	0..255	Maximum number of self reads supported by end-device.
	NBR_SUMMATIONS	0..255	Maximum number of summation registers in each data block.
	NBR_DEMANDS	0..255	Maximum number of demand registers in each data block.
	NBR_COIN_VALUES	0..255	Maximum number of coincident values saved concurrently in each data block.
	NBR_OCCUR	0..255	Maximum number of occurrences stored for a particular selection.

NBR_TIERS	0..255	Maximum number of tiers that data can be stored in.
NBR_PRESENT_DEMANDS	0..255	Maximum number of present demand values that can be stored.
NBR_PRESENT_VALUES	0..255	Maximum number of present values that can be stored.

9.3.2 TABLE 21 Actual register table

Table 21 Data description

ACT_REGS_TBL (Table 21) contains actual function values for registers.

TABLE ACT_REGS_TBL = DIM_REGS_TBL.REGS_RCD;

Identifier	*Value*	*Definition*
REG_FUNC1_FLAGS		
SEASON_INFO_FIELD_FLAG	FALSE	End device is not reporting the representative season in tables in this decade.
	TRUE	End device is reporting the representative season in tables in this decade.
DATE_TIME_FIELD_FLAG	FALSE	End device is not providing the date and time in tables in this decade.
	TRUE	End device is providing the date and time in tables in this decade.
DEMAND_RESET_CTR_FLAG	FALSE	End device is not counting the number of activations of demand resets.
	TRUE	End device is counting the number of activations of demand resets.
DEMAND_RESET_LOCK_FLAG	FALSE	Demand reset lockout is not enabled.
	TRUE	Demand reset lockout is enabled.
CUM_DEMAND_FLAG	FALSE	Cumulative demand is not in use by the end device.
	TRUE	Cumulative demand is in use by the end device.
CONT_CUM_DEMAND_FLAG	FALSE	Continuous cumulative demand is not in use by the end device.
	TRUE	Continuous cumulative demand is in use by the end device.
TIME_REMAINING_FLAG	FALSE	End device does not report time remaining in demand interval.
	TRUE	End device does report time remaining in demand interval.

REG_FUNC2_BFLD

SELF_READ_INHIBIT_OVERFLOW_FLAG		
	FALSE	End device is not inhibiting self reads once an overflow occurs.
	TRUE	End device is inhibiting self reads once an overflow occurs.
SELF_READ_SEQ_NBR_FLAG	FALSE	End device is not providing a self read sequential number for each entry.
	TRUE	End device is providing a self read sequential number.
DAILY_SELF_READ_FLAG		Indicates whether daily self reads are in use. These readings are taken at 00:00:00.
	FALSE	Daily reading is not in use.
	TRUE	Daily reading is in use.
WEEKLY_SELF_READ_FLAG		Indicates whether weekly self reads are in use. These readings are taken at 00:00:00 on Sunday.
	FALSE	Weekly reading is not in use.
	TRUE	Weekly reading is in use.
SELF_READ_DEMAND_RESET		Specifies whether the device will perform a self read whenever a demand reset (any method) is performed and whether the device will perform a demand reset whenever a self read (any method) is performed.
	0	End device shall not perform either a self read on every demand reset or a demand reset on every self read.
	1	End device shall perform a self read on every demand reset.
	2	End device shall perform demand reset on every self read.
	3	End device shall perform a self read on every demand reset and a demand reset on every self read.

REGS_RCD

REG_FUNC1_FLAGS		See definition of **REG_FUNC1_BFLD** above.
REG_FUNC2_FLAGS		See definition of **REG_FUNC2_BFLD** above.
NBR_SELF_READS		Number of self reads in use.
NBR_SUMMATIONS	0..255	Number of summation registers in each data block.
NBR_DEMANDS	0..255	Number of demand registers in each data block.
NBR_COIN_VALUES	0..255	Number of coincident values saved in each data block.
NBR_OCCUR	0..255	Number of occurrences stored for a particular selection.
NBR_TIERS	0..255	Number of tiers in use.

NBR_PRESENT_DEMANDS	0..255	Number of present demand values that are stored.
NBR_PRESENT_VALUES	0..255	Number of present values that are stored.

9.3.3 TABLE 22 Data selection table

Table 22 Data description

DATA_SELECTION_TBL (Table 22) contains grouped lists of source indices. These indices point towards array elements in **SOURCES_TBL** (Table 16) when this table is present. These lists are used to build the table that contains the data to be captured. The groupings are described below.

```
TYPE DATA_SELECTION_RCD = PACKED RECORD
      SUMMATION_SELECT  : ARRAY[ACT_REGS_TBL.NBR_SUMMATIONS] OF UINT8;
      DEMAND_SELECT     : ARRAY[ACT_REGS_TBL.NBR_DEMANDS] OF UINT8;
      MIN_OR_MAX_FLAGS  : SET((ACT_REGS_TBL.NBR_DEMANDS+7)/8);
      COINCIDENT_SELECT : ARRAY[ACT_REGS_TBL.NBR_COIN_VALUES]OF UINT8;
      COIN_DEMAND_ASSOC : ARRAY[ACT_REGS_TBL.NBR_COIN_VALUES] OF UINT8;
END;

TABLE DATA_SELECTION_TBL = DATA_SELECTION_RCD;
```

Identifier	*Value*	*Definition*
DATA_SELECTION_RCD		
SUMMATION_SELECT		A list of source indices that groups together bulk/energy/summation sources.
DEMAND_SELECT		A list of source indices that groups together demand sources.
MIN_OR_MAX_FLAGS		A set of bit flags, each corresponding to an entry in **DEMAND_SELECT**.
	FALSE	Indicates that the associated **DEMAND_SELECT** entry is a minimum.
	TRUE	Indicates that the associated **DEMAND_SELECT** entry is a maximum.
COINCIDENT_SELECT		A list of source indices that groups together demand sources with time period processing equal to that of the **DEMAND_SELECT** sources.
COIN_DEMAND_ASSOC		Each entry corresponds to an entry in **COINCIDENT_SELECT** and provides an index into **DEMAND_SELECT** identifying the demand for which this coincident value is taken.

9.3.4 TABLE 23 Current register data table

Table 23 Data description

CURRENT_REG_DATA_TBL (Table 23) contains the current register data.

```
TYPE COINCIDENTS_RCD = PACKED RECORD
      COINCIDENT_VALUES : ARRAY[ACT_REGS_TBL.NBR_OCCUR] OF NI_FMAT2;
END;
TYPE DEMANDS_RCD = PACKED RECORD
      IF ACT_REGS_TBL.DATE_TIME_FIELD_FLAG THEN
            EVENT_TIME        : ARRAY[ACT_REGS_TBL.NBR_OCCUR] OF
                                STIME_DATE;
      END;
      IF ACT_REGS_TBL.CUM_DEMAND_FLAG THEN
            CUM_DEMAND        : NI_FMAT1;
      END;
      IF ACT_REGS_TBL.CONT_CUM_DEMAND_FLAG THEN
            CONT_CUM_DEMAND   : NI_FMAT1;
      END;
      DEMAND                  : ARRAY[ACT_REGS_TBL.NBR_OCCUR] OF
                                NI_FMAT2;
END;

TYPE DATA_BLK_RCD = PACKED RECORD
      SUMMATIONS  : ARRAY[ACT_REGS_TBL.NBR_SUMMATIONS] OF NI_FMAT1;
      DEMANDS     : ARRAY[ACT_REGS_TBL.NBR_DEMANDS] OF DEMANDS_RCD;
      COINCIDENTS : ARRAY[ACT_REGS_TBL.NBR_COIN_VALUES]OF
COINCIDENTS_RCD;
END;

TYPE REGISTER_DATA_RCD = PACKED RECORD
      IF ACT_REGS_TBL.DEMAND_RESET_CTR_FLAG THEN
            NBR_DEMAND_RESETS : UINT8;
      END;
      TOT_DATA_BLOCK          : DATA_BLK_RCD;
      TIER_DATA_BLOCK         : ARRAY[ACT_REGS_TBL.NBR_TIERS] OF
                                DATA_BLK_RCD;
END;

TABLE CURRENT_REG_DATA_TBL = REGISTER_DATA_RCD;
```

Identifier	*Value*	*Definition*
COINCIDENTS_RCD		
COINCIDENT_VALUES		Coincident values selected by **DATA_SELECTION_TBL.COINCIDENT_SELECT** (Table 22).
DEMANDS_RCD		
EVENT_TIME		Array that contains the date and time of each minimum or maximum recorded. Entry in position 0 corresponds to the entry in position 0 of the DEMAND array. The presence of this

	field is dependent on capability flag in the **ACT_REG_TBL.DATE_TIME_FIELD_FLAG** (Table 21).
CUM_DEMAND	The value of the cumulative demand register.
CONT_CUM_DEMAND	The value of the continuous cumulative demand register.
DEMAND	Array that contains the minimums or maximums values. The value in position zero is the smallest minimum or largest maximum. In position 1 is the second smallest minimum or the second largest maximum and so on until the end of the array is reached.
DATA_BLK_RCD	
SUMMATIONS	Array that contains the values of the selected summation measurements.
DEMANDS	Array that contains the values of the selected min/max demand measurements.
COINCIDENTS	Array that contains the values of the selected measurement coincident to the selected minimum/maximum demands.
REGISTER_DATA_RCD	
NBR_DEMAND_RESETS	Indicates the number of demand resets executed by the end device.
TOT_DATA_BLOCK	Data block independent of TOU structures or other means of selection.
TIER_DATA_BLOCK	Data block dependent on TOU structures or other means of selection.

9.3.5 TABLE 24 Previous season data table

Table 24 Data description

PREVIOUS_SEASON_DATA_TBL (Table 24) contains a snapshot of the current register data taken at the last season change.

```
TYPE REGISTER_INFO_RCD = PACKED RECORD
      IF ACT_REGS_TBL.DATE_TIME_FIELD_FLAG THEN
            END_DATE_TIME     : STIME_DATE;
      END;
      IF ACT_REGS_TBL.SEASON_INFO_FIELD_FLAG THEN
            SEASON            : UINT8;
      END;
END;

TYPE PREVIOUS_SEASON_DATA_RCD = PACKED RECORD
      REGISTER_INFO           : REGISTER_INFO_RCD;
      PREV_SEASON_REG_DATA    : CURRENT_REG_DATA_TBL.REGISTER_DATA_RCD;
END;

TABLE PREVIOUS_SEASON_DATA_TBL = PREVIOUS_SEASON_DATA_RCD;
```

Identifier	*Value*	*Definition*
REGISTER_INFO_RCD		
END_DATE_TIME		The end device time when the snapshot of the current register data occurred.
SEASON		Current season value when the snapshot of the current register data occurred.
PREVIOUS_SEASON_DATA_RCD		
REGISTER_INFO		See definitions above.
PREV_SEASON_REG_DATA		Contains a snapshot of the **CURRENT_DATA_REG_TBL** (Table 23). stored at the time of the last season change. See definition in **CURRENT_DATA_REG_TBL** (Table 23).

9.3.6 TABLE 25 Previous demand reset data table

Table 25 Data description

PREVIOUS_DEMAND_RESET_DATA_TBL (Table 25) contains a snapshot of the current register data taken at the time of the last demand reset.

```
TYPE PREV_DEMAND_RESET_DATA_RCD = PACKED RECORD
      REGISTER_INFO     : PREVIOUS_SEASON_DATA_TBL.REGISTER_INFO_RCD;
      PREV_DEMAND_RESET_DATA  : CURRENT_REG_DATA_TBL.REGISTER_DATA_RCD;
END;

TABLE PREVIOUS_DEMAND_RESET_DATA_TBL = PREV_DEMAND_RESET_DATA_RCD;
```

Identifier	*Value*	*Definition*
PREV_DEMAND_RESET_DATA_RCD		
REGISTER_INFO		Contains snapshot of register informationinfomation included with previous demand reset register data.
PREV_DEMAND_RESET_DATA		Contains a snapshot of the **CURRENT_DATA_REG_TBL** (Table 23) stored at the time of the last demand reset.

9.3.7 TABLE 26 Self read data table

Table 26 Data description

SELF_READ_DATA_TBL (Table 26) contains self read data. Snapshots of current register data shall be taken from time to time by some sort of self read function.

```
TYPE SELF_READ_DATA_RCD = PACKED RECORD
      IF ACT_REGS_TBL.SELF_READ_SEQ_NBR_FLAG THEN
            SELF_READ_SEQ_NBR : UINT16;
      END;
      REGISTER_INFO     : PREVIOUS_SEASON_DATA_TBL.REGISTER_INFO_RCD;
      SELF_READ_REGISTER_DATA : CURRENT_REG_DATA_TBL.REGISTER_DATA_RCD;
END;
```

```
TYPE LIST_STATUS_BFLD = BIT FIELD OF UINT8
      ORDER_FLAG              : BOOL(0);
      OVERFLOW_FLAG           : BOOL(1);
      LIST_TYPE_FLAG          : BOOL(2);
      INHIBIT_OVERFLOW_FLAG   : BOOL(3);
      FILLER                  : FILL(4..7);
END;

TYPE SELF_READ_LIST_RCD = PACKED RECORD
      LIST_STATUS             : LIST_STATUS_BFLD;
      NBR_VALID_ENTRIES       : UINT8;
      LAST_ENTRY_ELEMENT      : UINT8;
      LAST_ENTRY_SEQ_NBR      : UINT16;
      NBR_UNREAD_ENTRIES      : UINT8;
      SELF_READS_ENTRIES      : ARRAY[ACT_REGS_TBL.NBR_SELF_READS] OF
                                    SELF_READ_DATA_RCD;
END;

TABLE SELF_READ_DATA_TBL = SELF_READ_LIST_RCD;
```

Identifier	*Value*	*Definition*
SELF_READ_DATA_RCD		
SELF_READ_SEQ_NBR		Sequence number assigned to the self read measurement. This number is initialized to 0 when the end device is configured. This number is incremented with each self read occurrence.
REGISTER_INFO		Contains a snapshot of register data taken at the time of the self read.
SELF_READ_REGISTER_DATA		A snapshot of current register data taken when requested by any means offered by the end device.
LIST_STATUS_BFLD		
ORDER_FLAG	FALSE	Self reads are transported in ascending order (N is older than N+1).
	TRUE	Self reads are transported in descending order (N is newer than N+1).
OVERFLOW_FLAG	FALSE	Overflow has not occurred.
	TRUE	An attempt was made to enter a self read such that the number of unread self reads would have exceeded the actual number of possible entries in the log.
LIST_TYPE_FLAG	FALSE	FIFO (First In First Out) as placed in self read list.
	TRUE	Circular list as placed in self read list.
INHIBIT_OVERFLOW_FLAG		The same value as **ACT_REG_TBL. SELF_READ_INHIBIT_OVERFLOW** (Table 21).
SELF_READ_LIST_RCD		
LIST_STATUS		See **LIST_STATUS_BFLD** above.
NBR_VALID_ENTRIES	0..255	Number of valid self reads stored in the self reads array. The range is zero (meaning no self reads in the

		array) to the actual dimension of the number of self reads.
LAST_ENTRY_ELEMENT	0..65535	The array element number of the newest valid entry in the self read array. This field is valid only if **NBR_VALID_ENTRIES** is greater than zero.
LAST_ENTRY_SEQ_NBR	0..65535	The sequence number of the last element in the self read array.
NBR_UNREAD_ENTRIES	0..255	The number of self read entries that have not been read. This number is only updated by a procedure.
SELF_READS_ENTRIES		An array of snapshots of register data taken at a prescribed instance in time.

9.3.8 TABLE 27 Present register selection table

Table 27 Data description

PRESENT_REGISTER_SELECT_TBL (Table 27) provides selections to sources to present demands and present values of data sources. These indices point towards array elements in **SOURCES_TBL** (Table 16) when Table 16 is present. The values selected are available in **PRESENT_REGISTER_DATA_TBL** (Table 28).

```
TYPE PRESENT_REGISTER_SELECT_RCD = PACKED RECORD
      PRESENT_DEMAND_SELECT   : ARRAY[ACT_REGS_TBL.NBR_PRESENT_DEMANDS]
                                OF UINT8;
      PRESENT_VALUE_SELECT    : ARRAY[ACT_REGS_TBL.NBR_PRESENT_VALUES]
                                OF UINT8;
END;

TABLE PRESENT_REGISTER_SELECT_TBL = PRESENT_REGISTER_SELECT_RCD;
```

Identifier	*Value*	*Definition*
PRESENT_REGISTER_SELECT_RCD		
PRESENT_DEMAND_SELECT		Array that contains data source selectors used to measure present demand. The present demand values shall be reported at the same index position in **PRESENT_REGISTER_DATA_TBL.PRESENT_DEMAND** (Table 28).
PRESENT_VALUE_SELECT		Array that contains data source selectors used to measure present value. The present values shall be reported at the same index position in **PRESENT_REGISTER_DATA_TBL. PRESENT_VALUES** (Table 28).

9.3.9 TABLE 28: Present register data table

Table 28 Data description

PRESENT_REGISTER_DATA_TBL (Table 28) contains the present demand and values as selected by the **PRESENT_REGISTER_SELECT_TBL** (Table 27). These indices point towards array elements in **SOURCES_TBL** (Table 16) when Table 16 is present.

```
TYPE PRESENT_DEMAND_RCD = PACKED RECORD
      IF ACT_REGS_TBL.TIME_REMAINING_FLAG THEN
            TIME_REMAINING    : TIME;
      END;
      DEMAND_VALUE            : NI_FMAT2;
END;

TYPE PRESENT_REGISTER_DATA_RCD = PACKED RECORD
      PRESENT_DEMAND    : ARRAY[ACT_REGS_TBL.NBR_PRESENT_DEMANDS] OF
                              PRESENT_DEMAND_RCD;
      PRESENT_VALUE     : ARRAY[ACT_REGS_TBL.NBR_PRESENT_VALUES]
      OF NI_FMAT1;
END;

TABLE PRESENT_REGISTER_DATA_TBL = PRESENT_REGISTER_DATA_RCD;
```

Identifier	*Value*	*Definition*
PRESENT_DEMAND_RCD		
	TIME_REMAINING	Time remaining to the end of the present demand interval
	DEMAND_VALUE	The present demand value.
PRESENT_REGISTER_DATA_RCD		
	PRESENT_DEMAND	Array that contains present demand selected by **PRESENT_REGISTER_SELECT_TBL.PRESENT_DEMAND_SELECT** (Table 27).
	PRESENT_VALUE	Array that contains present demand selected by **PRESENT_REGISTER_SELECT_TBL.PRESENT_DEMAND_SELECT** (Table 27).

9.4 DECADE 30: Local display tables

This decade contains tables associated with the transport of information required to establish a local display.

9.4.1 TABLE 30 Dimension display table

Table 30 Data description

DIM_DISP_TBL (Table 30) specifies the maximum dimensional values for local display operation in an end device. The constants defined are used for setting the absolute maximum limits of display array sizes.

```
TYPE DISP_FLAG_BFLD = BIT FIELD OF UINT8
      ON_TIME_FLAG            : BOOL(0);
      OFF_TIME_FLAG           : BOOL(1);
      HOLD_TIME_FLAG          : BOOL(2);
      FILLER                  : FILL(3..7);
END;
TYPE DISP_RCD = PACKED RECORD
      DISPLAY_CTRL            : DISP_FLAG_BFLD;
      NBR_DISP_SOURCES        : UINT16;
      WIDTH_DISP_SOURCES      : UINT8;
      NBR_PRI_DISP_LIST_ITEMS : UINT16;
      NBR_PRI_DISP_LISTS      : UINT8;
      NBR_SEC_DISP_LIST_ITEMS : UINT16;
      NBR_SEC_DISP_LISTS      : UINT8;
END;

TABLE DIM_DISP_TBL = DISP_RCD;
```

Identifier	*Value*	*Definition*
DISP_FLAG_BFLD		
ON_TIME_FLAG	FALSE	Programmable on time is not supported by the end device.
	TRUE	Programmable on time is supported by the end device.
OFF_TIME_FLAG	FALSE	Programmable off time is not supported by the end device.
	TRUE	Programmable off time is supported by the end device.
HOLD_TIME_FLAG	FALSE	Programmable hold time is not supported by the end device.
	TRUE	Programmable hold time is supported by the end device.
DISP_RCD		
DISPLAY_CTRL		See **DISP_FLAG_BFLD** above.
NBR_DISP_SOURCES	0..65535	Maximum number of display sources. A number representing the maximum number of manufacturer defined display sources, described in **DISP_SOURCE_TBL** (Table 32), and that may be used for the creation of display lists.
WIDTH_DISP_SOURCES	0..255	Maximum number of octets a manufacturer may use to define the row width of all entries in **DISP_SOURCE_TBL** (Table 32).
NBR_PRI_DISP_LIST_ITEMS	0..65535‘	Maximum number of primary display items per display list defined in **PRI_DISP_LIST_TBL** (Table 33). The maximum number of items in all primary display lists shallshould not exceed this value.

NBR_PRI_DISP_LISTS	0..255	Maximum number of primary display lists defined in **PRI_DISP_LIST_TBL** (Table 33).
NBR_SEC_DISP_LIST_ITEMS	0..65535	Maximum number of secondary display items per display list defined in **SEC_DISP_LIST_TBL** (Table 34). The maximum number of items in all secondary display lists shallshould not exceed this value.
NBR_SEC_DISP_LISTS	0..255	Maximum number of secondary display lists defined in **SEC_DISP_LIST_TBL** (Table 34).

9.4.2 TABLE 31 Actual display table

Table 31 Data description

ACT_DISP_TBL (Table 31) specifies the actual dimensional values for local display operation in an end device. The dimensions defined are used for setting the actual limits of display array sizes.

TABLE ACT_DISP_TBL = DIM_DISP_TBL.DISP_RCD;

Identifier	*Value*	*Definition*
DISP_FLAG_BFLD		
ON_TIME_FLAG	FALSE TRUE	On time is not programmable in the end device. On time is programmable in the end device.
OFF_TIME_FLAG	FALSE TRUE	Off time is not programmable in the end device. Off time is programmable in the end device.
HOLD_TIME_FLAG	FALSE TRUE	Hold time is not programmable in the end device. Hold time is programmable in the end device.
DISP_RCD		
DISPLAY_CTRL		See **DISP_FLAG_BFLD** above.
NBR_DISP_SOURCES	0..65535	Actual number of display sources. A number representing the actual number of manufacturer defined display sources, described in **DISP_SOURCE_TBL** (Table 32), and that should bemay be used for the creation of display lists.
WIDTH_DISP_SOURCES	0..255	Actual number of octets a manufacturer uses to define the row width of all entries in **DISP_SOURCE_TBL** (Table 32).
NBR_PRI_DISP_LIST_ITEMS	0..65535	Actual number of primary display items per display list defined in **PRI_DISP_LIST_TBL** (Table 33). The total number of items in all primary display lists shallshould not exceed this value.

NBR_PRI_DISP_LISTS	0..255	Actual number of primary display lists defined in **PRI_DISP_LIST_TBL** (Table 33).
NBR_SEC_DISP_LIST_ITEMS	0..65535	Actual number of secondary display items per display list defined in **SEC_DISP_LIST_TBL** (Table 34). The total number of items in all secondary display lists shallshould not exceed this value.
NBR_SEC_DISP_LISTS	0..255	Actual number of secondary display lists defined in **SEC_DISP_LIST_TBL** (Table 34).

9.4.3 TABLE 32 Display source table

Table 32 Data description

DISP_SOURCE_TBL (Table 32) defines the dimensions and size of manufacturer defined display identifications and format characteristics. Although manufacturer specific, this table's designation and dimensions (**ACT_DISP_TBL.WIDTH_DISP_SOURCES BY ACT_DISP_TBL.NBR_DISP_SOURCES**) (Table 31), permit the protocol to deposit local display description information. The actual interpretation of each display entry is not defined by this standard. The entry offsets to individual items in this table are used in display scroll sequences entered in tables **PRI_DISP_LIST_TBL** (Table 33) and **SEC_DISP_LIST_TBL** (Table 34).

```
TYPE DISP_SOURCE_DESC_RCD = PACKED RECORD
      DISPLAY_SOURCE    : ARRAY[ACT_DISP_TBL.WIDTH_DISP_SOURCES]
                              OF UINT8;
END;
TYPE DISPLAY_SOURCE_RCD  = PACKED RECORD
      DISPLAY_SOURCES   : ARRAY[ACT_DISP_TBL.NBR_DISP_SOURCES] OF
                              DISP_SOURCE_DESC_RCD;
END;

TABLE DISP_SOURCE_TBL = DISPLAY_SOURCE_RCD;
```

Identifier	*Value*	*Definition*
DISP_SOURCE_DESC_RCD		Manufacturer defined display description entry.
DISPLAY_SOURCE	0..255	These entries make up the **DISP_SOURCE_TBL** (Table 32). Indices to these entries (starting with 0) are used in subsequent display scrolling sequences.
DISPLAY_SOURCE_RCD		Manufacturer defined display description entries.
DISPLAY_SOURCES		This array contains the manufacturer defined display description entries.

9.4.4 TABLE 33 Primary display list table

Table 33 Data description

PRI_DISP_LIST_TBL (Table 33) contains a list of operational parameters and references into the table, **DISP_SOURCE_TBL**, (Table 32) necessary for programming the primary display lists, timing, and scroll sequences.

```
TYPE DISP_SCROLL1_BFLD = BIT FIELD OF UINT8
      ON_TIME           : UINT(0..3);
      OFF_TIME          : UINT(4..7);
END;

TYPE DISP_SCROLL2_BFLD  = BIT FIELD OF UINT8
      HOLD_TIME         : UINT(0..3);
      DEFAULT_LIST      : UINT(4..7);
END;

TYPE DISP_LIST_DESC_RCD = PACKED RECORD
      DISP_SCROLL1      : DISP_SCROLL1_BFLD;
      DISP_SCROLL2      : DISP_SCROLL2_BFLD;
      NBR_LIST_ITEMS    : UINT8;
END;

TYPE PRI_DISP_LIST_RCD  = PACKED RECORD
      PRI_DISP_LIST     : ARRAY[ACT_DISP_TBL.NBR_PRI_DISP_LISTS] OF
                          DISP_LIST_DESC_RCD;
      PRI_DISP_SOURCES  : ARRAY[ACT_DISP_TBL.NBR_PRI_DISP_LIST_ITEMS]
                          OF UINT16;
END;

TABLE PRI_DISP_LIST_TBL = PRI_DISP_LIST_RCD;
```

Identifier	*Value*	*Definition*
DISP_SCROLL1_BFLD		
ON_TIME	0..15	The number of seconds each display item is displayed.
OFF_TIME	0..15	The number of seconds display is blank between display items. 0 = no deliberate OFF Time.
DISP_SCROLL2_BFLD		
HOLD_TIME	0..15	The number of minutes to wait before automatically restarting the normal display scroll after hold has been initiated.
DEFAULT_LIST		Associates this display list with a standard display list type. The “normal” display list scrolling resumes after **HOLD_TIME** has expired. The following values are recognized:
	0	Display selectable via communication link only;
	1	Normal display;
	2	Alternate display;
	3	Test display;
	4..15	Unassigned.
DISP_LIST_DESC_RCD		
DISP_SCROLL1		This contains the **ON_TIME** and **OFF_TIME** values for the associated list.
DISP_SCROLL2		This contains the **HOLD_TIME** and **DEFAULT_LIST** values for the associated list.

NBR_LIST_ITEMS	0..255	Number of items in this display list. The total number of items (list entries) in all display lists of this table shallshould not exceed **ACT_DISP_TBL. NBR_PRI_DISP_LIST_ITEM** (Table 31).
PRI_DISP_LIST_RCD		
PRI_DISP_LIST		An array of list descriptor records.
PRI_DISP_SOURCES		Ordered list of references to display items contained in **DISP_SOURCE_TBL** (Table 32). This list establishes the order of the primary display. This standard does not preclude additional items from popping up or being displayed out of sequence as a result of an end device error condition or an end device status report being generated.

9.4.5 TABLE 34 Secondary display list table

Table 34 Data description

SEC_DISP_LIST_TBL (Table 34) contains a list of operational parameters and references into the table, **DISP_SOURCE_TBL**, (Table 32) necessary for programming the secondary display lists, timing, and scroll sequences.

```
TYPE SEC_DISP_LIST_RCD = PACKED RECORD
      SEC_DISP_LIST     : ARRAY[ACT_DISP_TBL.NBR_SEC_DISP_LISTS] OF
                              PRI_DISP_LIST_TBL.DISP_LIST_DESC_RCD;

      SEC_DISP_SOURCES  : ARRAY[ACT_DISP_TBL.NBR_SEC_DISP_LIST_ITEMS] OF
                              UINT16;
END;

TABLE SEC_DISP_LIST_TBL = SEC_DISP_LIST_RCD;
```

Identifier	*Value*	*Definition*
DISP_SCROLL1_BFLD		
ON_TIME	0..15	The number of seconds each display item is displayed.
OFF_TIME	0..15	The number of seconds display is blank between display items. 0 = no deliberate OFF Time.
DISP_SCROLL2_BFLD		
HOLD_TIME	0..15	The number of minutes to wait before automatically restarting the normal display scroll after hold has been initiated.
DEFAULT_LIST		Associates this display list with a standard display list type. The “normal” display list scrolling resumes after **HOLD_TIME** has expired. The following values are recognized:
	0	Display selectable via communication link only,
	1	Normal display;
	2	Alternate display;
	3	Test display;
	4..15	Unassigned.

DISP_LIST_DESC_RCD		
DISP_SCROLL1		This contains the **ON_TIME** and **OFF_TIME** values for the associated list.
DISP_SCROLL2		This contains the **HOLD_TIME** and **DEFAULT_LIST** values for the associated list.
NBR_LIST_ITEMS	0..255	Number of items in this display list. The total number of items (list entries) in all display lists of this table shall not exceed **ACT_DISP_TBL.NBR_PRI_DISP_LIST_ITEM** (Table 31).
SEC_DISP_LIST_RCD		
SEC_DISP_LIST		An array of list descriptor records.
SEC_DISP_SOURCES		Ordered list of references to display items contained in **DISP_SOURCE_TBL** (Table 32). This list establishes the order of the secondary display. This standard does not preclude additional items from popping up or being displayed out of sequence as a result of an end device error condition or an end device status report being generated.

9.5 DECADE 40: Security tables

The security tables provide holding areas for the placement of end device passwords and encryption/authentication keys used to establish group access permissions. Access permissions are used to limit table read or write access and procedure execute permissions to groups of users based on the interpretation of the password and encryption/authentication fields. The exact means for granting access are not defined by this standard.

9.5.1 TABLE 40 Dimension security limiting table

Table 40 Data description

DIM_SECURITY_LIMITING_TBL (Table 40) defines the maximum number of passwords and security access level entries supported by the end device.

```
TYPE SECURITY_RCD = PACKED RECORD
      NBR_PASSWORDS     : UINT8;
      PASSWORD_LEN      : UINT8;
      NBR_KEYS          : UINT8;
      KEY_LEN           : UINT8;
      NBR_PERM_USED     : UINT16;
END;

TABLE DIM_SECURITY_LIMITING_TBL = SECURITY_RCD;
```

Identifier	*Value*	*Definition*
SECURITY_RCD		
NBR_PASSWORDS	0	Access levels cannot be established using the standard passwords mechanism.

	1..255	Maximum number of passwords this end device is capable of supporting.
PASSWORD_LEN	1..20	Maximum length of passwords in octets this end device is capable of supporting.
	21..255	Invalid.
NBR_KEYS	0..255	Maximum number of keys for authentication/encryption this end device is capable of supporting.
KEY_LEN	0..255	Maximum length of keys in octets this end device is capable of supporting.
NBR_PERM_USED	0..65535	Maximum number of user-defined security access entries end device supports in addition to those defined through default access permissions.

9.5.2 TABLE 41 Actual security limiting table

Table 41 Data description

ACT_SECURITY_LIMITING_TBL (Table 41) defines the actual number of passwords and security access level entries supported by the end device.

```
TABLE ACT_SECURITY_LIMITING_TBL = DIM_SECURITY_LIMITING_TBL.
                                  SECURITY_RCD;
```

Identifier	*Value*	*Definition*
SECURITY_RCD		
NBR_PASSWORDS	0	Actual access levels are not established through the standard passwords mechanism.
	1..255	Actual number of passwords used by this end device.
PASSWORD_LEN	1..20	Actual length of passwords in octets this end device is supporting.
	21..255	Invalid.
NBR_KEYS	0..255	Actual number of keys this end device is supporting.
KEY_LEN	0..255	Actual length of keys this end device is supporting.
NBR_PERM_USED	0..65535	Actual number of user-defined security access entries end device supports in addition to those defined through default access permissions.

9.5.3 TABLE 42 Security table

Table 42 Data description

SECURITY_TBL (Table 42) stores actual end device passwords. It is used to associate passwords with read, write, and procedure execute access permission bits.

Any table or procedure whose access control bits match with respect to position at least one of the group access permissions bits, defined in **SECURITY_TBL** (Table 42) for the supplied password, can become a candidate for data transfers.

NOTES

1—Although it may be possible to read the security table, the values reported (read back) are not defined by this standard.

2—When access permissions are given to a procedure that accesses or modifies table data, then the invoker of that procedure shall in addition be required to have read or write access to the table that is read or modified.

3—When access permissions are given to a user-defined table that accesses or modifies other tables, then the invoker of that user-defined table shall in addition be required to have read or write access to the tables indirectly read or written.

```
TYPE ACCESS_PERMISSION_BFLD = BIT FIELD OF UINT8
      GROUP_PERM_0_FLAG       : BOOL(0);
      GROUP_PERM_1_FLAG       : BOOL(1);
      GROUP_PERM_2_FLAG       : BOOL(2);
      GROUP_PERM_3_FLAG       : BOOL(3);
      GROUP_PERM_4_FLAG       : BOOL(4);
      GROUP_PERM_5_FLAG       : BOOL(5);
      GROUP_PERM_6_FLAG       : BOOL(6);
      GROUP_PERM_7_FLAG       : BOOL(7);
END;

TYPE SECURITY_ENTRY_RCD = PACKED RECORD
      PASSWORD    : ARRAY[ACT_SECURITY_LIMITING_TBL.PASSWORD_LEN20]
                    OF UINT8;
      ACCESS_PERMISSIONS      : ACCESS_PERMISSION_BFLD;
END;

TYPE SECURITY_RCD = PACKED RECORD
      SECURITY_ENTRIES  : ARRAY[ACT_SECURITY_LIMITING_TBL.NBR_PASSWORDS]
                          OF SECURITY_ENTRY_RCD;
END;

TABLE SECURITY_TBL = SECURITY_RCD;
```

Identifier	*Value*	*Definition*
ACCESS_PERMISSIONS_BFLD		
GROUP_PERM_0_FLAG		
	FALSE	Group 0 privileges are not granted by this password.
	TRUE	Group 0 privileges are granted by this password.
GROUP_PERM_1_FLAG		
	FALSE	Group 1 privileges are not granted by this password.
	TRUE	Group 1 privileges are granted by this password.

GROUP_PERM_2_FLAG		
	FALSE	Group 2 privileges are not granted by this password.
	TRUE	Group 2 privileges are granted by this password.
GROUP_PERM_3_FLAG		
	FALSE	Group 3 privileges are not granted by this password.
	TRUE	Group 3 privileges are granted by this password.
GROUP_PERM_4_FLAG		
	FALSE	Group 4 privileges are not granted by this password.
	TRUE	Group 4 privileges are granted by this password.
GROUP_PERM_5_FLAG		
	FALSE	Group 5 privileges are not granted by this password.
	TRUE	Group 5 privileges are granted by this password. Read or write access to Group 5 tables may be granted to this user.
GROUP_PERM_6_FLAG		
	FALSE	Group 6 privileges are not granted by this password.
	TRUE	Group 6 privileges are granted by this password.
GROUP_PERM_7_FLAG		
	FALSE	Group 7 privileges are not granted by this password.
	TRUE	Group 7 privileges are granted by this password.
SECURITY_ENTRY_RCD		
PASSWORD		Password to be matched and be associated with group access permission bits.
ACCESS_PERMISSIONS		Group access permissions bits for this password.
SECURITY_RCD		
SECURITY_ENTRIES		Array containing a list of security limiting parameters.

9.5.4 TABLE 43 Default access control table

Table 43 Data description

DEFAULT_ACCESS_CONTROL_TBL (Table 43) is used to establish default table and procedure access permissions. These permissions are applied to any table that is not included in the **ACCESS_CONTROL_TABLE** (Table 44).

```
TYPE DEFAULT_ACCESS_TABLE_DEF_BFLD = BIT FIELD OF UINT16
      RESERVED          : FILL(0..12);
      ANY_READ_FLAG     : BOOL(13);
      ANY_WRITE_FLAG    : BOOL(14);
      FILL2             : FILL(15..15);
```

```
END;

TYPE DEFAULT_ACCESS_CONTROL_RCD = PACKED RECORD
      ACCESS_TABLE_DEFAULT    : DEFAULT_ACCESS_TABLE_DEF_BFLD;
      READ                    : SECURITY_TBL.ACCESS_PERMISSION_BFLD;
      WRITE                   : SECURITY_TBL.ACCESS_PERMISSION_BFLD;
END;

TYPE DEFAULT_ACCESS_RCD = PACKED RECORD
      TABLE_DEFAULT     : DEFAULT_ACCESS_CONTROL_RCD;
      PROCEDURE_DEFAULT : DEFAULT_ACCESS_CONTROL_RCD;
END;

TABLE DEFAULT_ACCESS_CONTROL_TBL = DEFAULT_ACCESS_RCD ;
```

Identifier	*Value*	*Definition*
DEFAULT_ACCESS_TABLE_DEF_BFLD		
RESERVED		Reserved. See 6.4
ANY_READ_FLAG	FALSE	Any unrecognized user does not have read access permission to any tables.
	TRUE	Any unrecognized user has read access permission to any table not defined in **ACCESS_CONTROL_TBL** (Table 44).
ANY_WRITE_FLAG	FALSE	Any unrecognized user does not have write access permission to any tables.
	TRUE	Any unrecognized user has write access permission to any table not defined in **ACCESS_CONTROL_TBL** (Table 44).
DEFAULT_ACCESS_CONTROL_ENTRY_RCD		
ACCESS_TABLE_DEF		Read and write access permissions for unrecognized users as defined above.
READ		Group security access permission bits for users that have been recognized and request read access to tables that are not listed in the **ACCESS_CONTROL_TBL** (Table 44).
WRITE		Group security access permission bits for users that have been recognized and request write access to tables that are not listed in the **ACCESS_CONTROL_TBL** (Table 44).
DEFAULT_ACCESS_RCD		
TABLE_DEFAULT		Default access control for all tables, except the **PROC_INITIATE_TBL** (Table 07).
PROCEDURE_DEFAULT		Default access control for all procedures using **PROC_INITIATE_TBL** (Table 07).

9.5.5 TABLE 44 Access control table

Table 44 Data description

ACCESS_CONTROL_TBL (Table 44) establishes table and procedure access permissions on a per table and per procedure basis for those tables not using the default access level.

```
TYPE ACCESS_CONTROL_ENTRY_RCD = PACKED RECORD
      ACCESS_TABLE_DEF  : TABLE_IDC_BFLD;
      READ              : ACCESS_PERMISSION_BFLD;
      WRITE             : ACCESS_PERMISSION_BFLD;
END;

TYPE ACCESS_CONTROL_RCD = PACKED RECORD
      ACCESS_CONTROL    : ARRAY[ACT_SECURITY_LIMITING_TBL.NBR_PERM_USED]
                          OF ACCESS_CONTROL_ENTRY_RCD;
END;

TABLE ACCESS_CONTROL_TBL = ACCESS_CONTROL_RCD;
```

Identifier	*Value*	*Definition*
TABLE_IDC_BFLD		
TBL_PROC_NBR	0..2047	Table or procedure number associated with this access permission entry.
STD_VS_MFG_FLAG	FALSE	Entry is a standard table or standard procedure.
	TRUE	Entry is a manufacturer table or manufacturer procedure.
PROC_FLAG	FALSE	Numeric ID is a table number.
	TRUE	Numeric ID is a procedure number.
FLAG1	FALSE	Associated table or procedure requires read access control bits. Only approved group members may access table or procedure.
	TRUE	Associated table or procedure has unrestricted read access. Any user may read the associated table or procedure.
FLAG2	FALSE	Associated table or procedure requires write access control bits. Only approved group members may write to table or procedure.
	TRUE	Associated table or procedure has unrestricted write or execute permission. Any user may write into the associated table or procedure.
FLAG3		Reserved.
ACCESS_CONTROL_ENTRY_RCD		
ACCESS_TABLE_DEF		Table identification and public access control.
READ		Group read access control bits. See **SECURITY_TBL.-**

	ACCESS_PERMISSION_BFLD (Table 42).
WRITE	Group write access control bits. See **SECURITY_TBL.-ACCESS_PERMISSION_BFLD** (Table 42).
ACCESS_CONTROL_RCD	
ACCESS_CONTROL	Array containing access permission for specific table or procedure.

9.5.6 TABLE 45 Key table

Table 45 Data description

KEY_TBL (Table 45) is a table established to contain authentication and/or encryption keys.

NOTE—Although it may be possible to read the **KEY_TBL** (Table 45), the values reported (read back) are not defined by this standard.

```
TYPE KEY_ENTRY_RCD = PACKED RECORD
      KEY   : ARRAY[ ACT_SECURITY_LIMITING_TBL.KEY_LEN] OF UINT8;
END;

TYPE KEY_RCD = PACKED RECORD
      KEY_ENTRIES : ARRAY[ACT_SECURITY_LIMITING_TBL.NBR_KEYS] OF
                        KEY_ENTRY_RCD;
END;

TABLE KEY_TBL = KEY_RCD;
```

Identifier	*Value*	*Definition*
KEY_ENTRY_RCD		
KEY		A key to be applied in the authentication and/or encryption processes. Use of any key is not defined by this standard.
KEY_RCD		
KEY_ENTRIES		Array of keys used to establish authentication and/or encryption.

9.6 DECADE 50: TOU tables

The tables in this decade provide information related to end device date and time and TOU operation.

9.6.1 TABLE 50 Dimension time and TOU table

Table 50 Data description

DIM_TIME_TOU_TBL (Table 50) defines the maximum capabilities for date and time clock control and TOU control for the end device.

```
TYPE TIME_FUNC_FLAG1_BFLD = BIT FIELD OF UINT8
      TOU_SELF_READ_FLAG            : BOOL(0);
      SEASON_SELF_READ_FLAG         : BOOL(1);
      SEASON_DEMAND_RESET_FLAG      : BOOL(2);
      SEASON_CHNG_ARMED_FLAG        : BOOL(3);
      SORT_DATES_FLAG               : BOOL(4);
      ANCHOR_DATE_FLAG              : BOOL(5);
      FILLER                        : FILL(6..7);
END;

TYPE TIME_FUNC_FLAG2_BFLD = BIT FIELD OF UINT8
      CAP_DST_AUTO_FLAG             : BOOL(0);
      SEPARATE_WEEKDAYS_FLAG        : BOOL(1);
      SEPARATE_SUM_DEMANDS_FLAG     : BOOL(2);
      SORT_TIER_SWITCHES_FLAG       : BOOL(3);
      CAP_TM_ZN_OFFSET_FLAG         : BOOL(4);
      FILLER                        : FILL(5..7);
END;

TYPE CALENDAR_BFLD = BIT FIELD OF UINT8
      NBR_SEASONS                   : UINT(0..3);
      NBR_SPECIAL_SCHED             : UINT(4..7);
END;

TYPE TIME_TOU_RCD = PACKED RECORD
      TIME_FUNC_FLAG1               : TIME_FUNC_FLAG1_BFLD;
      TIME_FUNC_FLAG2               : TIME_FUNC_FLAG2_BFLD;
      CALENDAR_FUNC                 : CALENDAR_BFLD;
      NBR_NON_RECURR_DATES          : UINT8;
      NBR_RECURR_DATES              : UINT8;
      NBR_TIER_SWITCHES             : UINT16;
      CALENDAR_TBL_SIZE             : UINT16;
END;

TABLE DIM_TIME_TOU_TBL = TIME_TOU_RCD;
```

Identifier	*Value*	*Definition*
TIME_FUNC_FLAG1_BFLD		
TOU_SELF_READ_FLAG	FALSE	End device is not capable of TOU triggered self-reads.
	TRUE	End device is capable of TOU triggered self-reads.
SEASON_SELF_READ_FLAG	FALSE	Self read with season change capability does not exist.

	TRUE	Self read with season change capability exists.
SEASON_DEMAND_RESET_FLAG		
	FALSE	Automatic demand reset with season change capability does not exist.
	TRUE	Automatic demand reset with season change capability exists.
SEASON_CHNG_ARMED_FLAG	FALSE	Season change arming capability does not exist.
	TRUE	Season change arming capability exists. That is, the end device is only armed to do a season change as a result of an indicated change of season. The actual season change does not occur until the next demand reset occurs (by manual or other means).
SORT_DATES_FLAG	FALSE	End device does not require the **CALENDAR_TBL.NON_RECURR_DATES** (Table 54) be presorted by date (Year, Month, Day) when received.
	TRUE	End device does require the **CALENDAR_TBL.NON_RECURR_DATES** (Table 54) be presorted by date (Year, Month, Day) when received. Non-recreoccurring dates starting with 1990 and ending with 2089 shall occur in that order. No sorting of recurring dates, if present, is assumed.
ANCHOR_DATE_FLAG	FALSE	End device is not capable of accepting an anchor date for the period/delta **RDATE** type (recurring date).
	TRUE	End device is capable of accepting an anchor date for the Period/Delta **RDATE** type (recurring date).

TIME_FUNC_FLAG2_BFLD		
CAP_DST_AUTO_FLAG	FALSE	End device is not capable of handling DST changes independently of dates supplied by the **CALENDAR_TBL** (Table 54).
	TRUE	End device is capable of handling DST changes independently of dates supplied by the **CALENDAR_TBL** (Table 54).
SEPARATE_WEEKDAYS_FLAG	FALSE	End device is not capable of having a different schedule for each of the five weekdays.
	TRUE	End device is capable of having a different schedule for each of the five weekdays.
SEPARATE_SUM_DEMANDS_FLAG		
	FALSE	End device is not capable of switching summation and demands independently.
	TRUE	End device is capable of switching summation and demands independently.
SORT_TIER_SWITCHES_FLAG	FALSE	End device does not require that **CALENDAR_TBL.TIER_SWITCHES** (Table 54) be presorted.
	TRUE	End device requires that **CALENDAR_TBL.TIER_SWITCHES** (Table 54) be presorted.
CAP_TM_ZN_OFFSET_FLAG	FALSE	Time zone offset capability is not supported by the end device.
	TRUE	Time zone offset capability is supported by the end device.
CALENDAR_BFLD		
NBR_SEASONS	0..15	Maximum number of seasons supported by this end device.
NBR_SPECIAL_SCHED	0..11	Maximum number of special schedules supported by this end device.
	12..15	Reserved.
TIME_TOU_RCD		
TIME_FUNC_FLAG1		See **TIME_FUNC_FLAG1_BFLD** above.
TIME_FUNC_FLAG2		See **TIME_FUNC_FLAG2_BFLD** above.
CALENDAR_FUNC		See **CALENDAR_BFLD** above.

NBR_NON_RECURR_DATES	0..255	Maximum number of non-reoccurring dates supported by the end device calendar. Each entry allows the control of one or more events like demand reset, self read, DST, season change and special schedule.
NBR_RECURR_DATES	0..255	Maximum number of reoccurring dates supported by the end device calendar. Each entry allows the definition of a recurring event like demand reset, self read, DST, season change and special schedule.
NBR_TIER_SWITCHES	0..65535	Maximum number of tier switches supported by the end device calendar. This number comprises the total collection of tier switches for all day types in the **CALENDAR_TBL.-DAILY_SCHEDULE_ID_MATRIX** (Table 54).
CALENDAR_TBL_SIZE	0..65535	Maximum octets of data transported by **CALENDAR_TBL** (Table 54). This includes recurring dates, non-recurring dates and tier switches. Anchor date is counted even if it is unused.

9.6.2 TABLE 51 Actual time and TOU table

Table 51 Data Description

ACT_TIME_TOU_TBL (Table 51) defines the actual capabilities for date and time clock control limits and TOU control limits for the end device. The data structure of this table is identical to **DIM_TIME_TOU_TBL** (Table 50). Elements are defined below.

TABLE ACT_TIME_TOU_TBL = DIM_TIME_TOU_TBL.TIME_TOU_RCD;

Identifier	*Value*	*Definition*
TIME_FUNC_FLAG1_BFLD		
TOU_SELF_READ_FLAG	FALSE	End device is not using TOU triggered self-reads.
	TRUE	End device is using TOU triggered self-reads.
SEASON_SELF_READ_FLAG	FALSE	Self read with season change capability is not being used.
	TRUE	Self read with season change capability is being used.
SEASON_DEMAND_RESET_FLAG		
	FALSE	Automatic demand reset when season change occurs is not in use.

	TRUE	Automatic demand reset when season change occurs is being used.
SEASON_CHNG_ARMED_FLAG		
	FALSE	Season change arming is not being used.
	TRUE	Season change arming is in use. That is, the end device is only armed to do a season change as a result of an indicated change of season. The actual season change does not occur until the next demand reset occurs (by manual or other means).
SORT_DATES_FLAG	FALSE	End device may or may not require the **CALENDAR_TBL.NON_RECURR_DATES** (Table 54) to be presorted by date (Year, Month, Day) when received.
	TRUE	End device does require the **CALENDAR_TBL.NON_RECURR_DATES** (Table 54) be presorted by date (Year, Month, Day) when received. Non-reoccurring dates starting with 1990 and ending with 2089 shallmust be sorted in that order. No sorting of reoccurring dates, if present, is required.
ANCHOR_DATE_FLAG	FALSE	End device will not accept an anchor date for the period/delta **RDATE** type (recurring date).
	TRUE	End device will accept an anchor date for the Period/Delta **RDATE** type (recurring date).
TIME_FUNC_FLAG2_BFLD		
CAP_DST_AUTO_FLAG	FALSE	End device is not handling DST changes independently of dates supplied by the **CALENDAR_TBL** (Table 54).
	TRUE	End device is handling DST changes independently of dates supplied by the **CALENDAR_TBL** (Table 54).
SEPARATE_WEEKDAYS_FLAG		
	FALSE	End device is not using a different schedule for each of the five weekdays.
	TRUE	End device is using a different schedule for each of the five weekdays.
SEPARATE_SUM_DEMANDS_FLAG		
	FALSE	End device is not switching summation and demands independently.
	TRUE	End device is switching summation and demands independently.

SORT_TIER_SWITCHES_FLAG		
	FALSE	End device does not require that **CALENDAR_TBL.TIER_SWITCHES** (Table 54) be presorted.
	TRUE	End device requires that **CALENDAR_TBL.TIER_SWITCHES** (Table 54) be presorted.
CAP_TM_ZN_OFFSET_FLAG	FALSE	Time zone offset capability is not available in the end device.
	TRUE	Time zone offset capability is available in the end device.
CALENDAR_BFLD		
NBR_SEASONS	0..15	Actual number of seasons in use by the end device.
NBR_SPECIAL_SCHED	0..11	Actual number of special schedules in use by the end device.
	12..15	Reserved.
TIME_TOU_RCD		
TIME_FUNC_FLAG1		See **TIME_FUNC_FLAG1_BFLD** above.
TIME_FUNC_FLAG2		See **TIME_FUNC_FLAG2_BFLD** above.
CALENDAR_FUNC		See **CALENDAR_BFLD** above.
NBR_NON_RECURR_DATES		
	0..255	Actual number of non-reoccurring dates supported by the end device calendar. Each entry allows the control of one or more events like demand reset, self read, daylight savings time, season change, and special schedule.
NBR_RECURR_DATES	0..255	Actual number of reoccurring dates supported by the end device calendar. Each entry allows the definition of a reoccurring event like demand reset, self read, daylight savings time, season change, and special schedule.
NBR_TIER_SWITCHES	0..65535	Actual number of tier switches supported by the end device calendar. This number comprises the total collection of tier switches for all day types in the **CALENDAR_TBL.-DAILY_SCHEDULE_ID_MATRIX** (Table 54).
CALENDAR_TBL_SIZE	0..65535	Actual octets of data transported by **CALENDAR_TBL** (Table 54). This includes the reoccurring dates, non-reoccurring dates and tier switches.

9.6.3 TABLE 52 Clock table

Table 52 Data description

CLOCK_ TBL (Table 52) provides the end device real time clock information.

```
TYPE TIME_DATE_QUAL_BFLD      = BIT FIELD OF UINT8
      DAY_OF_WEEK             : UINT(0..2);
      DST_FLAG                : BOOL(3);
      GMT_FLAG                : BOOL(4);
      TM_ZN_APPLIED_FLAG      : BOOL(5);
      DST_APPLIED_FLAG        : BOOL(6);
      FILLER                  : FILL(7..7);
END;

TYPE CLOCK_STATE_RCD = PACKED RECORD
      CLOCK_CALENDAR    : LTIME_DATE;
      TIME_DATE_QUAL    : TIME_DATE_QUAL_BFLD;
END;

TABLE CLOCK_TBL = CLOCK_STATE_RCD;
```

Identifier	*Value*	*Definition*
TIME_DATE_QUAL_BFLD		Status qualifying the end device time
DAY_OF_WEEK		Current day of the week
	0	Sunday
	1	Monday
	2	Tuesday
	3	Wednesday
	4	Thursday
	5	Friday
	6	Saturday
	7	Unassigned
DST_FLAG		DST status
	FALSE	End device time is not in DST
	TRUE	End device time is in DST
GMT_FLAG	FALSE	End device time does not correspond to GMT
	TRUE	End device time corresponds to GMT
TM_ZN_APPLIED_FLAG		
	FALSE	Time zone offset has not been applied to the end device time
	TRUE	Time zone offset has been applied to the end device time
DST_APPLIED_FLAG		
	FALSE	End device time does not include daylight savings adjustment
	TRUE	End device time includes daylight savings adjustment
CLOCK_STATE_RCD		

CLOCK_CALENDAR	Current end device time
TIME_DATE_QUAL	See **DATE_TIME_QUAL_BFLD** above

9.6.4 TABLE 53 Time offset table

Table 53 Data description

TIME_OFFSET_TBL (Table 53) provides time zone offset and DST information for the end device.

```
TYPE TIME_OFFSET_RCD = PACKED RECORD

      DST_TIME_EFF            : TIME;
      DST_TIME_AMT            : UINT8;
      IF ACT_TIME_TOU_TBL.CAP_TM_ZN_OFFSET_FLAG THEN
            TIME_ZONE_OFFSET  : INT16;
      END;
END;

TABLE TIME_OFFSET_TBL = TIME_OFFSET_RCD;
```

Identifier	*Value*	*Definition*
TIME_OFFSET_RCD		
DST_TIME_EFF		The time of day the end device will shifts to or from DST. The date of occurrence is based on entries in the **CALENDAR_TBL** (Table 54) and the **ACT_TIME_TOU_TBL.CAP_DST_AUTO_FLAG** (Table 51) flag.
DST_TIME_AMT	0..255	End device DST adjustment specified in minutes.
TIME_ZONE_OFFSET	–32767..32767	Offset for end device, representing the number of ± 1min segments of end device time offset.

9.6.5 TABLE 54 Calendar table

Table 54 Data description

CALENDAR_TBL (Table 54) is the schedule definition table for TOU metering activities.

```
TYPE CALENDAR_ACTION_BFLD = BIT FIELD OF UINT8
      CALENDAR_CTRL      : UINT(0..4);
      DEMAND_RESET_FLAG  : BOOL(5);
      SELF_READ_FLAG     : BOOL(6);
      FILLER             : FILL(7..7);
END;

TYPE RECURR_DATE_RCD = PACKED RECORD
      RECURR_DATE        : RDATE;
      CALENDAR_ACTION    : CALENDAR_ACTION_BFLD;
END;

TYPE NON_RECURR_DATE_RCD = PACKED RECORD
```

```
      NON_RECURR_DATE   : DATE;
      CALENDAR_ACTION   : CALENDAR_ACTION_BFLD;
END;
TYPE TIER_SWITCH_BFLD = BIT FIELD OF UINT16
      NEW_TIER                      : UINT(0..2);
      IF ACT_TIME_TOU_TBL.SEPARATE_SUM_DEMANDS_FLAG THEN
            SUMMATION_SWITCH_FLAG   : BOOL(3);
            DEMANDS_SWITCH_FLAG     : BOOL(4);
      ELSE
            FILLER                  : FILL(3..4);
      END;
      SWITCH_MIN                    : UINT(5..10);
      SWITCH_HOUR                   : UINT(11..15);
END;

TYPE TIER_SWITCH_RCD = PACKED RECORD
      TIER_SWITCH                   : TIER_SWITCH_BFLD;
      DAY_SCH_NUM                   : UINT8;
END;

TYPE CALENDAR_RCD = PACKED RECORD
      IF ACT_TIME_TOU_TBL.ANCHOR_DATE_FLAG THEN
            ANCHOR_DATE             : DATE;
      END;
      NON_RECURR_DATES  : ARRAY[ACT_TIME_TOU_TBL.NBR_NON_RECURR_DATES]
                              OF NON_RECURR_DATE_RCD;
      RECURR_DATES      : ARRAY[ACT_TIME_TOU_TBL.NBR_RECURR_DATES] OF
                              RECURR_DATE_RCD;
      TIER_SWITCHES     : ARRAY[ACT_TIME_TOU_TBL.NBR_TIER_SWITCHES] OF
                              TIER_SWITCH_RCD;
      IF ACT_TIME_TOU_TBL.SEPARATE_WEEKDAYS_FLAG THEN
      DAILY_SCHEDULE_ID_MATRIX : ARRAY[ACT_TIME_TOU_TBL.NBR_SEASONS,
                                 ACT_TIME_TOU_TBL.NBR_SPECIAL_SCHED+7]
                                    OF UINT8;
      ELSE
      DAILY_SCHEDULE_ID_MATRIX : ARRAY[ACT_TIME_TOU_TBL.NBR_SEASONS,
                                ACT_TIME_TOU_TBL.NBR_SPECIAL_SCHED+3]
                                    OF UINT8;
      END;
END;

TABLE CALENDAR_TBL = CALENDAR_RCD;
```

Identifier	*Value*	*Definition*
CALENDAR_ACTION_BFLD		
CALENDAR_CTRL:		Number specifying the calendar action that will takes place on the date specified. The actions and their codes are defined as follows:
	0	No action
	1	DST on
	2	DST off
	3	Select season 0
	4	Select season 1
	.	
	.	
	.	

18 Select season 15
19 Special schedule 0
20 Special schedule 1
.
.
.
30 Special schedule 11
31 End of list

DST on/off
DST on and off actions causes a time adjustment to be made at **TIME_OFFSET_TBL.DST_TIME_EFF** (Table 53) time. If **TIME_OFFSET_TBL** (Table 53) is not supported, the time adjustment is made at 02:00 A.M. and the effectivity amount equals 1 h.

If **ACT_TIME_TOU_TBL.CAP_DST_AUTO_FLAG** (Table 51) = TRUE then DST on/off actions in this table are ignored.

Season changes
If **ACT_TIME_TOU_TBL.-SEASON_CHNG_ARMED_FLAG** (Table 51) = FALSE, a season change action will causes a season change to occur at 00:00 A.M. hours on the date specified. If **ACT_TIME_TOU_TBL.-SEASON_CHNG_ARMED_FLAG** (Table 51) = TRUE, a season change action will “arm” a season change, but the season change will not occur until the next demand reset.

If **ACT_TIME_TOU_TBL.-SEASON_DEMAND_RESET_FLAG** (Table 51) = TRUE, a season change will causes a demand reset to occur automatically.

A self read of the current register data is saved in the **SELF_READ_DATA_TBL** (Table 26) if **ACT_TIME_TOU_TBL.SEASON_SELF_READ_FLAG** (Table 51) = TRUE when the season change occurs.

If **GEN_CONFIG_TBL.STD_TBLS_USED.-PREVIOUS_SEASON_DATA_TBL** (Table 00) = TRUE, a season change will causes a copy of the current register data to be saved in the **PREVIOUS_SEASON_DATA_TBL** (Table 24)

		Special schedules Special schedules are used for holidays or other special daily schedules. These schedules take effect at midnight (00:00 A.M.) and end 24 h later (23:59 P.M.). The special schedule takes precedence over the normal day of week schedule. End of list End of list action is to notify the utility that the **CALENDAR_TBL** (Table 54) entries are approaching expiration. Notification is by history message, display, or other means defined by the device manufacturer.
DEMAND_RESET_FLAG	FALSE TRUE	Demand reset shall not occur. Demand reset shall occur at 00:00 A.M. on the date specified. If **GEN_CONFIG_TBL.STD_TBLS_USED.-PREVIOUS_DEMAND_RESET_DATA_TBL** (Table 00) = TRUE, a demand reset will causes data to be saved in **PREVIOUS_DEMAND_RESET_DATA_TBL** (Table 25). If **ACT_REGS_TBL.-SELF_READ_DEMAND_RESET_FLAG** (Table 21) = 1 or 3, a self read shall occur in conjunction with the demand reset.
SELF_READ_FLAG	FALSE TRUE	Self read shall not occur. Self read shall occur at 00:00 A.M. on the date specified. If **ACT_REGS_TBL.-SELF_READ_DEMAND_RESET_FLAG** (Table 21) = 2 or 3, a demand reset shall occur in conjunction with the self read.
RECURR_DATE_RCD		
RECURR_DATE		Date definition of a reoccurring event. The reoccurrence can be yearly, monthly, or weekly. The reoccurring event is referenced to midnight.
CALENDAR_ACTION		Defined action that shall take place on this date. See **CALENDAR_ACTION_BFLD** above.
NON_RECURR_DATE_RCD		
NON_RECURR_DATE		Non-reoccurring event date. Action shall occur at midnight.
CALENDAR_ACTION		Defined action that shall take place on this date. See **CALENDAR_ACTION_BFLD** above.
TIER_SWITCH_BFLD		
NEW_TIER	0..7	Identifies the tier number that will take becomes effective at the time specified in the tier switch on the daily schedule type defined by the **DAILY_SCHEDULE_ID_MATRIX**.

SUMMATION_SWITCH_FLAG		Specifies whether or not this tier switch selects a new tier for summations. Only used when **ACT_TIME_TOU_TBL.-SEPARATE_SUM_DEMANDS_FLAG** (Table 51) = TRUE.
	FALSE	Tier switch is not specified for summations.
	TRUE	Tier switch is specified for summations.
DEMANDS_SWITCH_FLAG		Specifies whether or not this tier switch selects a new tier for demands. Only used when **ACT_TIME_TOU_TBL.-SEPARATE_SUM_DEMANDS_FLAG** (Table 51) = TRUE.
	FALSE	Tier switch is not specified for demands.
	TRUE	Tier switch is specified for demands.
SWITCH_MIN	0..59	Minute when the tier switch will occur
	60..63	Invalid.
SWITCH_HOUR	0..23	Hour when the tier switch will occur
	24..31	Invalid.
TIER_SWITCH_RCD		
TIER_SWITCH		Defines a switch point for a certain day type defined by the **DAILY_SCHEDULE_ID_MATRIX**.
DAY_SCH_NUM	0..255	Specifies the daily schedule type associated with the tier switch. All identifiers are stored in the **DAILY_SCHEDULE_ID_MATRIX**.
CALENDAR_RCD		
ANCHOR_DATE		Anchor date is only used in conjunction with an **RDATE TYPE** reoccurring date using the **PERIOD/OFFSET** mechanism. See 6.3.2. If specified, any reoccurring date using the **PERIOD/OFFSET** type shall use this date as a starting date.
NON_RECURR_DATES		Array containing non-reoccurring dates.
RECURR_DATES		Array containing reocurring dates.
TIER_SWITCHES		Array that contains the tier switches.
DAILY_SCHEDULE_ID_MATRIX		Two-dimensional array containing daily schedule type identifiers. The array is bounded by [seasons supported * (day types supported + special schedules supported)]. Each entry in the array consists of a single octet type identifier value. The identifier is used to tie tier switch entries to specific daily schedules. An example of a daily schedule is special schedule 1 in season 2, or Saturday day-type in season 1. The assignment of identifier codes is left to the utility's discretion.

9.6.6 TABLE 55 Clock state table

Table 55 Data description

CLOCK_STATE_TBL (Table 55) provides the end device real time clock information.

```
TYPE TIME_DATE_QUAL_BFLD = BIT FIELD OF UINT8
      DAY_OF_WEEK        : UINT(0..2);
      DST_FLAG           : BOOL(3);
      GMT_FLAG           : BOOL(4);
      TM_ZN_APPLIED_FLAG: BOOL(5);
      DST_APPLIED_FLAG   : BOOL(6);
      FILLER             : FILL(7..7);
END;

TYPE STATUS_BFLD = BIT FIELD OF UINT16
      IF ACT_TIME_TOU_TBL.SEPARATE_SUM_DEMANDS_FLAG THEN
            CURR_SUMM_TIER    : UINT(0..2);
            CURR_DEMAND_TIER  : UINT(3..5);
   ELSE
            CURR_TIER         : UINT(0..2);
            FILLER            : FILL(3..5);
      END;
      TIER_DRIVE              : UINT(6..7);
      SPECIAL_SCHD_ACTIVE     : UINT(8..11);
      SEASON                  : UINT(12..15);
END;

TYPE CLOCK_STATE_RCD = PACKED RECORD
      CLOCK_CALENDAR    : LTIME_DATE;
      TIME_DATE_QUAL    : CLOCK_TBL.TIME_DATE_QUAL_BFLD;
      STATUS            : STATUS_BFLD;
END;

TABLE CLOCK_STATE_TBL = CLOCK_STATE_RCD;
```

Identifier	*Value*	*Definition*
TIME_DATE_QUAL_BFLD		Status qualifying the end device time
DAY_OF_WEEK		Current day of the week
	0	Sunday
	1	Monday
	2	Tuesday
	3	Wednesday
	4	Thursday
	5	Friday
	6	Saturday
	7	Unassigned
DST_FLAG		DST status
	FALSE	End device time is not DST
	TRUE	End device time is DST
GMT_FLAG	FALSE	End device time does not correspond to GMT

	TRUE	End device time corresponds to GMT
TM_ZN_APPLIED_FLAG		
	FALSE	Time zone offset has not been applied to the end device time
	TRUE	Time zone offset has been applied to the end device time
DST_APPLIED_FLAG		
	FALSE	End device time does not include daylight savings adjustment.
	TRUE	End device time includes daylight savings adjustment.
STATUS_BFLD		Array of status entries of each TOU set. Each set contains:
CURR_SUMM_TIER		
	0..7	Active tier corresponding to summations. This variable is only used when the capability flag **ACT_TIME_TOU_TBL.-SEPARATE_SUM_DEMANDS_FLAG** (Table 51) = TRUE.
CURR_DEMAND_TIER		
	0..7	Active tier corresponding to demands. This variable is only used when the capability flag **ACT_TIME_TOU_TBL.-SEPARATE_SUM_DEMANDS_FLAG** (Table 51) = TRUE.
CURR_TIER	0..7	Number representing the tier that is currently active in the meter. This variable is only used when the capability flag **ACT_TIME_TOU_TBL.-SEPARATE_SUM_DEMANDS_FLAG** (Table 51) = FALSE.
TIER_DRIVE		Tier drive source code
	0	Tier selection is controlled by **CALENDAR_TBL.TIER_SWITCHES** (Table 54).
	1..3	Tier selection is not controlled by this standard
SPECIAL_SCHD_ACTIVE		
	0..11	Active special schedule number
	12..14	Reserved
	15	No special schedule active
SEASON	0..15	Current end device season number
CLOCK_STATE_RCD		
CLOCK_CALENDAR		Current end device time
TIME_DATE_QUAL		See **TIME_DATE_QUAL_BFLD** above
STATUS		See **STATUS_BFLD** above

9.6.7 TABLE 56 Time remaining table

Table 56 Data description

TIME_REMAIN_TBL (Table 56) provides predictive time quantities.

```
TYPE TIME_REMAIN_RCD = PACKED RECORD
      IF ACT_TIME_TOU_TBL.SEPARATE_SUM_DEMANDS_FLAG THEN
            SUMM_TIER_TIME_REMAIN   : UINT16;
            DEMAND_TIER_TIME_REMAIN : UINT16;
      ELSE
            TIER_TIME_REMAIN        : UINT16;
      END;
      SELF_READ_DAYS_REMAIN         : UINT8;
END;

TABLE TIME_REMAIN_TBL = TIME_REMAIN_RCD;
```

Identifier	*Value*	*Definition*
ITIME_REMAIN_RCD		
SUMM_TIER_TIME_REMAIN	0..65535	Minutes remaining until the next tier change that affects summations. Used when the capability flag **ACT_TIME_TOU_TBL.-SEPARATE_SUM_DEMANDS_FLAG** (Table 51) = TRUE.
DEMAND_TIER_TIME_REMAIN	0..65535	Minutes remaining before the next tier change that affects demands. Used when the capability flag **ACT_TIME_TOU_TBL.-SEPARATE_SUM_DEMANDS_FLAG** (Table 51) = TRUE.
TIER_TIME_REMAIN	0..65535	Time in minutes remaining before the next tier change
SELF_READ_DAYS_REMAIN	0..255	Days remaining before the next scheduled self read

9.7 DECADE 60: Load profile tables

These tables provide structures for load profile data.

9.7.1 TABLE 60 Dimension load profile table

Table 60 Data description

DIM_LP_TBL (Table 60) contains function limiting constants for the load profile application. These parameters provide for up to four independent sets of load profile data.

```
TYPE LP_FLAGS_BFLD = BIT FIELD OF UINT168
      LP_SET1_INHIBIT_OVF_FLAG      : BOOL(0);
      LP_SET2_INHIBIT_OVF_FLAG      : BOOL(1);
      LP_SET3_INHIBIT_OVF_FLAG      : BOOL(2);
      LP_SET4_INHIBIT_OVF_FLAG      : BOOL(3);
      BLK_END_READ_FLAG             : BOOL(4);
      BLK_END_PULSE_FLAG            : BOOL(5);
      SCALAR_DIVISOR_FLAG_SET1      : BOOL(6);
      SCALAR_DIVISOR_FLAG_SET2      : BOOL(7);
      SCALAR_DIVISOR_FLAG_SET3      : BOOL(8);
      SCALAR_DIVISOR_FLAG_SET4      : BOOL(9);
      EXTENDED_INT_STATUS_FLAG      : BOOL(10);
      SIMPLE_INT_STATUS_FLAG        : BOOL(11);
      FILLER                        : BOOL(12..15);
      END;

TYPE LP_FMATS_BFLD = BIT FIELD OF UINT8
      INV_UINT8_FLAG                : BOOL(0);
      INV_UINT16_FLAG               : BOOL(1);
      INV_UINT32_FLAG               : BOOL(2);
      INV_INT8_FLAG                 : BOOL(3);
      INV_INT16_FLAG                : BOOL(4);
      INV_INT32_FLAG                : BOOL(5);
      INV_NI_FMAT1_FLAG             : BOOL(6);
      INV_NI_FMAT2_FLAG             : BOOL(7);
END;

TYPE LP_SET_RCD = PACKED RECORD
      LP_MEMORY_LEN                 : UINT32;
      LP_FLAGS                      : LP_FLAGS_BFLD;
      LP_FMATS                      : LP_FMATS_BFLD;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET1_TBL_CNST THEN
            NBR_BLKS_SET1           : UINT16;
            NBR_BLK_INTS_SET1       : UINT16;
            NBR_CHNS_SET1           : UINT8;
            MAX_INT_TIME_SET1       : UINNIT8;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET2_TBL_CNST THEN
            NBR_BLKS_SET2           : UINT16;
            NBR_BLK_INTS_SET2       : UINT16;
            NBR_CHNS_SET2           : UINT8;
            MAX_INT_TIME_SET2       : UINNIT8;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET3_TBL_CNST THEN
            NBR_BLKS_SET3           : UINT16;
            NBR_BLK_INTS_SET3       : UINT16;
            NBR_CHNS_SET3           : UINT8;
            MAX_INT_TIME_SET3       : UINNIT8;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET4_TBL_CNST THEN
            NBR_BLKS_SET4           : UINT16;
            NBR_BLK_INTS_SET4       : UINT16;
            NBR_CHNS_SET4           : UINT8;
            MAX_INT_TIME_SET4       : UINNIT8;
      END;
END;

TABLE DIM_LP_TBL = LP_SET_RCD;
```

Identifier	*Value*	*Definition*
LP_FLAGS_BFLD		This bit field specifies load profile list management capabilities.
LP_SET1_INHIBIT_OVF_FLAG		
	FALSE	End device is not capable of inhibiting load profile set 1 once an overflow occurs.
	TRUE	End device is capable of inhibiting load profile set 1 once an overflow occurs.
LP_SET2_INHIBIT_OVF_FLAG		
	FALSE	End device is not capable of inhibiting load profile set 2 once an overflow occurs.
	TRUE	End device is capable of inhibiting load profile set 2 once an overflow occurs.
LP_SET3_INHIBIT_OVF_FLAG		
	FALSE	End device is not capable of inhibiting load profile set 3 once an overflow occurs.
	TRUE	End device is capable of inhibiting load profile set 3 once an overflow occurs.
LP_SET4_INHIBIT_OVF_FLAG		
	FALSE	End device is not capable of inhibiting load profile set 4 once an overflow occurs.
	TRUE	End device is capable of inhibiting load profile set 4 once an overflow occurs.
BLK_END_READ_FLAG		
	FALSE	End device is not capable of providing block register reading information.
	TRUE	End device is capable of providing block register reading information.
BLK_END_PULSE_FLAG		
	FALSE	End device is not capable of having a running pulse accumulator.
	TRUE	End device is capable of having a running pulse accumulator.
SCALAR_DIVISOR_FLAG_SET1		
	FALSE	End device is not capable of having scalars and divisors associated with set 1 load profile interval data.
	TRUE	End device is capable of having scalars and divisors associated with set 1 load profile interval data.

SCALAR_DIVISOR_FLAG_SET2

FALSE	End device is not capable of having scalars and divisors associated with set 2 load profile interval data.
TRUE	End device is capable of having scalars and divisors associated with set 2 load profile interval data.

SCALAR_DIVISOR_FLAG_SET3

FALSE	End device is not capable of having scalars and divisors associated with set 3 load profile interval data.
TRUE	End device is capable of having scalars and divisors associated with set 3 load profile interval data.

SCALAR_DIVISOR_FLAG_SET4

FALSE	End device is not capable of having scalars and divisors associated with set 4 load profile interval data.
TRUE	End device is capable of having scalars and divisors associated with set 4 load profile interval data.

EXTENDED_INT_STATUS_FLAG

FALSE	End device is not capable of returning extended interval status with load profile interval data.
TRUE	The end device is capable of returning extended interval status with load profile interval data.

SIMPLE_INT_STATUS_FLAG

FALSE	End device is not capable of returning simple interval status with load profile interval data.
TRUE	The end device is capable of returning simple interval status with load profile interval data.

LP_FMATS_BFLD — This set of booleans specifies the format(s) acceptable for use in **DIM_LP_TBL** (Table 60) and **ACT_LP_TBL** (Table 61).

INV_UINT8_FLAG

FALSE	**UINT8** format for intervals is not capable of being used.
TRUE	An interval format of **UINT8** is capable of being used.

INV_UINT16_FLAG

FALSE	**UINT16** format for intervals is not capable of being used.
TRUE	An interval format of **UINT16** is capable of being used.

INV_UINT32_FLAG	FALSE	**UINT32** format for intervals is not capable of being used.
	TRUE	An interval format of **UINT32** is capable of being used.
INV_INT8_FLAG	FALSE	**INT8** format for intervals is not capable of being used.
	TRUE	An interval format of **INT8** is capable of being used.
INV_INT16_FLAG	FALSE	**INT16** format for intervals is not capable of being used.
	TRUE	An interval format of **INT16** is capable of being used.
INV_INT32_FLAG	FALSE	**INT32** format for intervals is not capable of being used.
	TRUE	An interval format of **INT32** is capable of being used.
INV_NI_FMAT1_FLAG		
	FALSE	**NI_FMAT1** format for intervals is not capable of being used.
	TRUE	An interval format of **NI_FMAT1** is capable of being used.
INV_NI_FMAT2_FLAG		
	FALSE	**NI_FMAT2** format for intervals is not capable of being used.
	TRUE	An interval format of **NI_FMAT2** is capable of being used.
LP_SET_RCD		The information defining how each set of load profile information will works.
LP_MEMORY_LEN	0..4 294 967 295	The maximum number of octets of storage available for load profile data. This reflects the combined sizes of tables **LP_DATA_SET1_TBL** (Table 64), **LP_DATA_SET2_TBL** (Table 65), **LP_DATA_SET3_TBL** (Table 66), and **LP_DATA_SET4_TBL** (Table 67).
LP_FLAGS		See **LP_FLAGS_BFLD** above.
LP_FMATS		See **LP_FMATS_BFLD** above.
NBR_BLKS_SET1	0..65535	The maximum number of interval data blocks that can be contained in **LP_DATA_SET1_TBL** (Table 64).

NBR_BLK_INTS_SET1	0..65535	The maximum number of intervals per data block that can be contained in **LP_DATA_SET1_TBL** (Table 64).
NBR_CHNS_SET1	0..255	The maximum number of channels of load profile data that can be contained in **LP_DATA_SET1_TBL** (Table 64).
MAX_INT_TIME_SET1	0..255	The maximum time in minutes for load profile interval duration that can be contained in **LP_DATA_SET1_TBL** (Table 64).
NBR_BLKS_SET2	0..65535	The maximum number of interval data blocks that can be contained in **LP_DATA_SET2_TBL** (Table 65).
NBR_BLK_INTS_SET2	0..65535	The maximum number of intervals per data block that can be contained in **LP_DATA_SET2_TBL** (Table 65).
NBR_CHNS_SET2	0..255	The maximum number of channels of load profile data that can be contained in **LP_DATA_SET2_TBL** (Table 65).
MAX_INT_TIME_SET2	0..255	The maximum time in minutes for load profile interval duration that can be contained in **LP_DATA_SET2_TBL** (Table 65).
NBR_BLKS_SET3	0..65535	The maximum number of interval data blocks that can be contained in **LP_DATA_SET3_TBL** (Table 66).
NBR_BLK_INTS_SET3	0..65535	The maximum number of intervals per data block that can be contained in **LP_DATA_SET3_TBL** (Table 66).
NBR_CHNS_SET3	0..255	The maximum number of channels of load profile data that can be contained in **LP_DATA_SET3_TBL** (Table 66).
MAX_INT_TIME_SET3	0.255	The maximum time in minutes for load profile interval duration that can be contained in **LP_DATA_SET3_TBL** (Table 66).
NBR_BLKS_SET4	0..65535	The maximum number of interval data blocks that can be contained in **LP_DATA_SET4_TBL** (Table 67).
NBR_BLK_INTS_SET4	0..65535	The maximum number of intervals per data block that can be contained in **LP_DATA_SET4_TBL** (Table 67).

NBR_CHNS_SET4	0..255	The maximum number of channels of load profile data that can be contained in **LP_DATA_SET4_TBL** (Table 67).
MAX_INT_TIME_SET4	0..255	The maximum time in minutes for load profile interval duration that can be contained in **LP_DATA_SET4_TBL** (Table 67).

9.7.2 TABLE 61 Actual load profile table

Table 61 Data description

ACT_LP_TBL (Table 61) contains actual values for the load profile application limiting parameters. These parameters provide for up to four independent sets of load profile data.

TABLE ACT_LP_TBL = DIM_LP_TBL.LP_SET_RCD;

Identifier	*Value*	*Definition*
LP_FLAGS_BFLD		This bit field specifies load profile list management capabilities.
LP_SET1_INHIBIT_OVF_FLAG		
	FALSE	End device is not inhibiting load profile set 1 once an overflow occurs.
	TRUE	End device is inhibiting load profile set 1 once an overflow occurs.
LP_SET2_INHIBIT_OVF_FLAG		
	FALSE	End device is not inhibiting load profile set 2 once an overflow occurs.
	TRUE	End device is inhibiting load profile set 2 once an overflow occurs.
LP_SET3_INHIBIT_OVF_FLAG		
	FALSE	End device is not inhibiting load profile set 3 once an overflow occurs.
	TRUE	End device is inhibiting load profile set 3 once an overflow occurs.
LP_SET4_INHIBIT_OVF_FLAG		
	FALSE	End device is not inhibiting load profile set 4 once an overflow occurs.
	TRUE	End device is inhibiting load profile set 4 once an overflow occurs.
BLK_END_READ_FLAG		
	FALSE	End device is not providing block register reading information.
	TRUE	End device is providing block register reading information.

BLK_END_PULSE_FLAG

FALSE	End device does not have a running pulse accumulator.
TRUE	End device does have a running pulse accumulator.

SCALAR_DIVISOR_FLAG_SET1

FALSE	End device does not have scalars and divisors associated with set 1 load profile interval data.
TRUE	End device does have scalars and divisors associated with set 1 load profile interval data.

SCALAR_DIVISOR_FLAG_SET2

FALSE	End device does not have scalars and divisors associated with set 2 load profile interval data.
TRUE	End device has scalars and divisors associated with set 2 load profile interval data.

SCALAR_DIVISOR_FLAG_SET3

FALSE	End device does not have scalars and divisors associated with set 3 load profile interval data.
TRUE	End device does have scalars and divisors associated with set 3 load profile interval data.

SCALAR_DIVISOR_FLAG_SET4

FALSE	End device does not have scalars and divisors associated with set 4 load profile interval data.
TRUE	End device does have scalars and divisors associated with set 4 load profile interval data.

EXTENDED_INT_STATUS_FLAG

FALSE	End device does not return extended interval status with load profile interval data.
TRUE	End device returns extended interval status with load profile interval data.

SIMPLE_INT_STATUS_FLAG

FALSE	End device does not return simple interval status with load profile interval data.
TRUE	End device returns simple interval status with load profile interval data.

LP_FMATS_BFLD — This set of booleans specifies the format(s) in use in **LP_CTRL_TBL** (Table 62).

INV_UINT8_FLAG	FALSE	**UINT8** format can not be in use.
	TRUE	An interval format of **UINT8** is being used.
INV_UINT16_FLAG	FALSE	**UINT16** format can not be in use.
	TRUE	An interval format of **UINT16** is being used.
INV_UINT32_FLAG	FALSE	**UINT32** format can not be in use.
	TRUE	An interval format of **UINT32** can be used.

INV_INT8_FLAG	FALSE TRUE	**INT8** format can not be in use. An interval format of **INT8** is being used.
INV_INT16_FLAG	FALSE TRUE	**INT16** format can not be in use. An interval format of **INT16** is being used.
INV_INT32_FLAG	FALSE TRUE	**INT32** format can not be in use. An interval format of **INT32** is being used.
INV_NI_FMAT1_FLAG	FALSE TRUE	**NI_FMAT1** format can not be in use. An interval format of **NI_FMAT1** is being used.
INV_NI_FMAT2_FLAG	FALSE TRUE	**NI_FMAT2** format can not be in use. An interval format of **NI_FMAT2** is being used.
LP_SET_RCD		The information defining how each set of load profile information works.
LP_MEMORY_LEN	0..4 294 967 295	The current number of octets of storage being used for load profile data. This reflects the combined sizes of tables **LP_DATA_SET1_TBL** (Table 64), **LP_DATA_SET2_TBL** (Table 65), **LP_DATA_SET3_TBL** (Table 66), and **LP_DATA_SET4_TBL** (Table 67).
LP_FLAGS		See **LP_FLAGS_BFLD** above.
LP_FMATS		See **LP_FMATS_BFLD** above.
NBR_BLKS_SET1	0..65535	The actual number of interval data blocks in use in **LP_DATA_SET1_TBL** (Table 64).
NBR_BLK_INTS_SET1	0..65535	The actual number of intervals per data block contained in **LP_DATA_SET1_TBL** (Table 64).
NBR_CHNS_SET1	0..255	The actual number of channels of load profile data contained in **LP_DATA_SET1_TBL** (Table 64).
MAX_INT_TIME_SET1	0..255	The actual interval duration in minutes for load profile contained in **LP_DATA_SET1_TBL** (Table 64).
NBR_BLKS_SET2	0..65535	The actual number of interval data blocks in use in **LP_DATA_SET2_TBL** (Table 65).
NBR_BLK_INTS_SET2	0..65535	The actual number of intervals per data block contained in **LP_DATA_SET2_TBL** (Table 65).
NBR_CHNS_SET2	0..255	The actual number of channels of load profile data contained in **LP_DATA_SET2_TBL** (Table 65).

MAX_INT_TIME_SET2	0..255	The actual interval duration in minutes for load profile contained in **LP_DATA_SET2_TBL** (Table 65).
NBR_BLKS_SET3	0..65535	The actual number of interval data blocks in use in **LP_DATA_SET3_TBL** (Table 66).
NBR_BLK_INTS_SET3	0..65535	The actual number of intervals per data block contained in **LP_DATA_SET3_TBL** (Table 66).
NBR_CHNS_SET3	0..255	The actual number of channels of load profile data contained in **LP_DATA_SET3_TBL** (Table 66).
MAX_INT_TIME_SET3	0..255	The actual interval duration in minutes for load profile contained in **LP_DATA_SET3_TBL** (Table 66).
NBR_BLKS_SET4	0..65535	The actual number of interval data blocks in use in **LP_DATA_SET4_TBL** (Table 67).
NBR_BLK_INTS_SET4	0..65535	The actual number of intervals per data block contained in **LP_DATA_SET4_TBL** (Table 67).
NBR_CHNS_SET4	0..255	The actual number of channels of load profile data contained in **LP_DATA_SET4_TBL** (Table 67).
MAX_INT_TIME_SET4	0..255	The actual interval duration in minutes for load profile contained in **LP_DATA_SET4_TBL** (Table 67).

9.7.3 TABLE 62 Load profile control table

Table 62 Data description

LP_CTRL_TBL (Table 62) defines the data sources and formats used in the collection of load profile data.

```
TYPE LP_CTRL_FLAGS_BFLD = BIT FIELD OF UINT8
      END_RDG_FLAG                  : BOOL(0);
      FILLER                        : FILL(1..7);
END;

TYPE LP_SOURCE_SEL_RCD = PACKED RECORD
      CHNL_FLAG                     : LP_CTRL_FLAGS_BFLD;
      LP_SOURCE_SELECT              : UINT8;
      END_BLK_RDG_SOURCE_SELECT     : UINT8;
END;
```

```
TYPE DATA_SELECTION_RCD = PACKED RECORD
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET1_TBL THEN
            LP_SEL_SET1             : ARRAY[ACT_LP_TBL.NBR_CHNS_SET1]
                                          OF LP_SOURCE_SEL_RCD;
            INT_FMT_CDE1            : UINT8;
            IF ACT_LP_TBL.SCALAR_DIVISOR_FLAG_SET1LP_SET1_UNITS = 1 THEN
                  SCALARS_SET1      : ARRAY[ACT_LP_TBL.NBR_CHNS_SET1] OF
                                          UINT16;
                  DIVISOR_SET1      : ARRAY[ACT_LP_TBL.NBR_CHNS_SET1] OF
                                          UINT16;
            END;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET2_TBL THEN
            LP_SEL_SET2             : ARRAY[ACT_LP_TBL.NBR_CHNS_SET2] OF
                                           LP_SOURCE_SEL_RCD;
            INT_FMT_CDE2            : UINT8;
            IF ACT_LP_TBL.SCALAR_DIVISOR_FLAG_SET2LP_SET2_UNITS = 1 THEN
                  SCALARS_SET2      : ARRAY[ACT_LP_TBL.NBR_CHNS_SET2] OF
                                           UINT16;
                  DIVISOR_SET2      : ARRAY[ACT_LP_TBL.NBR_CHNS_SET2] OF
                                          UINT16;
            END;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET3_TBL THEN
            LP_SEL_SET3             : ARRAY[ACT_LP_TBL.NBR_CHNS_SET3]
                                          OF LP_SOURCE_SEL_RCD;
            INT_FMT_CDE3            : UINT8;
            IF ACT_LP_TBL.SCALAR_DIVISOR_FLAG_SET3LP_SET3_UNITS = 1 THEN
                  SCALARS_SET3      : ARRAY[ACT_LP_TBL.NBR_CHNS_SET3] OF
                                          UINT16;
                  DIVISOR_SET3      : ARRAY[ACT_LP_TBL.NBR_CHNS_SET3] OF
                                          UINT16;
            END;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET4_TBL THEN
            LP_SEL_SET4             : ARRAY[ACT_LP_TBL.NBR_CHNS_SET4]
                                           OF LP_SOURCE_SEL_RCD;
            INT_FMT_CDE4            : UINT8;
            IF ACT_LP_TBL.SCALAR_DIVISOR_FLAG_SET4LP_SET4_UNITS = 1 THEN
                  SCALARS_SET4      : ARRAY[ACT_LP_TBL.NBR_CHNS_SET4] OF
                                          UINT16;
                  DIVISOR_SET4      : ARRAY[ACT_LP_TBL.NBR_CHNS_SET4] OF
                                          UINT16;
            END;
      END;
END;

TABLE LP_CTRL_TBL = DATA_SELECTION_RCD;
```

Identifier	*Value*	*Definition*
LP_CTRL_FLAGS_BFLD		
END_RDG_FLAG	FALSE	The channel does not have an associated end reading.
	TRUE	The channel does have an associated end reading.
LP_SOURCE_SEL_RCD		
CHNL_FLAG		Flags associated with a particular channel.
INT_SOURCE_SELECT	0..255	This is an index into **SOURCES_TBL** (Table 16) that identifies source of the interval data for a specific channel.
END_BLK_RDG_SOURCE_SELECT	0..255	This is an index into **SOURCES_TBL** (Table 16) that identifies source of the block end time reading data for a specific channel.
DATA_SELECTION_RCD		Record that stores all information concerning each of the selected channels.
LP_SEL_SET1		This is an array of records containing indices into the **SOURCES_TBL** (Table 16) that identifies sources of data for each channel of interval data in **LP_DATA_SET1_TBL** (Table 64).
INT_FMT_CDE1		This is a single code selecting the format for all interval data in **LP_DATA_SET1_TBL** (Table 64).
		These codes are defined as follows:
	1	**UINT8**
	2	**UINT16**
	4	**UINT32**
	8	**INT8**
	16	**INT16**
	32	**INT32**
	64	**NI_FMAT1**
	128	**NI_FMAT2**
		All other code values are currently undefined.
SCALARS_SET1		An array of scalars applied to interval data before recording pulse data in load profile set 1.
DIVISORS_SET1		An array of divisors applied to interval data before recording pulse data in load profile set 1.
LP_SEL_SET2		This is an array of records containing indices into the **SOURCES_TBL** (Table 16) that identifies sources of data for each channel of interval data in **LP_DATA_SET2_TBL** (Table 65).

INT_FMT_CDE2

This is a single code selecting the format for all interval data in **LP_DATA_SET2_TBL** (Table 65).

These codes are defined as follows:

1	**UINT8**
2	**UINT16**
4	**UINT32**
8	**INT8**
16	**INT16**
32	**INT32**
64	**NI_FMAT1**
128	**NI_FMAT2**

All other code values are currently undefined.

SCALARS_SET2

An array of scalars applied to interval data before recording pulse data in load profile set 2.

DIVISORS_SET2

An array of divisors applied to interval data before recording pulse data in load profile set 2.

LP_SEL_SET3

This is an array of records containing indices into the **SOURCES_TBL** (Table 16) that identifies sources of data for each channel of interval data in **LP_DATA_SET3_TBL** (Table 66).

INT_FMT_CDE3

This is a single code selecting the format for all interval data in **LP_DATA_SET3_TBL** (Table 66).

These codes are defined as follows:

1	**UINT8**
2	**UINT16**
4	**UINT32**
8	**INT8**
16	**INT16**
32	**INT32**
64	**NI_FMAT1**
128	**NI_FMAT2**

All other code values are currently undefined.

SCALARS_SET3

An array of scalars applied to interval data before recording pulse data in load profile set 3.

DIVISORS_SET3

An array of divisors applied to interval data before recording pulse data in load profile set 3.

LP_SEL_SET4

This is an array of records containing indices into the **SOURCES_TBL** (Table 16) that identifies sources of data for each channel of interval data in **LP_DATA_SET4_TBL** (Table 67).

INT_FMT_CDE4 — This is a single code selecting the format for all interval data in **LP_DATA_SET4_TBL** (Table 67).

These codes are defined as follows:

1	**UINT8**
2	**UINT16**
4	**UINT32**
8	**INT8**
16	**INT16**
32	**INT32**
64	**NI_FMAT1**
128	**NI_FMAT2**

All other code values are currently undefined.

SCALARS_SET4 — An array of scalars applied to interval data before recording pulse data in load profile set 4.

DIVISORS_SET4 — An array of divisors applied to interval data before recording pulse data in load profile set 4.

9.7.4 TABLE 63 Load profile status table

Table 63 Data description

LP_STATUS_TBL (Table 63) contains the status of each load profile data set.

```
TYPE LP_SET_STATUS_BFLD = BIT FIELD OF UINT8
      BLOCK_ORDER                   : UINT(0..0);
      OVERFLOW_FLAG                 : BOOL(1);
      LIST_TYPE                     : UINT(2..2);
      BLOCK_INHIBIT_OVERFLOW_FLAG   : BOOL(3);
      INTERVAL_ORDER                : UINT(4..4);
      ACTIVE_MODE_FLAG              : BOOL(5);
      TEST_MODE                     : UINT(6..6);
      FILLER                        : FILL(7..7);
END;

TYPE LP_SET_STATUS_RCD = PACKED RECORD
      LP_SET_STATUS_FLAGS           : LP_SET_STATUS_BFLD;
      NBR_VALID_BLOCKS              : UINT16;
      LAST_BLOCK_ELEMENT            : UINT16;
      LAST_BLOCK_SEQ_NBR            : UINT32;
      NBR_UNREAD_BLOCKS             : UINT16;
      NBR_VALID_INT                 : UINT16;
END;

TYPE LP_STATUS_RCD = PACKED RECORD
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET1_TBL_CNST THEN
            LP_STATUS_SET1          : LP_SET_STATUS_RCD;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET2_TBL_CNST THEN
            LP_STATUS_SET2          : LP_SET_STATUS_RCD;
```

```
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET3_TBL_CNST THEN
            LP_STATUS_SET3          : LP_SET_STATUS_RCD;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.LP_DATA_SET4_TBL_CNST THEN
            LP_STATUS_SET4          : LP_SET_STATUS_RCD;
      END;
END;

TABLE LP_STATUS_TBL = LP_STATUS_RCD;
```

Identifier	*Value*	*Definition*
LP_SET_STATUS_BFLD		
BLOCK_ORDER	0	Blocks of load profile data are transported in ascending order (N is older than N+1).
	1	Blocks of load profile data are transported in descending order (N is newer than N+1).
OVERFLOW_FLAG	FALSE	Overflow has not occurred.
	TRUE	An attempt was made to enter an interval in a new data block such that the number of unread blocks exceeded the actual number of possible blocks in load profile storage.
LIST_TYPE	0	FIFO (First In-First Out) as placed in load profile storage.
	1	Circular list as placed in load profile storage.
BLOCK_INHIBIT_OVERFLOW_FLAG		The same value as **ACT_LP_TBL.LP_SET**n**_INHIBIT_OVF_FLAG**.
INTERVAL_ORDER	0	Intervals in each block of load profile are transported in ascending order (N is older than N+1).
	1	Intervals in each block of load profile are transported in descending order (N is newer than N+1).
ACTIVE_MODE_FLAG	FALSE	This data set is not collecting data.
	TRUE	This data set is collecting data.
TEST_MODE	0	This data set is in the normal mode.
	1	This data set is in the test mode.
LP_SET_STATUS_RCD		
LP_SET_STATUS_FLAGS		See **LP_SET_STATUS_BFLD**.
NBR_VALID_BLOCKS	0..65535	Number of valid load profile data blocks in load profile data tables **LP_DATA_SET1_TBL** (Table 64), **LP_DATA_SET2_TBL** (Table 65), **LP_DATA_SET3_TBL** (Table 66), and **LP_DATA_SET4_TBL** (Table 67), load profile block arrays. The range is zero (meaning no data blocks in load profile data table) to the actual dimension of the number of load profile data blocks. The block is considered valid when at least one interval is written.

LAST_BLOCK_ELEMENT	0..65535	The array element of the newest valid data block in the load profile data array. This field is valid only if **NBR_VALID_BLOCKS** is greater than zero.
LAST_BLOCK_SEQ_NBR	0..4 294 967 295	The sequence number (**LP_SEQ_NBR**) of the last element (**LAST_BLOCK_ELEMENT**) in the load profile data array.
NBR_UNREAD_BLOCKS	0..65535	The number of load profile blocks that have not been read. This number is only updated through a procedure.
NBR_VALID_INT	0..65535	Number of valid intervals stored in the last load profile block array. The range is zero (meaning no interval in the array) to the actual dimension of the number of intervals per block.

LP_STATUS_RCD

LP_STATUS_SET1	Status information for profile data set 1.
LP_STATUS_SET2	Status information for profile data set 2.
LP_STATUS_SET3	Status information for profile data set 3.
LP_STATUS_SET4	Status information for profile data set 4.

9.7.5 TABLE 64 Load profile data set 1 table

Table 64 Data description

LP_DATA_SET1_TBL (Table 64) contains information on load profile data set one. The parameters are as follows:

```
TYPE INT_FMT1_RCD = PACKED RECORD
      CASE LP_CTRL_TBL.INT_FMT_CDE1 OF
            1     :     ITEM  : UINT8;
            2     :     ITEM  : UINT16;
            4     :     ITEM  : UINT32;
            8     :     ITEM  : INT8;
            16    :     ITEM  : INT16;
            32    :     ITEM  : UINT32;
            64    :     ITEM  : NI_FMAT1;
            128   :     ITEM  : NI_FMAT2;
      END;
END;

TYPE INT_SET1_RCD = PACKED RECORD
      IF ACT_LP_TBL.EXTENDED_INT_STATUS_FLAG THEN
            EXTENDED_INT_STATUS INT_STATUS      :
                  ARRAY[(ACT_LP_TBLLP_ACT_LIM_TBL.NBR_CHNS_SET1/2)+1]
                        OF UINT8;
      END;
```

```
      INT_DATA :  ARRAY[ACT_LP_TBL_TBL.NBR_CHNS_SET1] OF INT_FMT1_RCD;
END;

TYPE READINGS_RCD = PACKED RECORD
      IF ACT_LP_TBL.BLK_END_READ_FLAG THEN
            BLOCK_END_READ    : NI_FMAT1;
      END;
      IF ACT_LP_TBL.BLK_END_PULSE_FLAG THEN
            BLOCK_END_PULSE   : UINT32;
      END;
END;

TYPE LP_BLK1_DAT_RCD = PACKED RECORD
      BLK_END_TIME      : STIME_DATE;
      END_READINGS      : ARRAY[ACT_LP_TBL_TBL.NBR_CHNS_SET1]
                              OF READINGS_RCD;
      IF ACT_LP_TBL.SIMPLE_INT_STATUS_FLAG THEN
            SIMPLE_INT_STATUS : SET((ACT_LP_TBL.NBR_BLK_INTS_SET1+7)/8);
      END;

      LP_INT            : ARRAY[ACT_LP_TBL_TBL.NBR_BLK_INTS_SET1]
                                    OF INT_SET1_RCD;
END;

TYPE LP_DATA_SET1_RCD = PACKED RECORD
      LP_DATA_SETS1     : ARRAY[ACT_LP_TBL_TBL.NBR_BLKS_SET1] OF
                          LP_BLK1_DAT_RCD;
END;

TABLE LP_DATA_SET1_TBL = LP_DATA_SET1_RCD;
```

INT_FMT1_RCD — Selector for the format of the channel data.

Identifier	*Value*	*Definition*
INT_SET1_RCD		
EXTENDED_INT_STATUS		This is one or more single octets of interval status conditions provided for all channels. One nibble (4 b) of status information are provided for each channel in the set, plus one nibble for status conditions common to all channels. This common status nibble is always the first nibble of the set. (That is, the high order 4 b of the first octet of **EXTENDED_INT_STATUS**). Since **EXTENDED_INT_STATUS** is an integer number of octets in length, this field contains one nibble of **FILL** if the number of channels in the set is even. This fill field, when present always occupies the last nibble of **EXTENDED_INT_STATUS.** (That is, the low order 4 b of the last octet of **EXTENDED_INT_STATUS**). The contents of the channel status nibble is a binary value defined as follows:

	0	No status flag
	1	Overflow
	2	Partial interval due to common state
	3	Long interval due to common state
	4	Skipped interval due to common state
	5	Interval contains test mode data
	6..15	Undefined
		The contents of the common status nibble are unitary flags defined as follows:
	0	DST is in effect during or at start of interval
	1	Power fail within interval
	2	Clock reset forward during interval
	3	Clock reset backwards during interval
INT_DATA		This is an array of interval values for each channel in the block, recorded at a single time. Channels are presented in the same order. The order of the channels matches their definition in **LP_CTRL_TBL** (Table 62).
READINGS_RCD		
BLOCK_END_READ		The value of the block end reading both data and readings.
BLOCK_END_PULSE	0..65355	The values of the accumulator at the end of a user defined interval.
LP_BLK1_DAT_RCD		
BLK_END_TIME		This parameter contains the ending date and time of the last interval of data entered in this data block.
END_READINGS		This is an array of readings of the end reading sources, taken at the interval end time of the last interval of data entered in the block. One end reading is provided for each channel in the block. The order of the readings matches their definition in **LP_CTRL_TBL** (Table 62). The format is controlled by **READING_RCD**.
SIMPLE_INT_STATUS		This is a set of status bits, one per interval, that specifies whether the corresponding interval is valid.
LP_INT		This is an array of interval records in the block. Intervals are stored in an chronological order, specified by **INTERVAL_ORDER**, with the oldest reading set appearing in the first record.
LP_DATA_SETS_RCD		This is an array of load profile data blocks.
LP_DATA_SETS1		This array contains the function limiting information about this load profile data set.

9.7.6 TABLE 65 Load profile data set 2 table

Table 65 Data description

LP_DATA_SET2_TBL (Table 65) contains information on load profile data set two. The parameters are as follows:

```
TYPE INT_FMT2_RCD = PACKED RECORD
      CASE LP_CTRL_TBL.INT_FMT_CDE2 OF
            1     :     ITEM  : UINT8;
            2     :     ITEM  : UINT16;
            4     :     ITEM  : UINT32;
            8     :     ITEM  : INT8;
            16    :     ITEM  : INT16;
            32    :     ITEM  : UINT32;
            64    :     ITEM  : NI_FMAT1;
            128   :     ITEM  : NI_FMAT2;
      END;
END;

TYPE INT_SET2_RCD = PACKED RECORD
      IF ACT_LP_TBL.EXTENDED_INT_STATUS_FLAG THEN
            EXTENDED_INT_STATUS :
                  ARRAY[(ACT_LP_TBLTBL_TBL.NBR_CHNS_SET2 / 2)+1] OF
                  UINT8;
      END;

      INT_DATA : ARRAY[ACT_LP_TBLTBL_TBL.NBR_CHNS_SET2] OF INT_FMT2_RCD;
END;

TYPE READINGS_RCD = PACKED RECORD
      IF ACT_LP_TBL.BLK_END_READ_FLAG THEN
            BLOCK_END_READ    : NI_FMAT1;
      END;
      IF ACT_LP_TBL.BLK_END_PULSE_FLAG THEN
            BLOCK_END_PULSE   : UINT32;
      END;
END;

TYPE LP_BLK2_DAT_RCD = PACKED RECORD
      BLK_END_TIME      : STIME_DATE;
      END_READINGS      : ARRAY[ACT_LP_TBLTBL_TBL.NBR_CHNS_SET2]
                              OF READINGS_RCD;

      IF ACT_LP_TBL.SIMPLE_INT_STATUS_FLAG THEN
            SIMPLE_INT_STATUS : SET((ACT_LP_TBL.NBR_BLK_INTS_SET2+7)/8);
      END;

      LP_INT      : ARRAY[ACT_LP_TBLTBL_TBL.NBR_BLK_INTS_SET2] OF
                        INT_SET2_RCD;
END;
```

```
TYPE LP_DATA_SET2_RCD = PACKED RECORD
      LP_DATA_SETS2     : ARRAY[ACT_LP_TBLTBL_TBL.NBR_BLKS_SET2] OF
                              LP_BLK2_DAT_RCD;
END;

TABLE LP_DATA_SET2_TBL = LP_DATA_SET2_RCD;
```

Identifier	*Value*	*Definition*
INT_FMT2_RCD		Selector for the format of the channel data.
INT_SET2_RCD		
EXTENDED_INT_STATUS		This is one or more single octets of interval status conditions provided for all channels. One nibble (4 b) of status information are provided for each channel in the set, plus one nibble for status conditions common to all channels. This common status nibble is always the first nibble of the set. (That is, the high order 4 b of the first octet of **EXTENDED_INT_STATUS**). Since **EXTENDED_INT_STATUS** is an integer number of octets in length, this field contains one nibble of **FILL** if the number of channels in the set is even. This fill field, when present always occupies the last nibble of **EXTENDED_INT_STATUS.** (That is, the low order 4 b of the last octet of **EXTENDED_INT_STATUS**). The contents of the channel status nibble is a binary value defined as follows:
	0	No status flag
	1	Overflow
	2	Partial interval due to common state
	3	Long interval due to common state
	4	Skipped interval due to common state
	5	Interval contains test mode data
	6..15	Undefined
		The contents of the common status nibble are unitary flags defined as follows:
	0	DST is in effect during or at start of interval
	1	Power fail within interval
	2	Clock reset forward during interval
	3	Clock reset backwards during interval
INT_DATA		This is an array of interval values for each channel in the block, recorded at a single time. Channels are presented in the same order. The order of the channels matches their definition in **LP_CTRL_TBL** (Table 62).

READINGS_RCD		
BLOCK_END_READ		The value of the block end reading both data and readings.
BLOCK_END_PULSE	0..65355	The values of the accumulator at the end of a user defined interval.
LP_BLK2_DAT_RCD		
BLK_END_TIME		This parameter contains the ending date and time of the last interval of data entered in this data block.
END_READINGS		This is an array of readings of the end reading sources, taken at the interval end time of the last interval of data entered in the block. One end reading is provided for each channel in the block. The order of the readings matches their definition in **LP_CTRL_TBL** (Table 62). The format is controlled by **READING_RCD**.
SIMPLE_INT_STATUS		This is a set of status bits, one per interval, that specifies whether the corresponding interval is valid.
LP_INT		This is an array of interval records in the block. Intervals are stored in an chronological order, specified by **INTERVAL_ORDER**, with the oldest reading set appearing in the first record.
LP_DATA_SETS_RCD		This is an array of load profile data blocks.
LP_DATA_SETS2		This array contains the function limiting information about this load profile data set.

9.7.7 TABLE 66 Load profile data set three table

Table 66 Data description

LP_DATA_SET3_TBL (Table 66) contains information on load profile data set three.

Parameters are as follows:

```
TYPE INT_FMT3_RCD = PACKED RECORD
      CASE LP_CTRL_TBL.INT_FMT_CDE3 OF
            1     :     ITEM  : UINT8;
            2     :     ITEM  : UINT16;
            4     :     ITEM  : UINT32;
            8     :     ITEM  : INT8;
            16    :     ITEM  : INT16;
            32    :     ITEM  : UINT32;
            64    :     ITEM  : NI_FMAT1;
            128   :     ITEM  : NI_FMAT2;
      END;
END;
```

```
TYPE INT_SET3_RCD = PACKED RECORD
      IF ACT_LP_TBL.EXTENDED_INT_STATUS_FLAG THEN
            EXTENDED_INT_STATUS :
                  ARRAY[(ACT_LP_TBLTBL_TBL.NBR_CHNS_SET3 / 2)+1]
                        OF UINT8;
      END;

      INT_DATA    : ARRAY[ACT_LP_TBLTBL_TBL.NBR_CHNS_SET3]
                        OF INT_FMT3_RCD;
END;

TYPE READINGS_RCD = PACKED RECORD
      IF ACT_LP_TBL.BLK_END_READ_FLAG THEN
            BLOCK_END_READ    : NI_FMAT1;
      END;
      IF ACT_LP_TBL.BLK_END_PULSE_FLAG THEN
            BLOCK_END_PULSE   : UINT32;
      END;
END;

TYPE LP_BLK3_DAT_RCD = PACKED RECORD
      BLK_END_TIME      : STIME_DATE;
      END_READINGS      : ARRAY[ACT_LP_TBLTBL_TBL.NBR_CHNS_SET3]
                          OF READINGS_RCD;

      IF ACT_LP_TBL.SIMPLE_INT_STATUS_FLAG THEN
            SIMPLE_INT_STATUS : SET((ACT_LP_TBL.NBR_BLK_INTS_SET3+7)/8);
      END;

      LP_INT      : ARRAY[ACT_LP_TBLTBL_TBL.NBR_BLK_INTS_SET3] OF
                        INT_SET3_RCD;
END;

TYPE LP_DATA_SET3_RCD = PACKED RECORD
      LP_DATA_SETS3     : ARRAY[ACT_LP_TBLTBL_TBL.NBR_BLKS_SET3] OF
                              LP_BLK3_DAT_RCD;
END;

TABLE LP_DATA_SET3_TBL = LP_DATA_SETS_RCD;
```

Identifier	*Value*	*Definition*
INT_FMT3_RCD		Selector for the format of the channel data.
INT_SET3_RCD		
EXTENDED_INT_STATUS		This is one or more single octets of interval status conditions provided for all channels. One nibble (4 b) of status information are provided for each channel in the set, plus one nibble for status conditions common to all channels. This common status nibble is always the first nibble of the set. (That is, the high order 4 b of the first octet of **EXTENDED_INT_STATUS**). Since **EXTENDED_INT_STATUS** is an integer number of octets in length, this field contains one nibble of **FILL** if the number of channels in the set is even.

This fill field, when present always occupies the last nibble of **EXTENDED_INT_STATUS**. (That is, the low order 4 b of the last octet of **EXTENDED_INT_STATUS**).

The contents of the channel status nibble is a binary value defined as follows:

0	No status flag
1	Overflow
2	Partial interval due to common state
3	Long interval due to common state
4	Skipped interval due to common state
5	Interval contains test mode data
6..15	Undefined

The contents of the common status nibble are unitary flags defined as follows:

0	DST is in effect during or at start of interval
1	Power fail within interval
2	Clock reset forward during interval
3	Clock reset backwards during interval

INT_DATA — This is an array of interval values for each channel in the block, recorded at a single time. Channels are presented in the same order.same order The order of the channels matches their definition in **LP_CTRL_TBL** (Table 62).

READINGS_RCD

BLOCK_END_READ — The value of the block end reading both data and readings.

BLOCK_END_PULSE 0..65355 — The values of the accumulator at the end of a user-defined interval.

LP_BLK3_DAT_RCD

BLK_END_TIME — This parameter contains the ending date and time of the last interval of data entered in this data block.

END_READINGS — This is an array of readings of the end reading sources, taken at the interval end time of the last interval of data entered in the block. One end reading is provided for each channel in the block. The order of the readings matches their definition in **LP_CTRL_TBL** (Table 62). The format is controlled by **READING_RCD**.

SIMPLE_INT_STATUS — This is a set of status bits, one per interval, that specifies whether the corresponding interval is valid.

LP_INT	This is an array of interval records in the block. Intervals are stored in an chronological order, specified by **INTERVAL_ORDER**, with the oldest reading set appearing in the first record.
LP_DATA_SETS_RCD	This is an array of load profile data blocks.
LP_DATA_SETS3	This array contains the function limiting information about this load profile data set.

9.7.8 TABLE 67 Load profile data set 4 table

Table 67 Data description

LP_DATA_SET4_TBL (Table 67) contains information on load profile data set four.

Parameters are as follows:

```
TYPE INT_FMT4_RCD = PACKED RECORD
      CASE LP_CTRL_TBL.INT_FMT_CDE4 OF
            1     :     ITEM  : UINT8;
            2     :     ITEM  : UINT16;
            4     :     ITEM  : UINT32;
            8     :     ITEM  : INT8;
            16    :     ITEM  : INT16;
            32    :     ITEM  : INT32;
            64    :     ITEM  : NI_FMAT1;
            128   :     ITEM  : NI_FMAT2;
      END;
END;

TYPE INT_SET4_RCD = PACKED RECORD
      IF ACT_LP_TBL.EXTENDED_INT_STATUS_FLAG THEN
            EXTENDED_INT_STATUS :
                  ARRAY[(ACT_LP_TBLTBL_TBL.NBR_CHNS_SET4 / 2)+1]
                        OF UINT8;
      END;

      INT_DATA    : ARRAY[ACT_LP_TBLTBL_TBL.NBR_CHNS_SET4]
                        OF INT_FMT4_RCD;
END;

TYPE READINGS_RCD = PACKED RECORD
      IF ACT_LP_TBL.BLK_END_READ_FLAG THEN
            BLOCK_END_READ    : NI_FMAT1;
      END:
      IF ACT_LP_TBL.BLK_END_PULSE_FLAG THEN
            BLOCK_END_PULSE   : UINT32;
      END:
END;

TYPE LP_BLK4_DAT_RCD = PACKED RECORD
      BLK_END_TIME      : STIME_DATE;
      END_READINGS      : ARRAY[ACT_LP_TBLTBL_TBL.NBR_CHNS_SET4]
                              OF READINGS_RCD;
```

```
      IF ACT_LP_TBL.SIMPLE_INT_STATUS_FLAG THEN
            SIMPLE_INT_STATUS : SET((ACT_LP_TBL.NBR_BLK_INTS_SET4+7)/8);
      END;

      LP_INT      : ARRAY[ACT_LP_TBLTBL_TBL.NBR_BLK_INTS_SET4] OF
                        INT_SET4_RCD;
END;

TYPE LP_DATA_SET4_RCD = PACKED RECORD
      LP_DATA_SETS4     : ARRAY[ACT_LP_TBLTBL_TBL.NBR_BLKS_SET4] OF
                              LP_BLK4_DAT_RCD;
END;

TABLE LP_DATA_SET4_TBL = LP_DATA_SET4_RCD;
```

Identifier	*Value*	*Definition*
INT_FMT4_RCD		Selector for the format of the channel data.
INT_SET4_RCD		
EXTENDED_INT_STATUS		This is one or more single octets of interval status conditions provided for all channels. One nibble (4 b) of status information are provided for each channel in the set, plus one nibble for status conditions common to all channels. This common status nibble is always the first nibble of the set. (That is, the high order 4 b of the first octet of **EXTENDED_INT_STATUS**). Since **EXTENDED_INT_STATUS** is an integer number of octets in length, this field contains one nibble of **FILL** if the number of channels in the set is even. This fill field, when present always occupies the last nibble of **EXTENDED_INT_STATUS**. (That is, the low order 4 b of the last octet of **EXTENDED_INT_STATUS**).
		The contents of the channel status nibble is a binary value defined as follows:
	0	No status flag
	1	Overflow
	2	Partial interval due to common state
	3	Long interval due to common state
	4	Skipped interval due to common state
	5	Interval contains test mode data
	6..15	Undefined
		The contents of the common status nibble are unitary flags defined as follows:
	0	DST is in effect during or at start of interval
	1	Power fail within interval
	2	Clock reset forward during interval
	3	Clock reset backwards during interval

INT_DATA		This is an array of interval values for each channel in the block, recorded at a single time. Channels are presented in the same order. The order of the channels matches their definition in **LP_CTRL_TBL** (Table 62).
READINGS_RCD		
BLOCK_END_READ		The value of the block end reading both data and readings.
BLOCK_END_PULSE	0..65355	The values of the accumulator at the end of a user-defined interval.
LP_BLK4_DAT_RCD		
BLK_END_TIME		This parameter contains the ending date and time of the last interval of data entered in this data block.
END_READINGS		This is an array of readings of the end reading sources, taken at the interval end time of the last interval of data entered in the block. One end reading is provided for each channel in the block. The order of the readings matches their definition in **LP_CTRL_TBL** (Table 62). The format is controlled by **READING_RCD**.
SIMPLE_INT_STATUS		This is a set of status bits, one per interval, that specifies whether the corresponding interval is valid.
LP_INT		This is an array of interval records in the block. Intervals are stored in an chronological order, specified by **INTERVAL_ORDER**, with the oldest reading set appearing in the first record.
LP_DATA_SETS_RCD		This is an array of load profile data blocks.
LP_DATA_SETS4		This array contains the function limiting information about this load profile data set.

9.8 DECADE 70: History and event logs

This decade contains the tables associated with the maintenance of the history and event logs.

9.8.1 TABLE 70 Dimension log table

Table 70 Data description

DIM_LOG_TBL (Table 70) defines the maximum size and capabilities of the history and event log decade.

```
TYPE LOG_FLAGS_BFLD = BIT FIELD OF UINT8
      EVENT_NUMBER_FLAG       : BOOL(0);
      HIST_DATE_TIME_FLAG     : BOOL(1);
      HIST_SEQ_NBR_FLAG       : BOOL(2);
      HIST_INHIBIT_OVF_FLAG   : BOOL(3);
      EVENT_INHIBIT_OVF_FLAG  : BOOL(4);
      FILLER                  : FILL(5..7);
END;

TYPE LOG_RCD = PACKED RECORD
      LOG_FLAGS               : LOG_FLAGS_BFLD;
      NBR_STD_EVENTS          : UINT8;
      NBR_MFG_EVENTS          : UINT8;
      HIST_DATA_LENGTH        : UINT8;
      EVENT_DATA_LENGTH       : UINT8;
      NBR_HISTORY_ENTRIES     : UINT16;
      NBR_EVENT_ENTRIES       : UINT16;
END;

TABLE DIM_LOG_TBL = LOG_RCD;
```

Identifier	*Value*	*Definition*
LOG_FLAGS_BFLD		
EVENT_NUMBER_FLAG	FALSE	A common event number can not be maintained in the history and event logs.
	TRUE	A common event number can be maintained in the history and event logs.
HIST_DATE_TIME_FLAG	FALSE	Date and time can not be maintained in the history log.
	TRUE	Date and time can be maintained in the history log.
HIST_SEQ_NBR_FLAG	FALSE	A sequence number can not be transported through the history log.
	TRUE	A sequence number can be transported through the history log.
HIST_INHIBIT_OVF_FLAG	FALSE	History log is not capable of inhibiting overflow.
	TRUE	History log is capable of inhibiting overflow.
EVENT_INHIBIT_OVF_FLAG	FALSE	Event log is not capable of being blocked.
	TRUE	Event log is capable of being blocked.
TYPE LOG_RCD		
LOG_FLAGS		See **LOG_FLAGS_BFLD** above.
NBR_STD_EVENTS	0..255	Maximum number of octets in the set **EVENTS_SUPPORTED_TBL.-STD_EVENTS_SUPPORTED** (Table 72).

NBR_MFG_EVENTS	0..255	Maximum number of octets in the set **EVENTS_SUPPORTED_TBL.-MFG_EVENTS_SUPPORTED** (Table 72).
HIST_DATA_LENGTH	0..255	Maximum number of octets in the **HISTORY_LOG_DATA_TBL.-HISTORY_ARGUMENT** (Table 74).
EVENT_DATA_LENGTH	0..255	Maximum number of octets in the **EVENT_LOG_DATA_TBL.EVENT_ARGUMENT** (Table 76).
NBR_HISTORY_ENTRIES	0..65535	Maximum number of entries in the history log in **HISTORY_LOG_DATA_TBL** (Table 74).
NBR_EVENT_ENTRIES	0..65535	Maximum number of entries in the event log in. **EVENT_LOG_DATA_TBL** (Table 76).

9.8.2 TABLE 71 Actual log table

Table 71 Data description

ACT_LOG_TBL (Table 71) defines the actual size and capabilities of the history and event log decade.

TABLE ACT_LOG_TBL = DIM_LOG_TBL.LOG_RCD;

Identifier	*Value*	*Definition*
LOG_FLAGS_BFLD		
EVENT_NUMBER_FLAG		
	FALSE	A common event number is not maintained in the history and event logs.
	TRUE	A common event number is maintained in the history and event logs.
HIST_DATE_TIME_FLAG		
	FALSE	A date and time is not maintained in the history log.
	TRUE	A date and time is maintained in the history log.
HIST_SEQ_NBR_FLAG		
	FALSE	A sequence number is not transported through the history log.
	TRUE	A sequence number is transported through the history log.
HIST_INHIBIT_OVF_FLAG		
	FALSE	History log is not inhibiting new entries when an overflow condition exists.
	TRUE	History log is inhibiting new entries when an overflow condition exists.

EVENT_INHIBIT_OVF_FLAG		
	FALSE	Event log is not inhibiting new entries when an overflow condition exists.
	TRUE	Event log is inhibiting new entries when an overflow condition exists.
TYPE LOG_RCD		
LOG_FLAGS		See **LOG_FLAG_BFLD** above.
NBR_STD_EVENTS	0..255	Number of octets in the set **EVENTS_SUPPORTED_TBL.-STD_EVENTS_SUPPORTED** (Table 72).
NBR_MFG_EVENTS	0..255	Number of octets in the set **EVENTS_SUPPORTED_TBL.-MFG_EVENTS_SUPPORTED** (Table 72).
HIST_DATA_LENGTH	0..255	Number of octets in the **HISTORY_LOG_DATA_TBL.-HISTORY_ARGUMENT** (Table 74).
EVENT_DATA_LENGTH	0..255	Number of octets in the **EVENT_LOG_DATA_TBL.EVENT_ARGUMENT** (Table 76).
NBR_HISTORY_ENTRIES	0..65535	Actual maximum number of entries in the history log.
NBR_EVENT_ENTRIES	0..65535	Actual maximum number of entries in the event log.

9.8.3 TABLE 72 Events identification table

Table 72 Data description

EVENTS_ID_TBL (Table 72) contains the events that are supported by the end device.

```
TYPE EVENTS_SUPPORTED_RCD = PACKED RECORD
      STD_EVENTS_SUPPORTED     : SET(ACT_LOG_TBL.NBR_STD_EVENTS);
      MFG_EVENTS_SUPPORTED     : SET(ACT_LOG_TBL.NBR_MFG_EVENTS);
END;
TABLE EVENTS_ID_TBL = EVENTS_SUPPORTED_RCD;
```

Identifier	*Value*	*Definition*
EVENTS_SUPPORTED_RCD		
STD_EVENTS_SUPPORTED		This set variable indicates which of the standard events are supported in the event log. See Annex B for standard event codes. Event codes are represented by bits 0 through (8 * **ACT_LOG_TBL.NBR_STD_EVENTS**–1), with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.
MFG_EVENTS_SUPPORTED		This set variable indicates which of the manufacturer events are supported in the event log. Events are enabled by bits 0 through (8 * **ACT_LOG_TBL.NBR_MFG_EVENTS**–1), with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.

9.8.4 TABLE 73 History log control table

Table 73 Data description

HISTORY_LOG_CTRL_TBL (Table 73) defines the history log codes to be written to the history log. It also defines the specific procedures and or table writes that are to be acknowledged in the history log. For a specific procedure or table to be acknowledged, the following three independent tests shall all be true:

1) The procedure or table shall be used in the end device, per the **GEN_CONFIG_TBL** (Table 00);
2) The appropriate history log code shall be used, per this table;
3) The procedure or table shall be requested to be acknowledged, per this table.

```
TYPE HISTORY_CTRL_RCD = PACKED RECORD
      STD_EVENTS_MONITORED_FLAGS : SET(ACT_LOG_TBL.NBR_STD_EVENTS);
      MFG_EVENTS_MONITORED_FLAGS : SET(ACT_LOG_TBL.NBR_MFG_EVENTS);
      STD_TBLS_MONITORED_FLAGS   : SET(GEN_CONFIG_TBL.DIM_STD_TBLS_USED);
      MFG_TBLS_MONITORED_FLAGS   : SET(GEN_CONFIG_TBL.DIM_MFG_TBLS_USED);
      STD_PROC_MONITORED_FLAGS   : SET(GEN_CONFIG_TBL.DIM_STD_PROC_USED);
      MFG_PROC_MONITORED_FLAGS   : SET(GEN_CONFIG_TBL.DIM_MFG_PROC_USED);
END;

TABLE HISTORY_LOG_CTRL_TBL = HISTORY_CTRL_RCD;
```

Identifier	*Value*	*Definition*
HISTORY_CTRL_RCD		
STD_EVENTS_MONITORED_FLAGS		Bit position is linearly associated with corresponding standard event code.
	FALSE	Turns off event recording for associated event code.
	TRUE	Turns on event recording for associated event code.
MFG_EVENTS_MONITORED_FLAGS		Bit position is linearly associated with corresponding manufacturer event code.
	FALSE	Turns off event recording for associated event code.
	TRUE	Turns on event recording for associated event code.
STD_TBLS_MONITORED_FLAGS		Bit position is linearly associated with corresponding standard table number.
	FALSE	Turns off event recording for associated table.
	TRUE	Turns on event recording for associated table.
MFG_TBLS_MONITORED_FLAGS		Bit position is linearly associated with corresponding manufacturer table number.
	FALSE	Turns off event recording for associated table.
	TRUE	Turns on event recording for associated table.
STD_PROC_MONITORED_FLAGS		Bit position is linearly associated with corresponding standard procedure number.
	FALSE	Turns off event recording for associated procedure.
	TRUE	Turns on event recording for associated procedure.
MFG_PROC_MONITORED_FLAGS		Bit position is linearly associated with corresponding manufacturer procedure number.
	FALSE	Turns off event recording for associated procedure.
	TRUE	Turns on event recording for associated procedure.

9.8.5 TABLE 74 History log data table

Table 74 Data description

HISTORY_LOG_DATA_TBL (Table 74) provides the history log.

```
TYPE LIST_STATUS_BFLD = BIT FIELD OF UINT8
      ORDER                   : UINT(0..0);
      OVERFLOW_FLAG           : BOOL(1);
      LIST_TYPE               : UINT(2..2);
      INHIBIT_OVERFLOW_FLAG   : BOOL(3);
      FILLER                  : FILL(4..7);
END;

TYPE HISTORY_ENTRY_RCD = PACKED RECORD
      IF ACT_LOG_TBL.HIST_DATE_TIME_FLAG THEN
            HISTORY_TIME      : LTIME_DATE;
      END;
      IF ACT_LOG_TBL.EVENT_NUMBER_FLAG THEN
            EVENT_NUMBER      : UINT16;
```

```
      END;
      IF ACT_LOG_TBL.HIST_SEQ_NBR_FLAG THEN
            HISTORY_SEQ_NBR   : UINT16;
      END;
      USER_ID                 : UINT16;
      HISTORY_CODE            : TABLE_IDB_BFLD;
      HISTORY_ARGUMENT        : ARRAY[ACT_LOG_TBL.HIST_DATA_LENGTH]
                                    OF UINT8;
END;

TYPE HISTORY_LOG_RCD = PACKED RECORD
      HIST_FLAGS              : LIST_STATUS_BFLD;
      NBR_VALID_ENTRIES       : UINT16;
      LAST_ENTRY_ELEMENT      : UINT16;
      LAST_ENTRY_SEQ_NBR      : UINT32;
      NBR_UNREAD_ENTRIES      : UINT16;
      ENTRIES                 : ARRAY[ACT_LOG_TBL.NBR_HISTORY_ENTRIES]
                                    OF HISTORY_ENTRY_RCD;
END;

TABLE HISTORY_LOG_DATA_TBL = HISTORY_LOG_RCD;
```

Identifier	*Value*	*Definition*
LIST_STATUS_BFLD		
ORDER	0	The log is transported in ascending order (element N is older than element N+1).
	1	The log is transported in descending order (element N is newer than element N+1).
OVERFLOW_FLAG	FALSE	Overflow has not occurred.
	TRUE	An attempt was made to enter an event such that the number of unread entries would have exceeded the actual number of possible entries in the log.
LIST_TYPE	0	FIFO—as placed in log.
	1	Circular—as placed in log.
INHIBIT_OVERFLOW_FLAG		The same value as **ACT_LOG_TBL.-HIST_INHIBIT_OVF_FLAG** (Table 71).
TABLE_IDB_BFLD		
TBL_PROC_NBR	0..2047	Event number logged.
STD_VS_MFG_FLAG	FALSE	Event number is standard defined.
	TRUE	Event number is manufacturer defined.
SELECTOR		Not used.
HISTORY_ENTRY_RCD		
HISTORY_TIME		Date and time of history log entry.
EVENT_NUMBER	0..65535	Event number common to both the history and event logs.
HISTORY_SEQ_NBR	0..65535	Sequence number associated with the history log only.

USER_ID	0..65535	The user identification associated with this history log entry. It comes from the log in procedure or from a communication session initiation sequence. A **USER_ID** of zero means the end device initiated the event. A **USER_ID** of one means the event was manually initiated.
HISTORY_CODE		Event code logged. See **TABLE_IDB_BFLD** above. For standard event codes, refer to Annex C.
HISTORY_ARGUMENT		Argument associated with a specific entry. For standard event arguments, refer to Annex C.
HISTORY_LOG_RCD		
HIST_FLAGS		See **LIST_STATUS_BFLD** above.
NBR_VALID_ENTRIES	0..65535	Number of valid entries in the log. The range is from zero (meaning the log is empty) to the actual dimension of the log.
LAST_ENTRY_ELEMENT	0..65535	The array element number of the newest valid entry in the log.
LAST_ENTRY_SEQ_NBR	0..4 294 967 295	The sequence number of the newest valid entry in the log.
NBR_UNREAD_ENTRIES	0..65535	The number of entries in the log that have not yet been read. It is only changed through a procedure.
ENTRIES		Array of history log entries.

9.8.6 TABLE 75 Event log control table

Table 75 Data description

EVENT_LOG_CTRL_TBL (Table 75) defines the event log codes to be written to the event log. It also defines which specific procedures and/or table writes that are to be acknowledged in the event log. For a specific procedure or table to be acknowledged, the following three independent tests shall all be true:

1) The procedure or table shall be used in the end device, per the **GEN_CONFIG_TBL** (Table 00);
2) The appropriate event code shall be used, per Table 75;
3) The procedure or table shall be requested to be acknowledged, per Table 75.

This data structure is identical to the structure in **HISTORY_LOG_CTRL_TBL** (Table 73). Elements are defined below.

```
TABLE EVENT_LOG_CTRL_TBL = HISTORY_LOG_CTRL_TBL.HISTORY_CTRL_RCD;
```

Identifier	*Value*	*Definition*
EVENT_CTRL_RCD		
STD_EVENTS_MONITORED_FLAGS		Bit position is linearly associated with corresponding standard event code.
	FALSE	Turns off event recording for associated event code.
	TRUE	Turns on event recording for associated event code.
MFG_EVENTS_MONITORED_FLAGS		Bit position is linearly associated with corresponding manufacturer event code.
	FALSE	Turns off event recording for associated event code.
	TRUE	Turns on event recording for associated event code.
STD_TBLS_MONITORED_FLAGS		Bit position is linearly associated with corresponding standard table number.
	FALSE	Turns off event recording for associated table.
	TRUE	Turns on event recording for associated table.
MFG_TBLS_MONITORED_FLAGS		Bit position is linearly associated with corresponding manufacturer table number.
	FALSE	Turns off event recording for associated table.
	TRUE	Turns on event recording for associated table.
STD_PROC_MONITORED_FLAGS		Bit position is linearly associated with corresponding standard procedure number.
	FALSE	Turns off event recording for associated procedure.
	TRUE	Turns on event recording for associated procedure.
MFG_PROC_MONITORED_FLAGS		Bit position is linearly associated with corresponding manufacturer procedure number.
		Turns off event recording for associated procedure.
		Turns on event recording for associated procedure.

9.8.7 TABLE 76 Event log data table

Table 76 Data description

EVENT_LOG_DATA_TBL (Table 76) provides the event log.

```
TYPE LIST_STATUS_BFLD = BIT FIELD OF UINT8
      ORDER                    : UINT(0..0);
      OVERFLOW_FLAG            : BOOL(1);
      LIST_TYPE                : UINT(2..2);
      INHIBIT_OVERFLOW_FLAG    : BOOL(3);
      FILL                     : FILL(4..7);
END;
TYPE EVENT_ENTRY_RCD = PACKED RECORD
      EVENT_TIME               : LTIME_DATE;
      IF ACT_LOG_TBL.EVENT_NUMBER_FLAG THEN
            EVENT_NUMBER       : UINT16;
      END;
      EVENT_SEQ_NBR            : UINT16;
      USER_ID                  : UINT16;
      EVENT_CODE               : TABLE_IDB_BFLD;
      EVENT_ARGUMENT           : ARRAY[ACT_LOG_TBL.EVENT_DATA_LENGTH]
                                     OF UINT8;
END;
TYPE EVENT_LOG_RCD = PACKED RECORD
      EVENT_FLAGS              : LIST_STATUS_BFLD;
      NBR_VALID_ENTRIES        : UINT16;
      LAST_ENTRY_ELEMENT       : UINT16;
      LAST_ENTRY_SEQ_NBR       : UINT32;
      NBR_UNREAD_ENTRIES       : UINT16;
      ENTRIES                  : ARRAY[ACT_LOG_TBL.NBR_EVENT_ENTRIES] OF
                                     EVENT_ENTRY_RCD;
END;
TABLE EVENT_LOG_DATA_TBL = EVENT_LOG_RCD;
```

Identifier	*Value*	*Definition*
LIST_STATUS_BFLD		
ORDER	0	The log is transported in ascending order (element N is older than element N+1).
	1	The log is transported in descending order (element N is newer than element N+1).
OVERFLOW_FLAG	FALSE	Overflow has not occurred.
	TRUE	An attempt was made to enter an event such that the number of unread entries would have exceeded the actual number of possible entries in the log.
LIST_TYPE	0 1	FIFO—as placed in log. Circular—as placed in log.
INHIBIT_OVERFLOW_FLAG		The same value as **ACT_LOG_TBL.-EVENT_INHIBIT_OVF_FLAG.**

TABLE_IDB_BFLD		
TBL_PROC_NBR	0..2047	Event code number logged.
STD_VS_MFG_FLAG	FALSE TRUE	Event code is standard defined. Event code is manufacturer defined.
SELECTOR		Not used.
EVENT_ENTRY_RCD		
EVENT_TIME		Date and time of event log entry.
EVENT_NUMBER	0..65535	event number common to both the history and event logs.
EVENT_SEQ_NBR	0..65535	Sequence number associated with the Event Log only.
USER_ID	0..65535	The User ID associated with this event log entry. It comes from the log in procedure or from a communication session initiation sequence. A **USER_ID** of zero means the end device initiated the event. A **USER_ID** of one means the event was manually initiated.
EVENT_CODE		Event code logged. See **TABLE_IDB_BFLD** above. For standard event codes, refer to Annex B.
EVENT_ARGUMENT		Argument associated with a specific entry. Refer to Annex B or manufacturer for details.
EVENT_LOG_RCD		
ENVENT_FLAGS		See **LIST_STATUS_BFLD** above.
NBR_VALID_ENTRIES	0..65535	Number of valid entries in the log. The range is zero (meaning the log is empty) to the actual dimension of the log.
LAST_ENTRY_ELEMENT	0..65535	The array element number of the newest valid entry in the log.
LAST_ENTRY_SEQ_NBR	0..4 294 967 295	The sequence number of the newest valid entry in the log.
NBR_UNREAD_ENTRIES	0..65535	The number of entries in the log that have not yet been read. It is only changed through a procedure.
ENTRIES		Array of event log entries.

9.9 DECADE 80: User-defined tables

These tables provide the capability for the end device user to build custom tables using the tables available in an end device.

9.9.1 TABLE 80 Dimension user-defined tables function limiting table

Table 80 Data description

DIM_UDT_FUNC_LIM_TBL (Table 80) contains maximum values and control parameters for the user-defined tables. These values set up to six independent sets of user-defined tables. The length of user-defined tables is a function of the tables defined in the **GEN_CONFIG_TBL** (Table 00).

```
TYPE UDT_CTRL_BFLD = BIT FIELD OF UINT8;
      NBR_UDTS                : UINT(0..2);
      INSTANCE_FLAG           : BOOL(3);
      DATA_ACCESS_METHOD      : UINT(4..5);
      FILLER                  : FILL(6..7);
END;

TYPE UDT_FUNC_LIM_RCD = PACKED RECORD
      NBR_XFR_LIST_ITEMS      : UINT16;
      UDT_FUNC_CTRL           : UDT_CTRL_BFLD;
      MAX_INSTANCE            : UINT8;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.UDT_0_TBL_CNST THEN
            UDT_0_SIZE        : UINT32;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.UDT_1_TBL_CNST THEN
            UDT_1_SIZE        : UINT32;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.UDT_2_TBL_CNST THEN
            UDT_2_SIZE        : UINT32;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.UDT_3_TBL_CNST THEN
            UDT_3_SIZE        : UINT32;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.UDT_4_TBL_CNST THEN
            UDT_4_SIZE        : UINT32;
      END;
      IF GEN_CONFIG_TBL.STD_TBLS_USED.UDT_5_TBL_CNST THEN
            UDT_5_SIZE        : UINT32;
      END;
END;

TABLE DIM_UDT_FUNC_LIM_TBL = UDT_FUNC_LIM_RCD;
```

Identifier	*Value*	*Definition*
UDT_CTRL_BFLD		
NBR_UDTS	0..6	The maximum number of user-defined tables, which can be defined concurrently by the list of data items.
	7	Reserved.
INSTANCE_FLAG	FALSE	Multiple table instances are not supported. Multiple instances are not supported. Ignore **MAX_INSTANCE**.
	TRUE	Multiple table instances are supported.
DATA_ACCESS_METHOD		Variable to designate possible methods of selecting table entries for placement in a user-defined table.

	0	Only complete tables can be mapped into user tables. Partial tables cannot be mapped.
	1	Offset-count access method is supported.
	2	Index-count access method is supported.
	3	Access methods 2 and 3 are supported.

DIM_UDT_FUNC_LIM_TBL

NBR_XFR_LIST_ITEMS		
	0..65535	The maximum number of **UINT16** entries used to make the list.
UDT_FUNC_CTRL		These are the limiting capabilities for user-defined tables selections. This defines the maximum number of user-defined tables supported in this decade and the data access method, as defined by **UDT_CTRL_BFLD**.
MAX_INSTANCE	0	Unused
	1..255	Maximum number of table sets supported by this end device. If **INSTANCE_FLAG** is FALSE, ignore.
UDT_0_SIZE	0..4,294,967,295	Maximum number of octets used to size **UDT_0_TBL** (Table 84)
UDT_1_SIZE	0..4,294,967,295	Maximum number of octets used to size **UDT_1_TBL** (Table 85)
UDT_2_SIZE	0..4 294 967 295	Maximum number of octets used to size **UDT_2_TBL** (Table 86)
UDT_3_SIZE	0..4 294 967 295	Maximum number of octets used to size **UDT_3_TBL** (Table 87)
UDT_4_SIZE	0..4 294 967 295	Maximum number of octets used to size **UDT_4_TBL** (Table 88)
UDT_5_SIZE	0..4 294 967 295	Maximum number of octets used to size **UDT_5_TBL** (Table 89)

9.9.2 TABLE 81 Actual user-defined tables function limiting table

Table 81 Data description

ACT_UDT_FUNC_LIM_TBL (Table 81) contains actual limiting values for the user-defined tables. These parameters define up to six independent sets of user-defined tables definitions. The length of each user-defined table is a function of the tables defined in the **GEN_CONFIG_TBL** (Table 00).

```
TABLE ACT_UDT_FUNC_LIM_TBL = DIM_UDT_FUNC_LIM_TBL.UDT_FUNC_LIM_RCD;
```

Identifier	*Value*	*Definition*
UDT_CTRL_BFLD		
NBR_UDTS	0..6	The actual number of user-defined tables, which are defined concurrently by the list of data items.
	7	Reserved.
INSTANCE_FLAG	FALSE	Multiple table sets are not used. Multiple instances are not supported. Ignore **MAX_INSTANCE**.
	TRUE	Multiple table sets are in use.
DATA_ACCESS_METHOD		Designate the actual method of selecting table entries for placement in a user-defined table.
	0	Complete tables are mapped into user tables. Partial table are not be mapped.
	1	Offset-count access method is used.
	2	Index-count access method is used.
	3	Not valid.
UDT_FUNC_LIM_RCD		
NBR_XFR_LIST_ITEMS	0.. 65535	The actual number of **UINT16** entries used in the list.
UDT_FUNC_CTRL		This is the actual method used for user-defined tables selections. This identifies the actual number of user-defined tables and the data access method used in this decade as defined by **UDT_FLAGS_BFLD.**
MAX_INSTANCE	0	Unused.
	1..255	Actual number of table sets used by this end device. If **INSTANCE_FLAG** is FALSE ignore.
UDT_0_SIZE	0..4 294 967 295	Actual number of octets used to size **UDT_0_TBL** (Table 84)
UDT_1_SIZE	0..4 294 967 295	Actual number of octets used to size **UDT_1_TBL** (Table 85)
UDT_2_SIZE	0..4 294 967 295	Actual number of octets used to size **UDT_2_TBL** (Table 86)
UDT_3_SIZE	0..4 294 967 295	Actual number of octets used to size **UDT_3_TBL** (Table 87)
UDT_4_SIZE	0..4 294 967 295	Actual number of octets used to size **UDT_4_TBL** (Table 88)
UDT_5_SIZE	0..4 294 967 295	Actual number of octets used to size **UDT_5_TBL** (Table 89)

9.9.3 TABLE 82 User-defined tables list table

Table 82 Data description

UDT_LIST_TBL (Table 82) defines data elements used in the generation of user-defined tables for this decade

```
TYPE SOURCE_ITEM_RCD = PACKED RECORD
      TABLE_ID                : TABLE_IDB_BFLD;
      IF ACT_UDT_FUNC_LIM_TBL.INSTANCE_FLAG THEN
            TABLE_INSTANCE     : UINT16;
      END;
      CASE ACT_UDT_FUNC_LIM_TBL.DATA_ACCESS_METHOD OF
            0: NO_DATA         : NIL;
            1: OFFSET          : UINT16;
            2: INDEX           : ARRAY[TABLE_ID.SELECTOR] OF UINT16;
      END;
      COUNT                    : UINT16;
END;

TYPE UDT_LIST_SEL_RCD = PACKED RECORD
      UDT_LIST : ARRAY[ACT_UDT_FUNC_LIM_TBL.NBR_XFR_LIST_ITEMS]
                  OF UINT16;
END;

TABLE UDT_LIST_TBL = UDT_LIST_SEL_RCD;
```

Identifier	*Value*	*Definition*
TABLE_IDB_BFLD		
TBL_PROC_NBR	0..2047	Information source table number. NOTE—The corresponding implementation bit shall be set in **GEN_CONFIG_TBL** (Table 00).
STD_VS_MFG_FLAG		Identifies the data item as originating from either the standard table or a manufacturer table.
	FALSE	Standard table.
	TRUE	Manufacturer table.
SELECTOR	0	When **DATA_ACCESS_METHOD** is zero the table access method is used.
	0..15	When the **DATA_ACCESS_METHOD** is one this field is the most significant 4 b (bits 16..19) of the offset to the data element required to locate the data item in a table.
	0..15	When **DATA_ACCESS_METHOD** is two, this number of indices is used to select an entry in a table.

SOURCE_ITEM_RCD		Structure of a single entry in the **UDT_LIST** array.
TABLE_INSTANCE	0..254 255..65535	Table set number. Not used.
OFFSET	0..65535	This field holds the least significant 16 b of the offset to the data element required to locate the data item in a table.
INDEX		This is an array of indices, which combine to define the index to an item entry in a table.
COUNT	0..65535	Length of data item(s) in octets. When the value is zero it is the end of the list.
UDT_LIST_SEL_RCD		
UDT_LIST		An array holding **ACT_UDT_FUNC_LIM_TBL. NBR_XFR_LIST_ITEM UINT16** quantities. This space is mapped using S**OURCE_ITEM_RCD** into variable length records that select table or partial tables for access via the user-defined tables.

9.9.4 TABLE 83 User-defined tables selection table

Table 83 Data description

UDT_SEL_TBL (Table 83) specifies what data elements are used in user tables defined for this decade. Each array entry in this table corresponds to a specific table in this decade following this table. For example, the third entry in this array corresponds to Table 83 + 3 = Table 86.

```
TYPE UDT_SET_RCD = PACKED RECORD
      FIRST_ITEM_NBR    : UINT16;
      LAST_ITEM_NBR     : UINT16;
END;

TYPE UDT_DATA_SETS_RCD = PACKED RECORD

      UDT_DATA_SETS     : ARRAY[ACT_UDT_FUNC_LIM_TBL.NBR_UDTS]
                              OF UDT_SET_RCD;
END;

TABLE UDT_SEL_TBL = UDT_DATA_SETS_RCD;
```

9.9.5 TABLE 84 User-defined table zero table

Table 84 Data description

This table is defined by the items in table **UDT_LIST_TBL** (Table 82) selected by the first entry in table **UDT_SEL_TBL** (Table 83).

```
TYPE UDT_0_RCD = PACKED RECORD
      UDT_0_DATA  : = ARRAY[ACT_UDT_FUNC_LIM_TBL.UDT_0_SIZE] OF UINT8;
END;
```

Identifier	*Value*	*Definition*
UDT_SETS_RCD		
FIRST_ITEM_NBR	0..65535	This is the item number in **UDT_LIST_TBL** (Table 82) specifying the first item in the user-defined tables selected by this entry.
LAST_ITEM_NBR	0..65535	This is the item number in **UDT_LIST_TBL** (Table 82) specifying the last item in the user-defined tables selected by this entry.
UDT_DATA_SETS_RCD		
UDT_DATA_SETS		Array containing up to **ACT_UDT_FUNC_LIM_TBL.NBR_UDTS** (Table 81) entries. Each entry defines the source selections for a user destination user-defined table.

```
TABLE UDT_0_TBL = UDT_0_RCD;
```

Identifier	*Value*	*Definition*
UDT_0_RCD		Array containing **ACT_UDT_FUNCT_LIM_TBL.UDT_0_SIZE** octets holding data defined by the first entry in the list defined in **UDT_SEL_TBL** (Table 83).

9.9.6 TABLE 85 User-defined table one table

Table 85 Data description

This table is defined by the items in table **UDT_LIST_TBL** (Table 82) selected by the first entry in table **UDT_SEL_TBL** (Table 83).

```
TYPE UDT_1_RCD = PACKED RECORD
      UDT_1_DATA  : ARRAY[ACT_UDT_FUNC_LIM_TBL.UDT_1_SIZE] OF UINT8;
END;

TABLE UDT_1_TBL = UDT_1_RCD;
```

Identifier	*Value*	*Definition*
UDT_1_RCD		Array containing **ACT_UDT_FUNCT_LIM_TBL.UDT_1_SIZE** octets holding data defined by the second entry in the list defined in **UDT_SEL_TBL** (Table 83).

9.9.7 TABLE 86 User-defined table two table

Table 86 Data description

This table is defined by the items in table **UDT_LIST_TBL** (Table 82) selected by the first entry in table **UDT_SEL_TBL** (Table 83).

```
TYPE UDT_2_RCD = PACKED RECORD
      UDT_2_DATA  : ARRAY[ACT_UDT_FUNC_LIM_TBL.UDT_2_SIZE] OF UINT8;
END;

TABLE UDT_2_TBL = UDT_2_RCD;
```

Identifier	*Value*	*Definition*
UDT_2_RCD		Array containing **ACT_UDT_FUNCT_LIM_TBL.UDT_2_SIZE** octets holding data defined by the third entry in the list defined in **UDT_SEL_TBL** (Table 83).

9.9.8 TABLE 87 User-defined table three table

Table 87 Data description

This table is defined by the items in table **UDT_LIST_TBL** (Table 82) selected by the first entry in table **UDT_SEL_TBL** (Table 83).

```
TYPE UDT_3_RCD = PACKED RECORD
      UDT_3_DATA  : ARRAY[ACT_UDT_FUNC_LIM_TBL.UDT_3_SIZE] OF UINT8;
END;

TABLE UDT_3_TBL = UDT_3_RCD;
```

Identifier	*Value*	*Definition*
UDT_3_RCD		Array containing **ACT_UDT_FUNCT_LIM_TBL.UDT_3_SIZE** octets holding data defined by the fourth entry in the list defined in **UDT_SEL_TBL** (Table 83).

9.9.9 TABLE 88 User-defined table four table

Table 88 Data description

This table is defined by the items in table **UDT_LIST_TBL** (Table 82) selected by the first entry in table **UDT_SEL_TBL** (Table 83).

```
TYPE UDT_4_RCD = PACKED RECORD
      UDT_4_DATA  : ARRAY[ACT_UDT_FUNC_LIM_TBL.UDT_4_SIZE] OF UINT8;
END;

TABLE UDT_4_TBL = UDT_4_RCD;
```

UDT_4_RCD		Array containing **ACT_UDT_FUNCT_LIM_TBL.UDT_4_SIZE** octets holding data defined by the fifth entry in the list defined in **UDT_SEL_TBL** (Table 83).

9.9.10 TABLE 89 User-defined table five table

Table 89 Data description

This table is defined by the items in table **UDT_LIST_TBL** (Table 82) selected by the first entry in table **UDT_SEL_TBL** (Table 83).

```
TYPE UDT_5_RCD = PACKED RECORD

      UDT_5_DATA  : ARRAY[ACT_UDT_FUNC_LIM_TBL.UDT_5_SIZE] OF UINT8;
END;

TABLE UDT_5_TBL = UDT_5_RCD;
```

UDT_5_RCD	Array containing **ACT_UDT_FUNCT_LIM_TBL.UDT_5_SIZE** octets holding data defined by the sixth entry in the list defined in **UDT_SEL_TBL** (Table 83).

Annex A

(normative)

ID Codes for meter equipment manufacturers

ABB	ABB
AMCO	American Meter Company
APTH	Aptech, Inc.
BMI	Badger Meter, Inc.
END	Endacom 2000 Inc.
GE	General Electric
ITD	International Teldata Corporation
JEM	Scientific Columbus Company
L&G	Landis & Gyr
MM	Master Meter, Inc.
ND	Nertec Design Inc.
PSI	Process Systems Inc.
SCH	Schlumberger
TEMP	For new manufacturers who are not yet included in this list
WEXL	WeXL

Annex B

(normative)

History and event log codes

Code	*Event*	*Argument*
00	No event	None
01	Primary power down	None
02	Primary power up	None
03	Time changed (old time)	None. Time tag if used equals old time.
04	Time changed (new time)	None. Time tag if used equals new time.
05	Time changed (old time)	Old time in **STIME_DATE** format.
06	Time changed (new time)	New time in **STIME_DATE** format.
07	End device accessed for read	None
08	End device accessed for write	None
09	Procedure invoked	Bit field of procedure number and standard/manufacturer flag (**UINT16**) **TABLE_IDC_BFLD** format.
10	Table written to	Bit field of table number and standard/manufacturer flag (**UINT16**) **TABLE_IDA_BFLD** format.
11	End device programmed	None
12	Communication terminated normally	None
13	Communication terminated abnormally	None
14	Reset list pointers	**LIST** (**INT8**) {Reference procedure 4, 9.1.8.1.5}
15	Update list pointers	**LIST** (**UINT8**) (Reference procedure 5, 9.1.8.1.6}
16	History log cleared	None
17	History log pointers updated	Value of procedure parameter (**UINT16**)
18	Event log cleared	None
19	Event log pointers updated	Value of procedure parameter (**UINT16**)
20	Demand reset occurred	None
21	Self read occurred	None
22	Daylight Savings Time On	None
23	Daylight Savings Time Off	None
24	Season change	New season number (**UINT8**)
25	Rate change	New rate (**UINT8**)
26	Special schedule activation	New special schedule (**UINT8**)
27	Tier switch change	New current tier (**UINT8**) followed by new demand tier (**UINT8**)
28	Pending table activation	Table number (**TABLE_IDA_BFLD**).
29	Pending table clear	Table number (**TABLE_IDA_BFLD**). Table removed from end device prior to activation.

Annex C

(normative)

Default sets for decade tables

The following values are assigned to the decade and decade+1 table values based upon the value of the TABLE 00 variable **DEFAULT_SET_USED**.

TABLE 11				
DEFAULT_SET_USED VARIABLE	1	2	3	4
PF_EXCLUDE_FLAG	0	0	1	0
RESET_EXCLUDE_FLAG	0	1	1	0
BLOCK_DEMAND_FLAG	0	1	1	1
SLIDING_DEMAND_FLAG	0	0	0	0
THERMAL_DEMAND_FLAG	0	0	0	0
SET1_PRESENT_FLAG	0	0	0	0
SET2_PRESENT_FLAG	0	0	0	0
NBR_UOM_ENTRIES	0	0	0	0
NBR_DEMAND_CTRL_ENTRIES	0	1	3	1
DATA_CTRL_LENGTH	1	1	1	1
NBR_DATA_CTRL_ENTRIES	1	2	2	2
NBR_CONSTANTS_ENTRIES	1	1	1	1
CONSTANTS_SELECTOR	2	2	2	2
NBR_SOURCES	1	2	2	2

TABLE 21			
SEASON_INFO_FIELD_FLAG	0	0	0
DATE_TIME_FIELD_FLAG	0	0	0
DEMAND_RESET_CTR_FLAG	0	1	1
DEMAND_RESET_LOCK_FLAG	0	1	1
CUM_DEMAND_FLAG	0	0	0
CONT_CUM_DEMAND_FLAG	0	0	0
TIME_REMAINING_FLAG	0	0	0

TABLE 21 *(CONTINUED)*			
SELF_READ_INHIBIT_OVERFLOW_FLAG	0	0	1
SELF_READ_SEQ_NBR_FLAG	0	0	1
DAILY_SELF_READ_FLAG	0	0	0
WEEKLY_SELF_READ_FLAG	0	0	0
SELF_READ_DEMAND_RESET	0	0	1
NBR_SELF_READS	0	0	1
NBR_SUMMATIONS	1	1	1
NBR_DEMANDS	0	1	1
NBR_COIN_VALUES	0	0	0
NBR_OCCUR	0	1	1
NBR_TIERS	0	0	3
NBR_PRESENT_DEMANDS	1	1	1
NBR_PRESENT_VALUES	1	1	1

TABLE 31				
ON_TIME_FLAG	0	0	0	0
OFF_TIME_FLAG	0	0	0	0
HOLD_TIME_FLAG	0	0	0	0
NBR_DISP_SOURCES	3	4	40	0
WIDTH_DISP_SOURCES	1	1	1	0
NBR_PRI_DISP_LIST_ITEMS	9	12	60	0
NBR_PRI_DISP_LISTS	3	3	3	0
NBR_SEC_DISP_LIST_ITEMS	3	3	3	0
NBR_SEC_DISP_LISTS	1	1	1	1

TABLE 41				
NBR_PASSWORDS	1	1	1	1
PASSWORD_LEN	20	20	20	20
NBR_KEYS	1	1	1	1
KEY_LEN	4	4	4	4
NBR_PERM_USED	1	1	1	1

TABLE 51				
TOU_SELF_READ_FLAG	0	0	1	0
SEASON_SELF_READ_FLAG	0	0	1	0
SEASON_DEMAND_RESET_FLAG	0	0	0	0
SEASON_CHNG_ARMED_FLAG	0	0	0	0
SORT_DATES_FLAG	0	0	1	0
ANCHOR_DATE_FLAG	0	0	0	0
CAP_DST_AUTO_FLAG	0	0	0	0
SEPARATE_WEEKDAYS_FLAG	0	0	0	0
SEPARATE_SUM_DEMANDS_FLAG	0	0	0	0
SORT_TIER_SWITCHES_FLAG	0	0	1	0
CAP_TM_ZN_OFFSET_FLAG	0	0	0	0
NBR_SEASONS	0	0	2	0
NBR_SPECIAL_SCHED	0	0	1	0
NBR_NON_RECURR_DATES	0	0	10	0
NBR_RECURR_DATES	0	0	10	0
NBR_TIER_SWITCHES	0	0	9	0
CALENDAR_TBL_SIZE	0	0	131	0

TABLE 61				
LP_SET1_INHIBIT_OVF_FLAG	0	0	0	1

TABLE 61 ***(CONTINUED)***				
LP_SET2_INHIBIT_OVF_FLAG	0	0	0	0
LP_SET3_INHIBIT_OVF_FLAG	0	0	0	0
LP_SET4_INHIBIT_OVF_FLAG	0	0	0	0
BLK_END_READ_FLAG	0	0	0	0
BLK_END_PULSE_FLAG	0	0	0	0
SCALAR_DIVISOR_FLAG_SET1	0	0	0	1
SCALAR_DIVISOR_FLAG_SET2	0	0	0	0
SCALAR_DIVISOR_FLAG_SET3	0	0	0	0
SCALAR_DIVISOR_FLAG_SET4	0	0	0	0
EXTENDED_INT_STATUS_FLAG	0	0	0	1
SIMPLE_INT_STATUS_FLAG	0	0	0	0
INV_UINT8_FLAG	0	0	0	0
INV_UINT16_FLAG	0	0	0	1
INV_UINT32_FLAG	0	0	0	0
INV_INT8_FLAG	0	0	0	0
INV_INT16_FLAG	0	0	0	0
INV_INT32_FLAG	0	0	0	0
INV_NI_FMAT1_FLAG	0	0	0	0
INV_NI_FMAT2_FLAG	0	0	0	0
LP_MEMORY_LEN	0	0	0	16384
NBR_BLKS_SET1	0	0	0	80
NBR_BLK_INTS_SET1	0	0	0	96
NBR_CHNS_SET1	0	0	0	1
MAX_INT_TIME_SET1	0	0	0	15
NBR_BLKS_SET2	0	0	0	0
NBR_BLK_INTS_SET2	0	0	0	0
NBR_CHNS_SET2	0	0	0	0

TABLE 61 ***(CONTINUED)***				
MAX_INT_TIME_SET2	0	0	0	0
NBR_BLKS_SET3	0	0	0	0
NBR_BLK_INTS_SET3	0	0	0	0
NBR_CHNS_SET3	0	0	0	0
MAX_INT_TIME_SET3	0	0	0	0
NBR_BLKS_SET4	0	0	0	0
NBR_BLK_INTS_SET4	0	0	0	0
NBR_CHNS_SET4	0	0	0	0
MAX_INT_TIME_SET4	0	0	0	0

TABLE 71				
EVENT_NUMBER_FLAG	1	1	1	1
HIST_DATE_TIME_FLAG	1	1	1	1
HIST_SEQ_NBR_FLAG	0	0	0	0
HIST_INHIBIT_OVF_FLAG	0	0	0	0
EVENT_INHIBIT_OVF_FLAG	1	1	1	1
NBR_STD_EVENTS	4	4	4	4
NBR_MFG_EVENTS	0	0	0	0
HIST_DATA_LENGTH	2	2	2	2
EVENT_DATA_LENGTH	2	2	2	2
NBR_HISTORY_ENTRIES	100	100	100	100
NBR_EVENT_ENTRIES	100	100	100	100

Annex D

(normative)

Indices for partial table access

The following values are for use with the partial table access method identified as index-count.

TABLE 000—GEN_CONFIG_TBL								
FORMAT_CONTROL_1	000							
FORMAT_CONTROL_2	001							
FORMAT_CONTROL_3	002							
MANUFACTURER	003							
NAMEPLATE_TYPE	004							
DEFAULT_SET_USED	005							
MAX_PROC_PARM_LENGTH	006							
MAX_RESP_DATA_LEN	007							
STD_VERSION_NO	008							
STD_REVISION_NO	009							
DIM_STD_TBLS_USED	010							
DIM_MFG_TBLS_USED	011							
DIM_STD_PROC_USED	012							
DIM_MFG_PROC_USED	013							
DIM_MFG_STATUS_USED	014							
NBR_PENDING	015							
STD_TBLS_USED	016							
MFG_TBLS_USED	017							
STD_PROC_USED	018							
MFG_PROC_USED	019							
STD_TBLS_WRITE	020							
MFG_TBLS_WRITE	021							

TABLE 001—GENERAL_MFG_ID_TBL								
MANUFACTURER	000							
ED_MODEL	001							
HW_VERSION_NUMBER	002							
HW_REVISION_NUMBER	003							
FW_VERSION_NUMBER	004							
FW_REVISION_NUMBER	005							
MFG_SERIAL_NUMBER	006							

TABLE 002—DEVICE_NAMEPLATE_TBL								
(Gas device structure)								
G_ED_TYPE	000							
G_MAX_PRESS	001							
G_MAX_PRESS	001	000						
G_UOM_PRESS	001	001						
G_MAX_FLOW	002							
G_MAX_FLOW	002	000						
G_UOM_FLOW	002	001						
G_GEAR_PIPE_SIZE	003							
G_COMPENSATION	004							
(Water device structure)								
W_WATER_DEVICE_BFLD	000							
(Electric device structure)								
E_Kh	000							
E_Kt	001							
E_INPUT_SCALAR	002							
E_ED_CONFIG(ARRAY)	003							
E_ED_CONFIG(ITEM)	003	[n]						
E_ELEMENTS	004							
E_VOLTS	005							
E_AMPS	006							

E_CLASS_MAX_AMPS	006	000						
E_TA	006	001						

TABLE 003—ED_MODE_STATUS_TBL								
ED_MODE	000							
ED_STD_STATUS1	001							
ED_STD_STATUS2	002							
ED_MFG_STATUS	003							
ED_MFG_STATUS	003							
ED_MFG_STATUS (ITEM)	003	**000**						

TABLE 004—PENDING_STATUS_TBL								
STANDARD_PENDING	000							
MANUFACT_PENDING	001							
LAST_ACTIVATION_DATE_TIME	002							
NBR_PENDING_ACTIVATION	003							
PENDING_TABLES(ARRAY)	004							
PENDING_TABLES(ITEM)	004	[n]						
EVENT	004	[n]	000					
EVENT_SELECTOR	004	[n]	000	000				
EVENT_STORAGE(ARRAY)	004	[n]	000	001				
EVENT_STORAGE(ITEM)	004	[n]	000	001	[n]			
TABLE_SELECTOR	004	[n]	001					

TABLE 005—DEVICE_IDENT_TBL								
IDENTIFICATION	000							

TABLE 006—UTIL_INFO_TBL								
OWNER_NAME	000							
UTILITY_DIV	001							

SERVICE_POINT_ID	002							
ELEC_ADDR	003							
DEVICE_ID	004							
UTIL_SER_NO	005							
CUSTOMER_ID	006							
COORDINATE_1	007							
COORDINATE_2	008							
COORDINATE_3	009							
TARIFF_ID	010							
EX1_SW_VENDOR	011							
EX1_SW_VERSION_NUMBER	012							
EX1_SW_REVISION_NUMBER	013							
EX2_SW_VENDOR	014							
EX2_SW_VERSION_NUMBER	015							
EX2_SW_REVISION_NUMBER	016							
PROGRAMMER_NAME	017							
MISC_ID	018							

TABLE 007—PROC_INITIATE_TBL								
PROC	000							
SEQ_NBR	001							
PARM	002							

TABLE 008—PROC_RESPONSE_TBL								
PROC	000							
SEQ_NBR	001							
RESULT_CODE	002							
RESP_DATA	003							

TABLE 010—DIM_SOURCES_LIM_TBL								
SOURCE_FLAGS	000							
NBR_UOM_ENTRIES	001							
NBR_DEMAND_CTRL_ENTRIES	002							
DATA_CTRL_LENGTH	003							
NBR_DATA_CTRL_ENTRIES	004							
NBR_CONSTANT_ENTRIES	005							
CONSTANTS_SELECTOR	006							
NBR_SOURCES	007							

TABLE 011—ACT_SOURCES_LIM_TBL								
SOURCE_FLAGS	000							
NBR_UOM_ENTRIES	001							
NBR_DEMAND_CTRL_ENTRIES	002							
DATA_CTRL_LENGTH	003							
NBR_DATA_CTRL_ENTRIES	004							
NBR_CONSTANT_ENTRIES	005							
CONSTANTS_SELECTOR	006							
NBR_SOURCES	007							

TABLE 012—UOM_ENTRY_TBL								
UOM_ENTRY(ARRAY)	000							
UOM_ENTRY(ITEM)	000	[n]						

TABLE 013—DEMAND_CONTROL_TBL								
RESET_EXCLUSION	000							
P_FAIR_RECOHNTN_TM	001							
P_FAIL_EXCLUSION	002							
COLD_LOAD_PICKUP	003							
INTERVAL_VALUE(ARRAY)	004							

INTERVAL_VALUE(ITEM)	004	[n]						
SUB_INT	004	[n]	000					
INT_MULTIPLIER	004	[n]	001					
INT_LENGTH	004	[n]	002					

TABLE 014—DATA_CONTROL_RCD								
SOURCES_ID(ARRAY)	000							
SOURCES_ID(ITEM)	000	[n]						
SOURCE_ID(ARRAY)	000	[n]	000					
SOURCE_ID(ITEM)	000	[n]	000	[n]				

TABLE 015—CONSTANTS_TBL								
SELECTION(ARRAY)	000							
SELECTION(ITEM)	000	[n]						
(GAS DEVICE—AGA3)								
GAS_DP_PARM	000	[n]	000					
GAS_DP_ZERO	000	[n]	000	000				
GAS_DP_FULLSCALE	000	[n]	000	001				
GAS_DP_SHUTOFF	000	[n]	001					
GAS_SHUTOFF	000	[n]	001	000				
GAS_PRESS_PARM	000	[n]	002					
GAS_PRESS_ZERO	000	[n]	002	000				
GAS_PRESS_FULLSCALE	000	[n]	002	001				
BASE_PRESSURE	000	[n]	002	002				
GAS_AGA3_CORR_FCTR	000	[n]	003					
AUX_CORR_FCTR	000	[n]	003	000				
GAS_AGA3_CORR	000	[n]	003	001				
PIPE_ORIF_DIA	000	[n]	003	002				
PIPE_DIA	000	[n]	003	002	000			
ORIF_DIA	000	[n]	003	002	001			

TAP_UP_DN	000	[n]	003	003				
GAS_PRESS_PARM	000	[n]	003	004				
GAS_PRESS_ZERO	000	[n]	003	004	000			
GAS_PRESS_FULLSCALE	000	[n]	003	004	001			
BASE_PRESSURE	000	[n]	003	004	002			
GAS_TEMP_PARM	000	[n]	003	005				
GAS_TEMP_ZERO	000	[n]	003	005	000			
GAS_TEMP_FULLSCALE	000	[n]	003	005	001			
BASE_TEMP	000	[n]	003	005	002			
GAS_ENERGY	000	[n]	004					
GAS_ENERGY_ZERO	000	[n]	004	000				
GAS_ENERGY_FULL	000	[n]	004	001				
(Gas device—AGA7)								
GAS_AGA7_CORR	000	[n]	000					
GAS_PRESS_PARM	000	[n]	000	000				
GAS_PRESS_ZERO	000	[n]	000	000	000			
GAS_PRESS_FULLSCALE	000	[n]	000	000	001			
BASE_PRESSURE	000	[n]	000	000	002			
GAS_TEMP_PARM	000	[n]	000	001				
GAS_TEMP_ZERO	000	[n]	000	001	000			
GAS_TEMP_FULLSCALE	000	[n]	000	001	001			
BASE_TEMP	000	[n]	000	001	002			
AUX_CORR_FCTR	000	[n]	000	002				
GAS_AGA7_CORR	000	[n]	000	003				
GAS_ENERGY	000	[n]	001					
GAS_ENERGY_ZERO	000	[n]	001	000				
GAS_ENERGY_FULL	000	[n]	001	001				
(Electric device)								
MULTIPLIER	000	[n]	000					
OFFSET	000	[n]	001					

SET1_CONSTANTS	000	[n]	002					
SET_FLAGS	000	[n]	002	000				
RATIO_F1	000	[n]	002	001				
RATIO_P1	000	[n]	002	002				
SET2_CONSTANTS	000	[n]	003					
SET_FLAGS	000	[n]	003	000				
RATIO_F1	000	[n]	003	001				
RATIO_P1	000	[n]	003	002				

TABLE 016—SOURCES_TBL								
SOURCES_LINK(ARRAY)	000							
SOURCES_LINK(ITEM)	000	[n]						

TABLE 020—DIM_REGS_TBL								
REG_FUNC1_FLAGS	000							
REG_FUNC2_FLAGS	001							
NBR_SELF_READS	002							
NBR_SUMMATIONS	003							
NBR_DEMANDS	004							
NBR_COIN_VALUES	005							
NBR_OCCUR	006							
NBR_TEIRS	007							
NBR_PRESENT_DEMANDS	008							
NBR_PRESENT_VALUES	009							

TABLE 021—ACT_REGS_TBL								
REG_FUNC1_FLAGS	000							
REG_FUNC2_FLAGS	001							
NBR_SELF_READS	002							
NBR_SUMMATIONS	003							

NBR_DEMANDS	004							
NBR_COIN_VALUES	005							
NBR_OCCUR	006							
NBR_TEIRS	007							
NBR_PRESENT_DEMANDS	008							
NBR_PRESENT_VALUES	009							

TABLE 022—DATA_SELECTION_TBL								
SUMMATION_SELECT(ARRAY)	000							
SUMMATION_SELECT(ITEM)	000	[n]						
DEMAND_SELECT(ARRAY)	001							
DEMAND_SELECT(ITEM)	001	[n]						
MIN_OR_MAX_FLAGS	002							
COINCIDENT_SELECT(ARRAY)	003							
COINCIDENT_SELECT(ITEM)	003	[n]						
COIN_DEMAND_ASSOC(ARRAY)	004							
COIN_DEMAND_ASSOC(ITEM)	004	[n]						

TABLE 23—CURRENT_REG_DATA_TBL								
NMB_DEMAND_RESETS	000							
TOT_DATA_BLOCK	001							
SUMMATIONS(ARRAY)	001	000						
SUMMATIONS(ITEM)	001	000	[n]					
DEMANDS(ARRAY)	001	001						
DEMANDS(ITEM)	001	001	[n]					
EVENT_TIME(ARRAY)	001	001	[n]	000				
EVENT_TIME(ITEM)	001	001	[n]	000	[n]			
CUM_DEMAND	001	001	[n]	001				
CONT_CUM_DEMAND	001	001	[n]	002				
DEMAND(ARRAY)	001	001	[n]	003				

DEMAND(ITEM)	001	001	[n]	003	[n]			
COINCIDENTS(ARRAY)	001	002						
COINCIDENTS(ITEM)	001	002	[n]					
COINCIDENT_VALUES(ARRAY)	001	002	[n]	000				
COINCIDENT_VALUES(ITEM)	001	002	[n]	000	[n]			
TIER_DATA_BLOCK(ARRAY)	002							
TIER_DATA_BLOCK(ITEM)	002	[n]						
SUMMATIONS (ARRAY)	002	[n]	000					
SUMMATIONS(ITEM)	002	[n]	000	[n]				
DEMANDS(ARRAY)	002	[n]	001					
DEMANDS(ITEM)	002	[n]	001	[n]				
EVENT_TIME(ARRAY)	002	[n]	001	[n]	000			
EVENT_TIME(ITEM)	002	[n]	001	[n]	000	[n]		
CUM_DEMAND	002	[n]	001	[n]	001			
CONT_CUM_DEMAND	002	[n]	001	[n]	002			
DEMAND(ARRAY)	002	[n]	001	[n]	003			
DEMAND(ITEM)	002	[n]	001	[n]	003	[n]		
COINCIDENTS(ARRAY)	002	[n]	002					
COINCIDENTS(ITEM)	002	[n]	002	[n]				
COINCIDENT_VALUES(ARRAY)	002	[n]	002	[n]	000			
COINCIDENT_VALUES(ITEM)	002	[n]	002	[n]	000	[n]		

TABLE 24— PREVIOUS_SEASON_DATA_TBL								
REGISTER_INFO	000							
END_DATE_TIME	000	000						
SEASON	000	001						
PREV_SEASON_REG_DATA	001							
NMB_DEMAND_RESETS	001	000						
TOT_DATA_BLOCK	001	001						
SUMMATIONS(ARRAY)	001	001	000					

SUMMATIONS(ITEM)	001	001	000	[n]				
DEMANDS(ARRAY)	001	001	001					
DEMANDS(ITEM)	001	001	001	[n]				
EVENT_TIME(ARRAY)	001	001	001	[n]	000			
EVENT_TIME(ITEM)	001	001	001	[n]	000	[n]		
CUM_DEMAND	001	001	001	[n]	001			
CONT_CUM_DEMAND	001	001	001	[n]	002			
DEMAND(ARRAY)	001	001	001	[n]	003			
DEMAND(ITEM)	001	001	001	[n]	003	[n]		
COINCIDENTS(ARRAY)	001	001	002					
COINCIDENTS(ITEM)	001	001	002	[n]				
COINCIDENT_VALUES(ARRAY)	001	001	002	[n]	000			
COINCIDENT_VALUES(ITEM)	001	001	002	[n]	000	[n]		
TIER_DATA_BLOCK(ARRAY)	001	002						
TIER_DATA_BLOCK(ITEM)	001	002	[n]					
SUMMATIONS(ARRAY)	001	002	[n]	000				
SUMMATIONS(ITEM)	001	002	[n]	000	[n]			
DEMANDS(ARRAY)	001	002	[n]	001				
DEMANDS(ITEM)	001	002	[n]	001	[n]			
EVENT_TIME(ARRAY)	001	002	[n]	001	[n]	000		
EVENT_TIME(ITEM)	001	002	[n]	001	[n]	000	[n]	
CUM_DEMAND	001	002	[n]	001	[n]	001		
CONT_CUM_DEMAND	001	002	[n]	001	[n]	002		
DEMAND(ARRAY)	001	002	[n]	001	[n]	003		
DEMAND(ITEM)	001	002	[n]	001	[n]	003	[n]	
COINCIDENTS(ARRAY)	001	002	[n]	002				
COINCIDENTS(ITEM)	001	002	[n]	002	[n]			
COINCIDENT_VALUES(ARRAY)	001	002	[n]	002	[n]	000		
COINCIDENT_VALUES(ITEM)	001	002	[n]	002	[n]	000	[n]	

TABLE 25—PREVIOUS_ DEMAND_RESET_DATA_TBL								
REGISTER_INFO	000							
END_DATE_TIME	000	000						
SEASON	000	001						
PREV_DEMAND_RESET_DATA	001							
NMB_DEMAND_RESETS	001	000						
TOT_DATA_BLOCK	001	001						
SUMMATIONS(ARRAY)	001	001	000					
SUMMATIONS(ITEM)	001	001	000	[n]				
DEMANDS(ARRAY)	001	001	001					
DEMANDS(ITEM)	001	001	001	[n]				
EVENT_TIME(ARRAY)	001	001	001	[n]	000			
EVENT_TIME(ITEM)	001	001	001	[n]	000	[n]		
CUM_DEMAND	001	001	001	[n]	001			
CONT_CUM_DEMAND	001	001	001	[n]	002			
DEMAND(ARRAY)	001	001	001	[n]	003			
DEMAND(ITEM)	001	001	001	[n]	003	[n]		
COINCIDENTS(ARRAY)	001	001	002					
COINCIDENTS(ITEM)	001	001	002	[n]				
COINCIDENT_VALUES(ARRAY)	001	001	002	[n]	000			
COINCIDENT_VALUES(ITEM)	001	001	002	[n[	000	[n]		
TIER_DATA_BLOCK(ARRAY)	001	002						
TIER_DATA_BLOCK(ITEM)	001	002	[n]					
SUMMATIONS(ARRAY)	001	002	[n]	000				
SUMMATIONS(ITEM)	001	002	[n]	000	[n]			
DEMANDS(ARRAY)	001	002	[n]	001				
DEMANDS(ITEM)	001	002	[n]	001	[n]			
EVENT_TIME(ARRAY)	001	002	[n]	001	[n]	000		
EVENT_TIME(ITEM)	001	002	[n]	001	[n]	000	[n]	
CUM_DEMAND	001	002	[n]	001	[n]	001		

CONT_CUM_DEMAND	001	002	[n]	001	[n]	002		
DEMAND(ARRAY)	001	002	[n]	001	[n]	003		
DEMAND(ITEM)	001	002	[n]	001	[n]	003	[n]	
COINCIDENTS(ARRAY)	001	002	[n]	002				
COINCIDENTS(ITEM)	001	002	[n]	002	[n]			
COINCIDENT_VALUES(ARRAY)	001	002	[n]	002	[n]	000		
COINCIDENT_VALUES(ITEM)	001	002	[n]	002	[n]	000	[n]	

TABLE 26—SELF_READ_DATA_TBL								
LIST_STATUS	000							
NBR_VALID_ENTRIES	001							
LAST_ENTRY_ELEMENT	002							
LAST_ENTRY_SEQ_NUM	003							
NBR_UNREAD_ENTRIES	004							
SELF_READS_ENTRIES(ARRAY)	005							
SELF_READS_ENTRIES(ITEM)	005	[n]						
SELF_READ_SEQ_NUM	005	[n]	000					
REGISTER_INFO	005	[n]	001					
END_DATE_TIME	005	[n]	001	000				
SEASON	005	[n]	001	001				
SELF_READ_REGISTER_DATA	005	[n]	002					
NMB_DEMAND_RESETS	005	[n]	002	000				
TOT_DATA_BLOCK	005	[n]	002	001				
SUMMATIONS(ARRAY)	005	[n]	002	001	000			
SUMMATIONS(ITEM)	005	[n]	002	001	000	[n]		
DEMANDS(ARRAY)	005	[n]	001	001	001			
DEMANDS(ITEM)	005	[n]	002	001	001	[n]		
EVENT_TIME(ARRAY)	005	[n]	002	001	001	[n]	000	
EVENT_TIME(ITEM)	005	[n]	002	001	001	[n]	000	[n]
CUM_DEMAND	005	[n]	002	001	001	[n]	001	

CONT_CUM_DEMAND	005	[n]	002	001	001	[n]	002	
DEMAND(ARRAY)	005	[n]	002	001	001	[n]	003	
DEMAND(ITEM)	005	[n]	002	001	001	[n]	003	[n]
COINCIDENTS(ARRAY)	005	[n]	002	001	002			
COINCIDENTS(ITEM)	005	[n]	002	001	002	[n]		
COINCIDENT_VALUES(ARR)	005	[n]	002	001	002	[n]	000	
COINCIDENT_VALUES(ITM)	005	[n]	002	001	002	[n]	000	[n]
TIER_DATA_BLOCK(ARRAY)	005	[n]	003					
TIER_DATA_BLOCK(ITEM)	005	[n]	003	[n]				
SUMMATIONS(ARRAY)	005	[n]	003	[n]	000	000		
SUMMATIONS(ITEM)	005	[n]	003	[n]	000	[n]		
DEMANDS(ARRAY)	005	[n]	003	[n]	001			
DEMANDS(ITEM)	005	[n]	003	[n]	001	[n]		
EVERY_TIME(ARRAY)	005	[n]	003	[n]	001	[n]	000	
EVERY_TIME(ITEM)	005	[n]	003	[n]	001	[n]	000	[n]
CUM_DEMAND	005	[n]	003	[n]	001	[n]	001	
DEMAND(ARRAY)	005	[n]	003	[n]	001	[n]	002	
DEMAND(ITEM)	005	[n]	003	[n]	001	[n]	002	[n]
COINCIDENTS(ARRAY)	005	[n]	003	[n]	002			
COINCIDENTS(ITEM)	005	[n]	003	[n]	002	[n]		
COINCIDENT_VALUES(ARR)	005	[n]	003	[n]	002	[n]	000	
COINCIDENT_VALUES(ITM)	005	[n]	003	[n]	002	[n]	000	[n]

TABLE 027— PRESENT_REGISTER_SELECT_TBL								
PRESENT_DEMAND_SELECT(ARRAY)	000							
PRESENT_DEMAND_SELECT(ITEM)	000	[n]						
PRESENT_VALUE_SELECT(ARRAY)	001							
PRESENT_VALUE_SELECT(ITEM)	001	[n]						

TABLE 028—PRESENT_REGISTER_DATA_TBL								
PRESENT_DEMAND(ARRAY)	000							
PRESENT_DEMAND(ITEM)	000	[n]						
TIME_REMAINING	000	[n]	000					
DEMAND_VALUE	000	[n]	001					
PRESENT_VALUE(ARRAY)	001							
PRESENT_VALUE(ITEM)	001	[n]						

TABLE 030—DIM_DISP_TBL								
DISPLAY_CTRL	000							
NBR_DISP_SOURCES	001							
WIDTH_DISP_SOURCES	002							
NBR_PRI_DISP_LIST_ITEMS	003							
NBR_PRI_DISP_LISTS	004							
NBR_SEC_DISP_LIST_ITEMS	005							
NBR_SEC_DISP_LISTS	006							

TABLE 031—ACT_DISP_TBL								
DISPLAY_CTRL	000							
NBR_DISP_SOURCES	001							
WIDTH_DISP_SOURCES	002							
NBR_PRI_DISP_LIST_ITEMS	003							
NBR_PRI_DISP_LISTS	004							
NBR_SEC_DISP_LIST_ITEMS	005							
NBR_SEC_DISP_LISTS	006							

TABLE 032-DISP_SOURCE_TBL								
DISPLAY_SOURCES(ARRAY)	000							
DISPLAY_SOURCES(ITEM)	000	[n]						
DISPLAY_SOURCE(ARRAY)	000	[n]	000					

DISPLAY_SOURCE(ITEM)	000	[n]	000	[n]				

TABLE 033—PRI_DISP_LIST_TBL								
PRI_DISP_LIST(ARRAY)	000							
PRI_DISP_LIST(ITEM)	000	[n]						
DISP_SCROLL1	000	[n]	000					
DISP_SCROLL2	000	[n]	001					
NBR_LIST_ITEMS	000	[n]	002					
PRI_DISP_SOURCES(ARRAY)	001							
PRI_DISP_SOURCES(ITEM)	001	[n]						

TABLE 034—SEC_DISP_LIST_TBL								
SEC_DISP_LIST(ARRAY)	000							
SEC_DISP_LIST(ITEM)	000	[n]						
DISP_SCROLL1	000	[n]	000					
DISP_SCROLL2	000	[n]	001					
NBR_LIST_ITEMS	000	[n]	002					
SEC_DISP_SOURCES(ARRAY)	001							
SEC_DISP_SOURCES(ITEM)	001	[n]						

TABLE 040— DIM_SECURITY_LIMITING_TBL								
NBR_PASSWORDS	000							
PASWWORD_LEN	001							
NBR_KEYS	002 001							
KEY_LEN	003 002							
NBR_PERM_USED	004 003							

TABLE 041—ACT_SECURITY_LIMITING_TBL								
NBR_PASSWORDS	000							
PASWWORD_LEN	001							
NBR_KEYS	002 001							
KEY_LEN	003 002							
NBR_PERM_USED	004 003							

TABLE 042—SECURITY_TBL								
SECURITY_ENTRIES(ARRAY)	000							
SECURITY_ENTRIES(ITEM)	000	[n]						
PASSWORD	000	[n]	000					
ACCESS_PERMISSIONS	000	[n]	001					

TABLE 043—DEFAULT_ACCESS_CONTROL_TBL								
TABLE_DEFAULT	000							
ACCESS_TABLE_DEFAULT	000	000						
READ	000	001						
WRITE	000	002						
PROCEDURE_DEFAULT	001							
ACCESS_TABLE_DEFAULT	001	000						
READ	001	001						
WRITE	001	002						

TABLE 044—ACCESS_CONTROL_TBL								
ACCESS_CONTROL(ARRAY)	000							
ACCESS_CONTROL(ITEM)	000	[n]						
ACCESS_TABLE_DEF	000	[n]	000					
READ	000	[n]	001					

WRITE	000	[n]	002					
TABLE 045—KEY_TBL								
KEY_ENTRIES(ARRAY)	000							
KEY_ENTRIES(ITEM)	000	[n]						
KEY(ARRAY)	000	[n]	000					
KEY(ITEM)	000	[n]	000	[n]				

TABLE 050—DIM_TIME_TOU_TBL								
TIME_FUNC_FLAG1	000							
TIME_FUNC_FLAG2	001							
CALENDAR_FUNC	002							
NBR_NON_RECURR_DATES	003							
NBR_RECURR_DATES	004							
NBR_TEIR_SWITCHES	005							
CALENDAR_TBL_SIZE	006							

TABLE 051—ACT_TIME_TOU_TBL								
TIME_FUNC_FLAG1	000							
TIME_FUNC_FLAG2	001							
CALENDAR_FUNC	002							
NBR_NON_RECURR_DATES	003							
NBR_RECURR_DATES	004							
NBR_TEIR_SWITCHES	005							
CALENDAR_TBL_SIZE	006							

TABLE 052—CLOCK_TBL								
CLOCK_CALENDAR	000							
TIME_DATE_QUAL	001							
TABLE 053—TIME_OFFSET_TBL								
DST_TIME_EFF	000							

DST_TIME_AMT	001							
TIME_ZONE_OFFSET	002							

TABLE 054—CALENDAR_TBL								
ANCHOR_DATE	000							
NON_RECURR_DATES(ARRAY)	001							
NON_RECURR_DATES(ITEM)	001	[n]						
NON_RECURR_DATE	001	[n]	000					
CALENDAR_ACTION	001	[n]	001					
RECURR_DATES(ARRAY)	002							
RECURR_DATES(ITEM)	002	[n]						
RECURR_DATE	002	[n]	000					
CALENDAR_ACTION	002	[n]	001					
TIER_SWITCHES(ARRAY)	003							
TIER_SWITCHES(ITEM)	003	[n]						
TIER_SWITCH	003	[n]	000					
DAY_SCH_NUM	003	[n]	001					
DAY_SCHEDULE_ID_MATRIX(ARRAY)	004							
DAY_SCHEDULE_ID_MATRIX(ITEM)	004	[n]						

TABLE 055—CLOCK_STATE_TBL								
CLOCK_CALANDER	000							
TIME_DATE_QUAL	001							
STATUS	002							

TABLE 056—TIME_REMAIN_TBL								
SUMM_TIER_TIME_REMAIN	000							
DEMAND_TIER_TIME_REMAIN	001							
TIER_TIME_REMAIN	002							
SELF_READ_DAYS_REMAIN	003							

TABLE 060—DIM_LP_TBL								
LP_MEMORY_LEN	000							
LP_FLAGS	001							
LP_FMATS	002							
NBR_BLKS_SET1	003							
NBR_BLK_INTS_SET1	004							
NBR_CHNS_SET1	005							
MAX_INT_TIME_SET1	006							
NBR_BLKS_SET2	007							
NBR_BLK_INTS_SET2	008							
NBR_CHNS_SET2	009							
MAX_INT_TIME_SET2	010							
NBR_BLKS_SET3	011							
NBR_BLK_INTS_SET3	012							
NBR_CHNS_SET3	013							
MAX_INT_TIME_SET3	014							
NBR_BLKS_SET4	015							
NBR_BLK_INTS_SET4	016							
NBR_CHNS_SET4	017							
MAX_INT_TIME_SET4	018							

TABLE 061—ACT_LP_TBL								
LP_MEMORY_LEN	000							
LP_FLAGS	001							
LP_FMATS	002							
NBR_BLKS_SET1	003							
NBR_BLK_INTS_SET1	004							
NBR_CHNS_SET1	005							
MAX_INT_TIME_SET1	006							

NBR_BLKS_SET2	007							
NBR_BLK_INTS_SET2	008							
NBR_CHNS_SET2	009							
MAX_INT_TIME_SET2	010							
NBR_BLKS_SET3	011							
NBR_BLK_INTS_SET3	012							
NBR_CHNS_SET3	013							
MAX_INT_TIME_SET3	014							
NBR_BLKS_SET4	015							
NBR_BLK_INTS_SET4	016							
NBR_CHNS_SET4	017							
MAX_INT_TIME_SET4	018							

TABLE 062—LP_CNTL_TBL								
LP_SEL_SET1(ARRAY)	000							
LP_SEL_SET1(ITEM)	000	[n]						
CHNL_FLAG	000	[n]	000					
INT_SOURCE_SELECT	000	[n]	001					
END_BLK_RDG_SOURCE_SELECT	000	[n]	002					
INT_FMT_CDE1	001							
SCALARS_SET1	002							
DIVISORS_SET1	003							
LP_SEL_SET2(ARRAY)	004							
LP_SEL_SET2(ITEM)	004	[n]						
CHNL_FLAG	004	[n]	000					
INT_SOURCE_SELECT	004	[n]	001					
END_BLK_RDG_SOURCE_SELECT	004	[n]	002					
INT_FMT_CDE2	005							
SCALARS_SET2	006							
DIVISORS_SET2	007							

LP_SEL_SET3(ARRAY)	008							
LP_SEL_SET3(ITEM)	008	[n]						
CHNL_FLAG	008	[n]	000					
INT_SOURCE_SELECT	008	[n]	001					
END_BLK_RDG_SOURCE_SELECT	008	[n]	002					
INT_FMT_CDE3	009							
SCALARS_SET3	010							
DIVISORS_SET3	011							
LP_SEL_SET4(ARRAY)	012							
LP_SEL_SET4(ITEM)	012	[n]						
CHNL_FLAG	012	[n]	000					
INT_SOURCE_SELECT	012	[n]	001					
END_BLK_RDG_SOURCE_SELECT	012	[n]	002					
INT_FMT_CDE4	013							
SCALARS_SET4	014							
DIVISORS_SET4	015							

TABLE 063—LP_STATUS_TBL								
LP_STATUS_SET1	000							
LP_SET_STATUS_FLAGS	000	000						
NBR_VALID_BLOCKS	000	001						
LAST_BLOCK_ELEMENT	000	002						
LAST_BLOCK_SEQ_NUM	000	003						
NBR_UNREAD_BLOCKS	000	004						
NBR_VALID_INT	000	005						
LP_STATUS_SET2	001							
LP_SET_STATUS_FLAGS	001	000						
NBR_VALID_BLOCKS	001	001						
LAST_BLOCK_ELEMENT	001	002						
LAST_BLOCK_SEQ_NUM	001	003						

NBR_UNREAD_BLOCKS	001	004						
NBR_VALID_INT	001	005						
LP_STATUS_SET3	002							
LP_SET_STATUS_FLAGS	002	000						
NBR_VALID_BLOCKS	002	001						
LAST_BLOCK_ELEMENT	002	002						
LAST_BLOCK_SEQ_NUM	002	003						
NBR_UNREAD_BLOCKS	002	004						
NBR_VALID_INT	002	005						
LP_STATUS_SET4	003							
LP_SET_STATUS_FLAGS	003	000						
NBR_VALID_BLOCKS	003	001						
LAST_BLOCK_ELEMENT	003	002						
LAST_BLOCK_SEQ_NUM	003	003						
NBR_UNREAD_BLOCKS	003	004						
NBR_VALID_INT	003	005						

TABLE 064—LP_DATA_SET1_TBL								
LP_DATA_SETS1(ARRAY)	000							
LP_DATA_SETS1(ITEM)	000	[n]						
BLK_END_TIME	000	[n]	000					
END_READINGS(ARRAY)	000	[n]	001					
END_READINGS(ITEM)	000	[n]	001	[n]				
BLOCK_END_READ	000	[n]	001	[n]	000			
BLOCK_END_PLUSE	000	[n]	001	[n]	001			
SIMPLE_INT_STATUS	000	[N]	002					
LP_INT(ARRAY)	000	[n]	003002					
LP_INT(ITEM)	000	[n]	003002	[n]				
EXTENDED_INT_STATUS(ARRAY)	000	[n]	003002	[n]	000			

EXTENDED_INT_STATUS(ITEM)	000	[n]	003002	[n]	000	[n]		
INT_DATA(ARRAY)	000	[n]	003002	[n]	001			
INT_DATA(ITEM)	000	[n]	003002	[n]	001	[n]		
ITEM(UINT8)	000	[n]	003	[n]	001	[n]	000	
ITEM(UINT16)	000	[n]	003	[n]	001	[n]	001	
ITEM(UINT32)	000	[n]	003	[n]	001	[n]	002	
ITEM(INT8)	000	[n]	003	[n]	001	[n]	003	
ITEM(INT16)	000	[n]	003	[n]	001	[n]	004	
ITEM(INT32)	000	[n]	003	[n]	001	[n]	005	
ITEM(NI_FMAT1)	000	[n]	003	[n]	001	[n]	006	
ITEM(NI_FMAT2)	000	[n]	003	[n]	001	[n]	007	

TABLE 065—LP_DATA_SET2_TBL								
LP_DATA_SETS2(ARRAY)	000							
LP_DATA_SETS2(ITEM)	000	[n]						
BLK_END_TIME	000	[n]	000					
END_READINGS(ARRAY)	000	[n]	001					
END_READINGS(ITEM)	000	[n]	001	[n]				
BLOCK_END_READ	000	[n]	001	[n]	000			
BLOCK_END_PLUSE	000	[n]	001	[n]	001			
SIMPLE_INT_STATUS	000	[n]	002					
LP_INT(ARRAY)	000	[n]	003002					
LP_INT(ITEM)	000	[n]	003002	[n]				
EXTENDED_INT_STATUS(ARRAY)	000	[n]	003002	[n]	000			
EXTENDED_INT_STATUS(ITEM)	000	[n]	003002	[n]	000	[n]		
INT_DATA(ARRAY)	000	[n]	003002	[n]	001			
INT_DATA(ITEM)	000	[n]	003002	[n]	001	[n]		
ITEM(UINT8)	000	[n]	003	[n]	001	[n]	000	
ITEM(UINT16)	000	[n]	003	[n]	001	[n]	001	

ITEM(UINT32)	000	[n]	003	[n]	001	[n]	002	
ITEM(INT8)	000	[n]	003	[n]	001	[n]	003	
ITEM(INT16)	000	[n]	003	[n]	001	[n]	004	
ITEM(INT32)	000	[n]	003	[n]	001	[n]	005	
ITEM(NI_FMAT1)	000	[n]	003	[n]	001	[n]	006	
ITEM(NI_FMAT2)	000	[n]	003	[n]	001	[n]	007	

TABLE 066—LP_DATA_SET3_TBL								
LP_DATA_SETS3(ARRAY)	000							
LP_DATA_SETS3(ITEM)	000	[n]						
BLK_END_TIME	000	[n]	000					
END_READINGS(ARRAY)	000	[n]	001					
END_READINGS(ITEM)	000	[n]	001	[n]				
BLOCK_END_READ	000	[n]	001	[n]	000			
BLOCK_END_PLUSE	000	[n]	001	[n]	001			
SIMPLE_INT_STATUS	000	[n]	002					
LP_INT(ARRAY)	000	[n]	003 002					
LP_INT(ITEM)	000	[n]	003 002	[n]				
EXTENDED_INT_STATUS(ARRAY)	000	[n]	003 002	[n]	000			
EXTENDED_INT_STATUS(ITEM)	000	[n]	003 002	[n]	000	[n]		
INT_DATA(ARRAY)	000	[n]	0030 02	[n]	001			
INT_DATA(ITEM)	000	[n]	0030 02	[n]	001	[n]		
ITEM(UINT8)	000	[n]	003	[n]	001	[n]	000	
ITEM(UINT16)	000	[n]	003	[n]	001	[n]	001	
ITEM(UINT32)	000	[n]	003	[n]	001	[n]	002	
ITEM(INT8)	000	[n]	003	[n]	001	[n]	003	
ITEM(INT16)	000	[n]	003	[n]	001	[n]	004	

ITEM(INT32)	000	[n]	003	[n]	001	[n]	005	
ITEM(NI_FMAT1)	000	[n]	003	[n]	001	[n]	006	
ITEM(NI_FMAT2)	000	[n]	003	[n]	001	[n]	007	

TABLE 067—LP_DATA_SET4_TBL								
LP_DATA_SETS4(ARRAY)	000							
LP_DATA_SETS4(ITEM)	000	[n]						
BLK_END_TIME	000	[n]	000					
END_READINGS(ARRAY)	000	[n]	001					
END_READINGS(ITEM)	000	[n]	001	[n]				
BLOCK_END_READ	000	[n]	001	[n]	000			
BLOCK_END_PLUSE	000	[n]	001	[n]	001			
SIMPLE_INT_STATUS	000	[n]	002					
LP_INT(ARRAY)	000	[n]	003002					
LP_INT(ITEM)	000	[n]	003002	[n]				
EXTENDED_INT_STATUS(ARRAY)	000	[n]	003002	[n]	000			
EXTENDED_INT_STATUS(ITEM)	000	[n]	003002	[n]	000	[n]		
INT_DATA(ARRAY)	000	[n]	003002	[n]	001			
INT_DATA(ITEM)	000	[n]	003002	[n]	001	[n]		
ITEM(UINT8)	000	[n]	003	[n]	001	[n]	000	
ITEM(UINT16)	000	[n]	003	[n]	001	[n]	001	
ITEM(UINT32)	000	[n]	003	[n]	001	[n]	002	
ITEM(INT8)	000	[n]	003	[n]	001	[n]	003	
ITEM(INT16)	000	[n]	003	[n]	001	[n]	004	
ITEM(INT32)	000	[n]	003	[n]	001	[n]	005	
ITEM(NI_FMAT1)	**000**	**[n]**	**003**	**[n]**	**001**	**[n]**	**006**	
ITEM(NI_FMAT2)	**000**	**[n]**	**003**	**[n]**	**001**	**[n]**	**007**	

TABLE 070—DIM_LOG_TBL								
LOG_FLAGS	001							
NBR_STD_EVENTS	002							
NBR_MFG_EVENTS	003							
HIST_DATA_LENGTH	004							
EVENT_DATA_LENGTH	005							
NBR_HISTORY_ENTRIES	006							
NBR_EVENT_ENTRIES	007							

TABLE 071—ACT_LOG_TBL								
LOG_FLAGS	001							
NBR_STD_EVENTS	002							
NBR_MFG_EVENTS	003							
HIST_DATA_LENGTH	004							
EVENT_DATA_LENGTH	005							
NBR_HISTORY_ENTRIES	006							
NBR_EVENT_ENTRIES	007							

TABLE 072—EVENTS_ID_TBL								
STD_EVENTS_SUPPORTED	000							
MFG_EVENTS_SUPPORTED	001							

TABLE 073—HISTORY_LOG_CTRL_TBL								
STD_EVENTS_MONITORED_FLAGS	000							
MFG_EVENTS_MONITORED_FLAGS	001							
STD_TBLS_MONITORED_FLAGS	002							
MFG_TBLS_MONITORED_FLAGS	003							
STD_PROC_MONITORED_FLAGS	004							
MFG_PROC_MONITORED_FLAGS	005							

TABLE 074—HISTORY_LOG_DATA_TBL								
HIST_FLAGS	000							
NBR_VALID_ENTRIES	001							
LAST_ENTRY_ELEMENT	002							
LAST_ENTRY_SEQ_NBR	003							
NBR_UNREAD_ENTRIES	004							
ENTRIES(ARRAY)	005							
ENTRIES(ITEM)	005	[n]						
HISTORY_TIME	005	[n]	000					
EVENT_NUMBER	005	[n]	001					
HISTORY_SEQ_NBR	005	[n]	002					
USER_ID	005	[n]	003					
HISTORY_CODE	005	[n]	004					
HISTORY_ARGUMENT(ARRAY)	005	[n]	005					
HISTORY_ARGUMENT(ITEM)	005	[n]	005	[n]				

TABLE 075—EVENT_LOG_CTRL_TBL								
STD_EVENTS_MONITORED_FLAGS	000							
MFG_EVENTS_MONITORED_FLAGS	001							
STD_TBLS_MONITORED_FLAGS	002							
MFG_TBLS_MONITORED_FLAGS	003							
STD_PROC_MONITORED_FLAGS	004							
MFG_PROC_MONITORED_FLAGS	005							

TABLE 076—EVENT_LOG_DATA_TBL								
EVENT_FLAGS	000							
NBR_VALID_ENTRIES	001							
LAST_ENTRY_EMLEMENT	002							
LAST_ENTRY_SEQ_NBR	003							
NBR_UNREAD_ENTRIES	004							

ENTRIES(ARRAY)	005							
ENTRIES(ITEM)	005	[n]						
EVENT_TIME	005	[n]	000					
EVENT_NUMBER	005	[n]	001					
EVENT_SEQ_NBR	005	[n]	002					
USER_ID	005	[n]	003					
EVENT_CODE	005	[n]	004					
EVENT_ARGUMENT(ARRAY)	005	[n]	005					
EVENT_ARGUMENT(ITEM)	005	[n]	005	[n]				

TABLE 080 - DIM_UDT_FUNC_LIM_TBL								
NBR_XFR_LIST_ITEMS	000							
UDT_FUNC_CTRL	001							
MAX_INSTANCE	002							
UDT_0_SIZE	003							
UDT_1_SIZE	004							
UDT_2_SIZE	005							
UDT_3_SIZE	006							
UDT_4_SIZE	007							
UDT_5_SIZE	008							

TABLE 081—ACT_UDT_FUNC_LIM_TBL								
NBR_XFR_LIST_ITEMS	000							
UDT_FUNC_CTRL	001							
MAX_INSTANCE	002							
UDT_0_SIZE	003							
UDT_1_SIZE	004							
UDT_2_SIZE	005							
UDT_3_SIZE	006							
UDT_4_SIZE	007							

UDT_5_SIZE	008							

TABLE 082 - UDT_LIST_TBL								
UDT_LIST(ARRAY) (BYTE/OFFSET)	000							
UDT_LIST(ITEM)	000	[n]						

TABLE 083 - UDT_SET_TBL								
UDT_DATA_SETS(ARRAY)	000							
UDT_DATA_SETS(ITEM)	000	[n]						
FIRST_ITEM_NUMBER	000	[n]	000					
LAST_ITEM_NUMBER	000	[n]	001					

TABLE 084 - UDT_0_TBL								
UDT_0_TBL(ARRAY)	000							
UDT_0_TBL(ITEM)	000	[n]						

TABLE 085 - UDT_1_TBL								
UDT_1_TBL(ARRAY)	000							
UDT_1_TBL(ITEM)	000	[n]						

TABLE 086 - UDT_2_TBL								
UDT_2_TBL(ARRAY)	000							
UDT_2_TBL(ITEM)	000	[n]						

TABLE 087 - UDT_3_TBL								
UDT_3_TBL(ARRAY)	000							
UDT_3_TBL(ITEM)	000	[n]						
TABLE 088 - UDT_4_TBL								
UDT_4_TBL(ARRAY)	000							
UDT_4_TBL(ITEM)	000	[n]						

TABLE 089 - UDT_5_TBL								
UD_5_TBL(ARRAY)	000							
UDT_5_TBL(ITEM)	000	[n]						